www.tutorial4us.com

Tutorial4Us

College Projects

```
void main()
{
 int no;
 clrscr();
 printf("Hello");
}
```

Basic Program

C Tutorial

JAVA Programming

tutorial4us.com

A Perfect Place for All Tutorials Resources

Core Java | Servlet | JSP | JDBC | Struts | Hibernate | Spring

Java Projects | C | C++ | DS | Interview Questions | JavaScript

College Projects | eBooks | Interview Tips | Forums | Java Discussions

For More Tutorials Stuff Visit

www.tutorial4us.com

A Perfect Place for All Tutorials Resources

Core Java

(K V Rao Notes)

www.tutorial4us.com

Date / /
Page

~~Introduction~~

Architecture Required for Development of

Standalone Application - Any prog runs in the context of local disk, whose results are not shairable.

Distribcted Application - are those which runs in the context of browser.
e.g. www.google.com.

Two companies come forword for distribcted apps - Internet s/w

SUN → JAVA

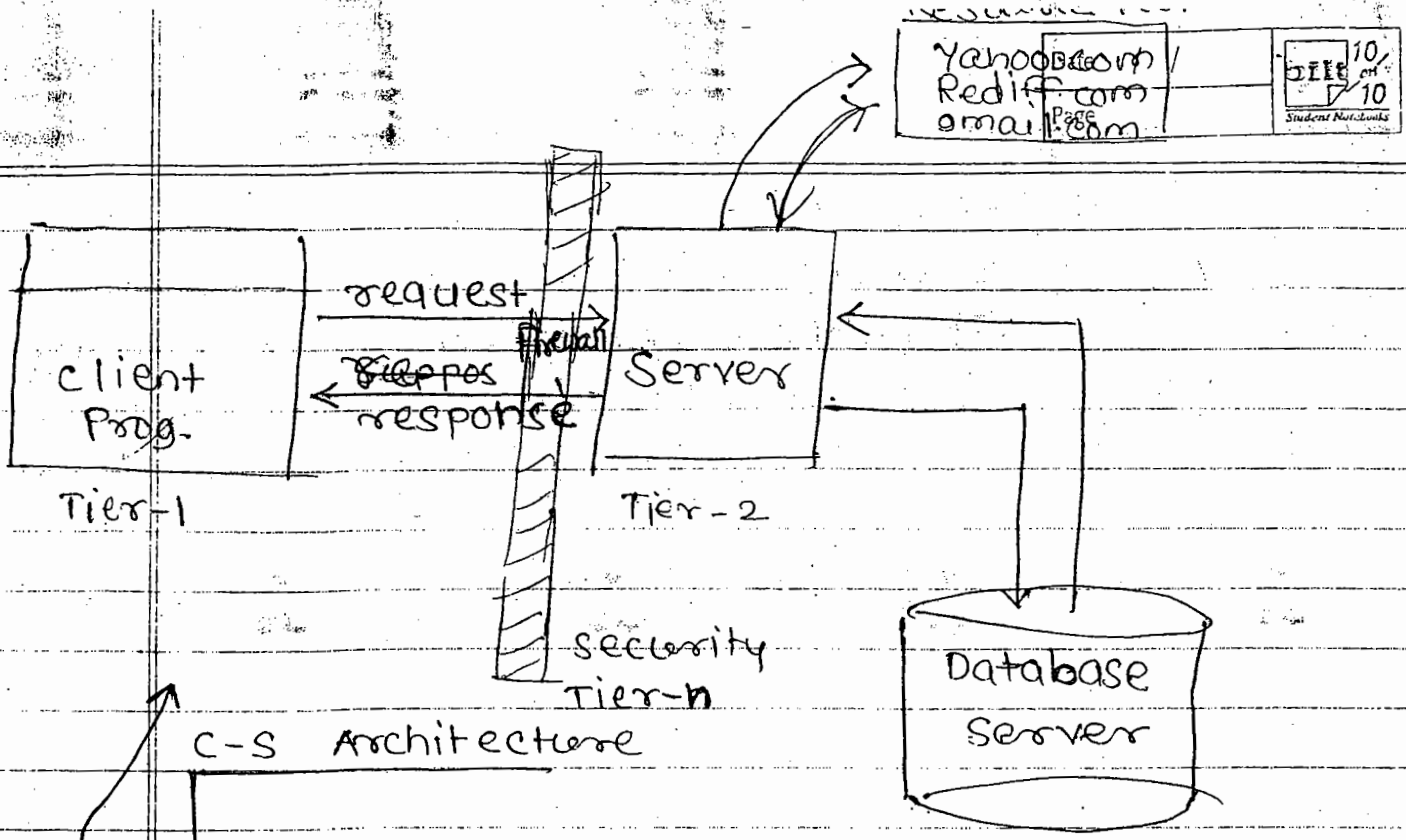
microsoft → .Net

JAVA
J2SE
J2EE
J2ME

Distribcted apps are those which run in the context of browser or world wide web and whose results are shairable across the globe.

In real world all websites comes un distribcted applications.

According to industry standards to develop distribcted apps we follow a well known architecture called client-server architecture.



C-S Architecture

- 2-Tier Arch (CP, SP)
- 3- Tier Arch (CP, SP, DB)
- n - Tier Arch (CP, FP, SP, DB)

The following diagram gives the workflow in the client server architecture.

According to the industry standard, client-server architecture is having 3 types. They are

1) 2-tier architecture - It contains,
   - (a) set of client progs
   - (b) set of server -//-

2) 3-tier archi - It contains,

(c) set of database server progs.

(3) n-tier architecture- It contains,
(a) set of client progs.
(b) set of firewall progs.
(c) set of server progs.
(d) set of database server progs.

A client is a java program which always make a request to get the services from server side programs.

In real world applications, group of software engineers work on development of client side applications.

A server is also a program or software which will accept a client request, process client request and gives response back to clients.

The basic advantage of server softwar is that to get concurrent access.

The following table gives, the real world software name under their vendors

server s/w Name	vendor name
Tom cat	Apache Jakartha s/w Foundath
Weblogic	BEA corporation
Websphere	IBM
glassfish	SUN microsystem
oracle 10g	Oracle corporation etc.
Pramati	Pramati s/w services (Hyd based)

Firewall programs are those which are always validating client request to check wheather they are virus oriented or virus free.

In simple words firewall programs are known as security programs.

Resource pool is the one which is available in the server softwar for placing server side programs.

Database softcoare is the one which is used for storing the data permanently with effective security.

In most of the industry java projects next use of oracle databse.

Protocol- A protocal is the set of rules used for exchanging data bet' client & server applications.

currenly the internet world is using a predefined protocal called http.

Java is one of the programming language /technology used for developing distributed applications by making use of client server architecture.

Java language developed at SUN microsyste. in the year 1990 under the guidance of James Gosling and others.

Originally sun microsystem is one of the acedemic university (Standford university Network) developed the rules for JAVA & those rules are programatically implemented as developed by Javasoft INC, USA. Javasoft is one of the software division of Sunmicr system.

Whatever the software developed in the year 1990, SUN microsystem has released on the name of 'OAK', which is original name of JAVA (Scientifically OAK is one of the tree name). The OAK has taken 18 months to develop (1½ year).

The software OAK is able to fulfill few requirements of the industry and unable to fulfill some other requirements of the industry.

The software OAK revised at SUN microsystem under the guidance of James Gosling, and released to the industry on the name of JAVA in the year 1995. ~~JAVA~~ scientificaly JAVA is one of the coffiee seed name.

The software JAVA is available in the industry in 3 categories, they are

J2SE – JAVA 2 standard Edition
J2EE – JAVA 2 Enterprise – "–
J2ME – Java 2 micro / mobile – "–

J2SE is used for developing client side applications.

J2EE used for developing server side applications.

To exchange the data between client side applications and server side applications we use a protocol called http. In general to exchange the data between J2SE & J2EE applications we use a predefined protocol called HTTP.

J2ME is used for developing mobile/wireless applications by making use of a predifined protocol called WAP (Wireless Access/Application protocol).

Q. What are the differences between ftp and http?

	ftp	http
1)	ftp comes under UDP	HTTP comes under TCP
2)	FTP is one of the non acknowledgement oriented protocol	HTTP is one of the acknowledgement oriented protocol.
3)	Ftp is one of the stateful protocol	http is one of the stateless protocol.

Q. Define stateful protocol and stateless protocol

→ A stateful is protocol is one which maintains an identity of a client forever.

A stateless protocol is one which maintain an identity of client for a limited period of time.

It is highly recomanded for the programmers to develop online application with stateless protocols but not with stateful protocols.

by SUN		Industry Experts
~~Tech~~ Tech version	S/w version	Performance comparison.
JAVA	JDK 1.0 JDK 1.1	Tortoise
JAVA 2	JDK 1.2	Dog
~~JAVA 2~~	JDK 1.3 JDK 1.4	Horse
JAVA 5	JDK 1.5	Tiger S/w (SUN)
	JDK 1.6	Mustang (car) (FORD icon)

The above table gives soft technolgy vesion of java, software vesion of java & performance comparison.

As an today J2SE is known as JSE.

J2EE is known as JEE.(Java EE).

J2ME is also known as JME.

24/01/2011

Features or buzzwords of Java

Features of a language are nothing but the set of services or facilities provided by the language vendors to the industry programmers.

Java language provides 13 features to the industry programmers to develop various applications they are,

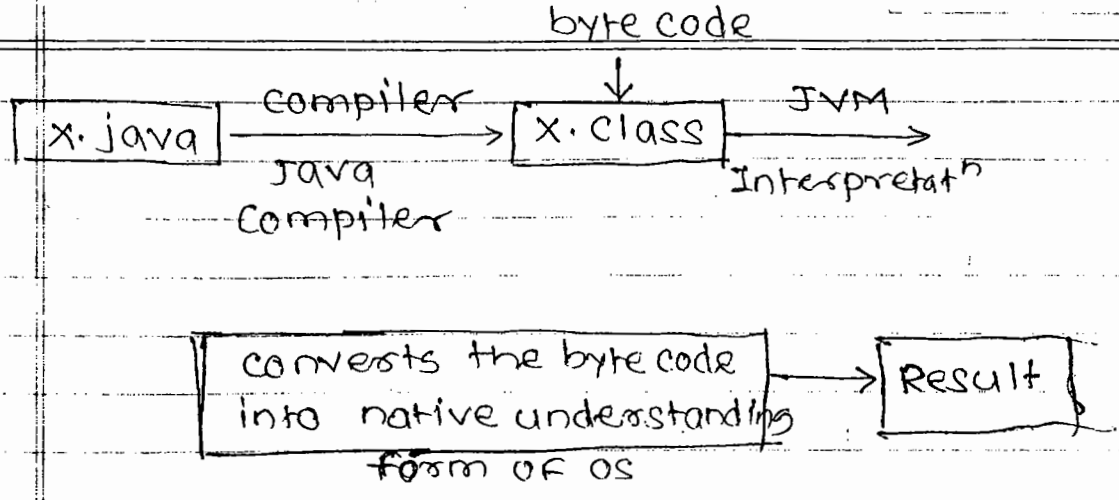
1) simple
2) Platform independent
3) Architectural neutral.
4) Portable
5) Multithreaded.
6) Networked.
7) Distributed.
8) High performace
9) Robust
10) Dynamic
11) Interpreted
12) secured
13) OOPL (Object oriented Prog. Lang.).

Simple-

Java is one of the simple programming language because of the following points

a) Java programming does not support a complex concept called pointers in other words, every java program is totally free from pointers so that coe get application development time is very less and application execution time is also less because of magic of byte code.

Let us consider the following diagram:



Q. Define bytecode & JVM?

→ Byte code is the set of optimised instructions generated during compilation phase & it is more powerful than ordinary pointer code.

JVM is set of programs developed by Sunmicrosystem & supplied as a part of jdk for reading line by line of byte code & it converts into native understanding form of O.S.

Java language is one of the compiled & interpreted programming language.

b) Memory allocation -

(3) Java programming execution environment contain inbuilt garbage collection program for automatic memory management for strengthing the performace of Java / J2EE applications.

Q. Define a Guabage collector.

→ Garbage collector is one of the backgrou und system java program which is running internally along with our regular java program for collecting unused / unreferenced memory space for improving the performace of our applications.

Note- Java programming does not support destructor concept in place of destructors, we have guarbage collector program.

Tuesday, 25/01/2011

(c) Java programming is containing rich set of API (Application Programming Interface).

Q. Define an API?

→ An API is a collection of packages. A Package is ~~a~~ the collection of classes, interfaces and subpacrges. A subpackage is a collection of classes interfaces and subsubpackages etc.

(d) Java programming is containing userfriendly syntaxes so that we can develop effective applications. In other words if any language is providing user friendly syntaxes, we can develop errorfree applications.

The

2) Platform Independent -

A plateforom independent technology / language related applications runs on every OS,

without considering their vendors.

In order to say particular language is plateform independent if and only if it has to satisfy the following properties.

a) The language must have data types & they must take same amount of memory space.

b) The language must contain some special internal programs which will convert from one format of one OS to another format of another OS.

The languages like C, C++ are treated as plateform dependent applications. They cause;

a) The data types of C, C++ takes different amount of memory space on different OS.

b) The C, C++ softwares does not contain any special programs for converting one format of one OS to another format of another OS.

The language like java is treated as platform independent language because,

a) All the datatypes of java language takes the same amount of memory space on all OS.

b) The software of java (JDK) contains some internal programs which

will convert from one format of one OS to another format of another OS.

Note - The OS DOS understands every language program in its understanding format called MOZARTA where as UNIX operating System understands everything in embeded linking format.

3) Architectural Neutral -

An application is said to be architectural neutral if an only if the application runs on every processor without considering their architectures and vendors (providers).

A language is said to be architectural neutral, it has to satisfy the following property;

The software must have the special internal programs which will convert the factors of one processor to factors of another processor.

The languages like C, C++ etc are considered architectural dependent because the above property is not satisfied.

The language like java is one of the architectural neutral language because it satisfies the above property.

The basic aim of sun microsystem is that java applications must run without considering about software benchmarks and hardware benchmarks.

④ Portable- A portable application is one which will run on every operating system and on every processor without considering their vendors and architectures.

The applications of C, C++, Pascal etc are treated as non portable applications.

All the java applications are treated as portable applications.

According to SUN Microsystem-

Portability = Platform Independent
Architectural Neutral.

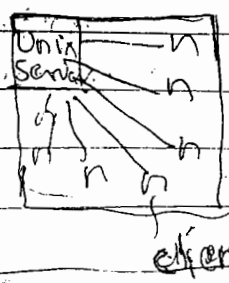
Conclusion-

It is highly recommanded in the industry to distributed applications with portable softwares only but not by using non portable softwares.

⑤ Multithreading-

	by (default)	
* Java program - (Foreground thread)	n	
* FGT / child	1	Multiple
* BGT / Parent (Background Thread)	1	1

* multiTasking is one of the specialized form of OS.



Multithreading is one of the distinct facility of java programming.

The basic aim of multithreading is that to achive concurrent execution.

A flow of control is known as thread.

The basic uses of thread is to execute user/programmer defined functions.

If any java program is containing multiple flow of controls then that java program is known as multithreaded.

The languages like C, C++, paskal, Cobol etc are treated as single threaded modeling languages. Because their programming runtime environment contains single flow of control.

With a single threaded modelling languages we can achive only sequential execution but not concurrent execution and these languages does not contain any library for development of multithreading applications.

The languages like java & .Net are treated as multithreaded modelling languages Because their execution environment contains multiple flow of controls.

Multithreaded modelling languages provides both sequential & concurrent execut^{n} Multithreading langu programming languages provides a library for development of multithreading applications.

In java programming multithedling applications are developed by following,

① java.lang.thread (class).

(b) java.lang.Runnable (Interface)

The realtime implimentat^n of multithreading concept is that to develop realtime or real world servers such as Tomcat, weblogic, ~~webtogic~~, websphere etc.

Whenever we write any java program there exists two types of threads, they are

1. Foreground / child
2. Background / parent

A Foreground thread is one which always executes logic of the java program or user/programmer defined methods of java.

A Background thread is one which always monitors execution status of foreground threads.

* By default there exist single Foreground thread & single background thread in every java program.

Programmitically there is possibility of creating multiple foreground threads and recommanded to have single background thread.

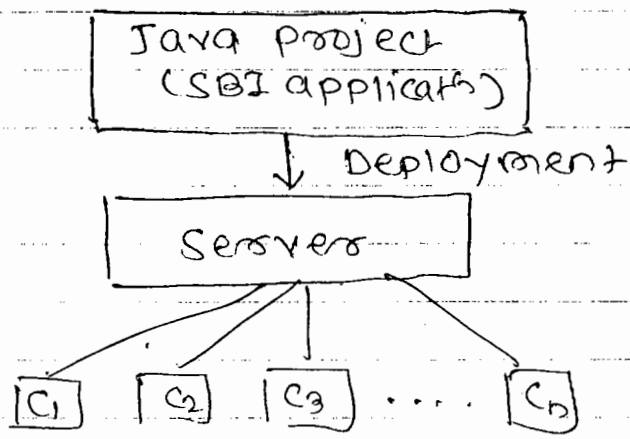
Multithreading is one of the specialized form multitasking ~~form~~ concept of operating system.

Q. How do you justify "Every java program is multithreaded"?

A centralized application is one which runs in the contex of single server, and it can be accessible across the globe.

e.g. All banking softwares are comes under centralized applications.

The digramatic representation of centralized application is shown below.



The basic limitation of centralized application is that once the server is down, clients are unable to communicate unless and until it is reloaded again.

This makes us to understand; centralized applications makes to understand less availability of data.

All the centralized applications in the industry are not containing public URL.

Distributed applications are those which runs in the context of multiple servers and they can be accessible across

internet & intranet applications and they are running untrusted and trusted networks so that industry is treated Java is one of the networked programming language.

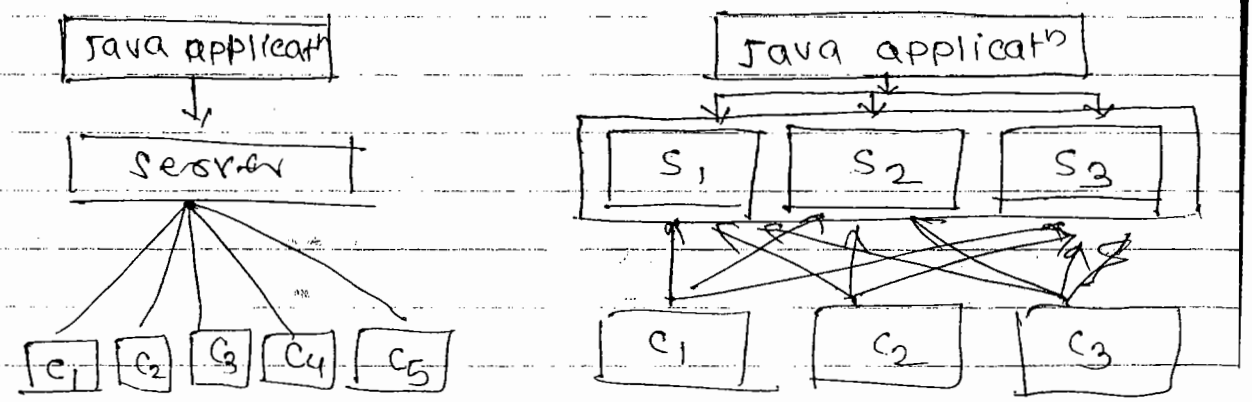
Note:- If any machine does not have storing & processing capability then that machine is known as non autonomous otherwise it is known as autonomous.

Q. Define intranet application & internet applicat?

→ An intranet application is one which can be accessed in limited distance.

An internet application is one which can be accessed across the globe.

3. Distributed :-



According to the industry standards, in java realtime applications, applicat's are classified into two types they are,

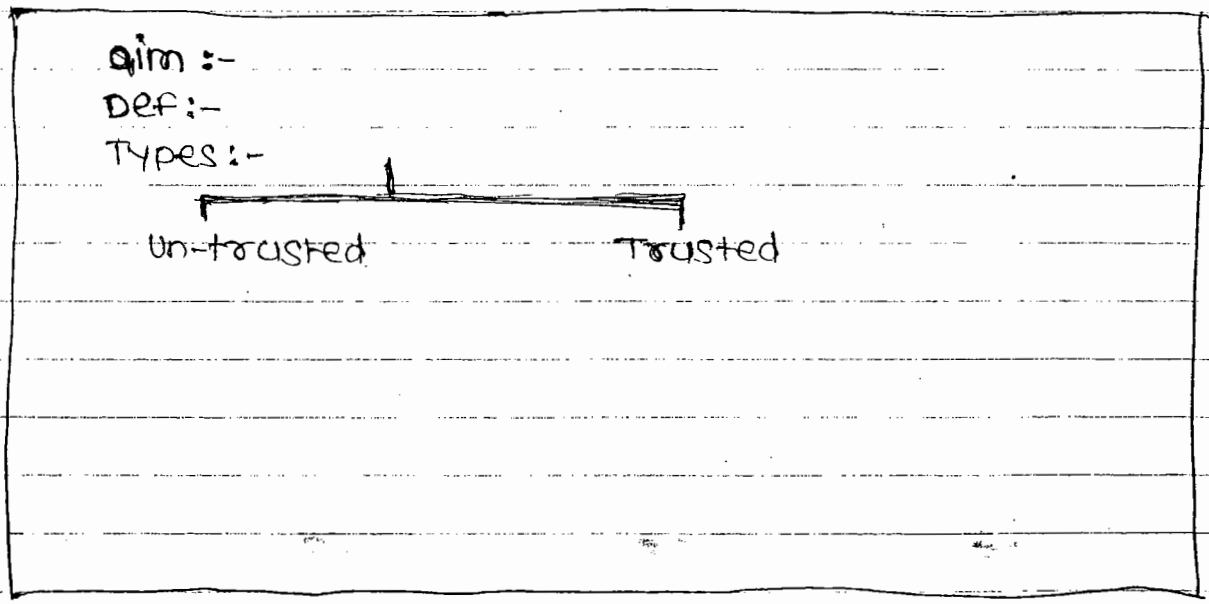
1. centralized applications.
2. Distributed applications.

When we run the java program, For executing the logic of our java program, there exist one thread known as foreground thread. To monitor the status of foreground thread, there exist one more background java program known as guarbage collector.

For executing guarbage collector there exists one thread known as background thread hence in java programming exect^n environment there exists above two threads.

Therefore every java program is containing multiple threads.

6. Networked -



The basic aim of networking is to share the data between multiple machines which are located either in the same network (local sharing) or in different network (global sharing).

Collection of interconnected autonomous nonautonomous computers with the server

is known as Networking.

According to industry standards we have two types of networks:

1. Untrusted networks
2. Trusted

'An untrusted network is one in which their exists " collection of interconnected non autonomous computers with the server."

This network architecture preferred by those organizations whose buisiness operations are restricted to a specific place knowon as small scale organizations. These organizations prefer to have intranet applications to run on untrusted networks.

To develop the intranet applications we use J2SE (network programming concept)

Trusted network is one in which there exist "collection of interconnected autonomous computers with the server".

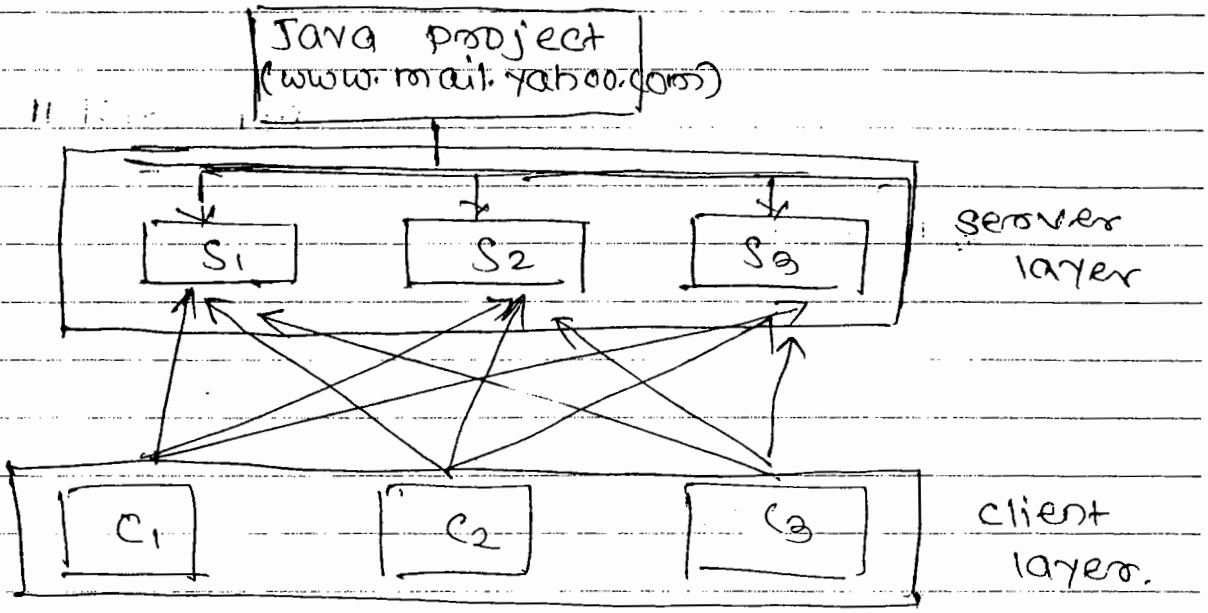
This network architecture preferred by those organizations whose buisiness operations spread across the globe. (large scale organizations). These organizations prefers to have internet applications by making use of J2EE technologies.

Finally using java one can develop

the globe.

All the real time websites are comes under distributed applications.

The digramatic representation for distributed application is shown below.



The basic advantage of distributed application is that, even though one server is down, clients are able to communicate with other servers.

This makes us to understand distributed application provides more availability of data.

The negligible limitation of distributed application is that more investmet is required for designing of high end server

Each and every distributed applicat have public URL. (e.g. www.mail.yahoo.com),

Q. What is the difference between URL & URI?

The place or address where the application is running is known as URL.

The place/address where the application is residing is known as URI.

Fri, 28/01/11

8. High performance-

Java is one of the high performance programming language because of the following reasons;

ⓐ Because of magic of bytecode.

(we know that 'bytecode is the set of optimized instructions generated by the java compiler during compilat^{n} phase and it is more faster than ordinary pointer code.)

ⓑ Because of magic of garbage collector, automatic memory management is taken care. (We know that garbage collector is the one of the system background java program runs along with our regular java program for collecting unused memory space for improving the performance of java applications).

ⓒ Java programming is free from pointers so that a java programmer can develop an application as early

Whenever we store any real constant value in a float datatype variable then float datatype organises the real constant value in such a way that after the dot (.) it takes 8 decimal places.

In order to treat any real constant value as a float data type value then the real constant value must be followed by a letter F in java otherwise it is a complete time error.

Range of any DT = (No. of bits available in lang. which are understandable by computer)

int short

2^8 2^{16}

$= 256$ $= 65536$

If any category contain amore than one datatype then to use an appropriate datatype from perticular category, to store a data in the main memory, we must know its range.

~~The formulae for calculating range of any~~

Float category datatypes

A float datatype or real data is one which is represented in the form of scale, precision.

ex. xxx yyy

here, xxx is called scale.

yyy is called precision.

In order to represent float data or real data in the main memory of the computer we use float category data types.

Float category datatypes are divided into two types & they given in the fonowing table;

SR NO.	Data type	size	range	No. of dec Places.
1	float	4	+x to -(x+1)	8
2	double	8	+y to -(y+1)	16

user defined datatype.

S is variable of student (object) holding multiple values either of same type or different type of both.

1/1/2017

Fundamental datatypes in Java -

In java programming we have 8 fundamental datatypes, which are categorised into 4 categories or groups; they are;

1. ~~In java programming~~
1. Integer category datatype.
2. float
3. character
4. Boolean

① Integer category datatype -

This category datatypes are used for storing integer data in the main memory of computer by allocating sufficient amount of memory space. In other words, this category datatype are used for representing whole numbers in the memory of computers.

Integer category datatypes are divided into four types which are given in following table:

S.NO	Datatype	Size	Range
1	byte	1	+127 to -128
2	short	2	+32767 to -32768
3	int	4	+x to -(x+1)
4	long	8	+y to -(y+1).

type.

e.g. int a[] = {10, 20, 30}; // valid

int b[] = {100, 'A', "ABC"}; // invalid.

In most of the programming languages the concept of arrays comes under derived datatypes.

Userdefined datatypes are those which are developed by programmers by making use of appropriate features of the language.

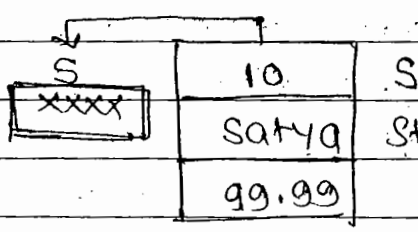
Userdefined datatypes related variables allows us to store multiple values either of same type or different type or both.

e.g. In C language, user defined data types can be developed by using struct, union, enum etc.

Similarly in C++ we use structures, unions, classes etc.

In java programming userdefined datatypes can be developed by using the features of classes and interfaces.

Student s = New Student();



In above example student is one of the class name in java which is treated as

Datatypes are used for storing the input of the program into the main memory of the computer by allocating sufficient amount of memory space in the main memory of the computer.

In other words, data types are used for representing the input of the user in the main memory of the computer.

In general every programming language is containing three categories of datatypes. They are

1. Fundamental / built-in datatypes
2. Derived datatypes
3. User/ custom/ secondary/ programmer defined datatypes.

Fundamental datatypes are those whose variables allows us to store only one value but they never allows us to store multiple values of same type.

e.g. int a ; // valid
a = 10, 20, 30; // invalid.

Derived datatypes are those whose variables allows us to store multiple values of same type. But they never allows to store multiple values of different

Interpreted -

In the older versions of java compilation phase is so faster than interpretation phase. ~~In~~.

Industry has complained to the SUN microsystem saying that compilation phase is very faster and interpretation phase is very slow.

So solve this issue, sun microsystem has developed a program called JIT (Just in time compiler). And added as a part of JVM to speed up the interpretation phase.

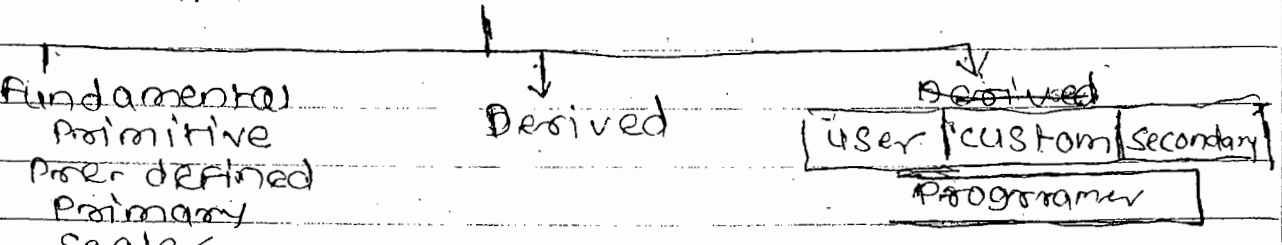
In the current versions of JAVA, interpretation phase is so faster than compilation phase.

Hence JAVA is one of the highly interpreted programming language.

Q. Define JIT.

→ JIT is the set of programs developed by sun microsystem and added as a part of JVM, to speed up the interpretation phase by reading the entire section of the bytecode and converting native understanding form of OS.

Data Types -



as possible compaired to any other the language programmers. So that performance of the java programmer is improved.

9. Robust (strong) -

When we write any program in any programming language, we come across various types of errors.

Broadly errors are classified into two types; they are

(a) Compile time errors
(b) Runtime errors

Compile time errors are those which are resulted during the program compilation when the programmer is not following Syntax of the language.

Runtime errors are those ~~when~~ which are resulted during program execution when the normal user enters invalid input at runtime.

The languages like C, C++, pascal, COBOL are not able to address runtime errors effectively hence these languages related applications are not robust.

The language like java is able to address the runtime errors effectively by using a concept of exception handling.

Hence the java language related applications are robust.

Def. of exception- Runtime error of java program is known as exception.

Therefore java is one of the robust language because of the inbuilt facility called exception handling.

10) Dynamic-

In most of the programming language, memory allocation techniques are divided into most two types they are;

ⓐ static memory allocath-

ⓑ Dynamic memory allocath.

A static memory allocation is one in which the memory will be allocated at compile time.

The limitations of static memory allocations are,

① waste of memory space (the number of values we enter at runtime are less than the size of the array)

② loss of data (the number of values we enter at runtime are more than the size of the array.)

③ overlapping of existing data (if any array contains overlapped values and if we try to insert a

new value within the existing memory location of the array then we need to move the elements of a array either to downwards or upwards which is quite impossible in arrays. So that the new value will be replaced ~~by~~ the existing value.

Because of the above limitations, Java programming does not follow static memory allocation. But it always follows dynamic memory allocation.

The process of allocating the memory space to the input of the program at a runtime is known as dynamic memory allocation.

Dynamic memory allocation eliminates all the drawbacks of Static memory allocat^n

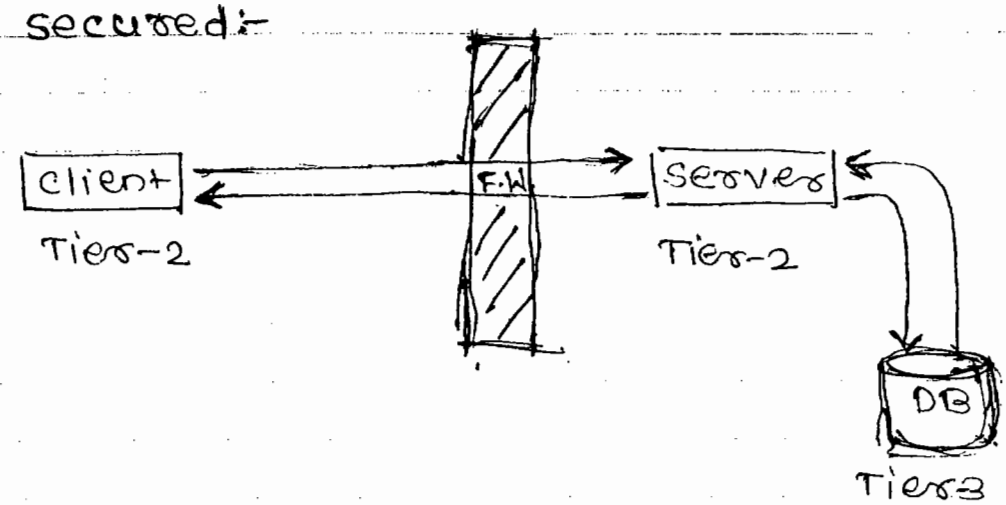
~~In~~

In Java programming to allocate memory space dynamically we use an operator called new. 'new' operator is known as dynamic memory allocation operator.

Q. Define an array.

→ Defⁿ - An array is a collective name given to a ~~collect~~ group of consequtive memory locations which are all reffered by similar / homogeneous elements.

secured:-



Security is one of the principle in information technology to validate wheather is the user is authorised or unauthorised.

If we develop any project other than java language then as a programmer we need to write our own special programs for dealing with security. Which indicates other than java language all the proggramming languages by default unsecured.

In the case of java projects, as a java programmer we need not to write any special programs on our own to deal with the security, which indicates the API of Javaa is containing readily available security programs. (which are implimented from network security subject.)

Because of the readily available security programs Industry experts felt that java is one of the high secured language.

Boolean category-datatype -

Boolean category datatype is used for representing or storing logical values i.e. true or false.

In java programming, to represent boolean values or logical values, we use a datatype called boolean b.

Boolean datatype takes zero byte of main memory space because boolean datatype of java implimented by SUN microsystem with a concept of flip-flops.

A flip flop is a general purpose register which stores one bit of information (one & true & zero false).

Note- In C, C++ (Turbo) boolean datatype is not available for representing true & false values but a true value can be treated as non-zero value. And false values can be represented by zero.

Note- All the keywords in java must starts with small letters.

Q. Define a keyword or reserved word.

A keyword is one which will have a special meaning to the compiler of a language.

e.g. do, while, for, int etc.

Variables in java -

If we enter any input to the program, then the input must be stored in the main memory of the computer in terms of variables.

Without variable's concept one can not represent or store the input of the user in the main memory of the computer.

Variables are also known as building blocks of programming languages.

Def^n of a variable - Variable is an identifier whose value is changing during execution of the program.

Rules for writing variables -

When we write some java application we may use collection of variables. When we are using variables in java, the java programmer is highly recommanded to use the following rules.

The first letter of the variable must be an alphabet.

The length of the variable in java can be allowed upto 32 characters.

No special symbols are allowed except underscore.

No keywords are reserved words to be used as variables.

Hence every input of the user must be stored in terms of variables.

Variable declaration -

The process of allocating sufficient amount of memory space along with a variable name in the main memory of computer is known as variable declaration.

Before using any variable in java that variable must be declared.

Syntax -

datatype $v_1, v_2 \ldots v_n$ - here datatype represents either fundamental or derived or user defined.

$v_1, v_2, \ldots, v_n$ represents valid variable names of java.

e.g. int a, b, c.

float f1, f2, f3;

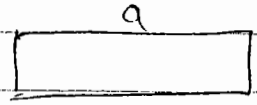
char c1, c2, c3

boolean b, b2.

Let us consider the following statement;

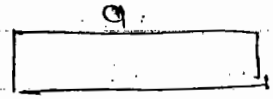
When the above statement is executed then the system will perform the following actions:

a) We get sufficient memory space depends upon data type.

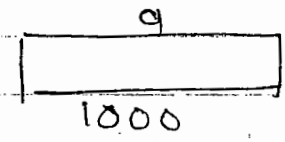


b) Whatever memory space is created in above state, the memory

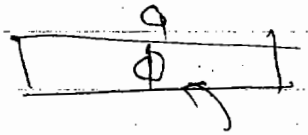
space is qualified by variable name a.



© Each & every variable in java must have a distinct or unique address. Address is for system to process the value and variable is for the programmer to access the value.



ⓓ Memory is one of the essential component, the system will place default values depends on the datatypes.



Note- 1) The default value of the integer category variables is zero.
The default value of the float category variables is 0.0.
The default value of character category variables is nothing.
The default value of boolean category variables is false.

Initialization of variables-

The process of placing programmer defined value within the variable without placing default value is known as initialization of a variable.

Whenever we decided to place predecided values in the variables then we must choose the concept of initialization.

Syntax

datatype $v_1 = val_1, v_2 = val_2 \ldots v_n = val_n$;

$val_1, val_2, \ldots, val_n$ represents list of predecided values placing in the variables $v_1, v_2, \ldots, v_n$ respectively.

ex. float PI=3.1417f;
int a=10, b=20; etc;

constants in java-

final

variable level

final datatype $v_1 = val_1, \ldots v_n = val_n$

final float PI=3.1417f;
PI= PI+1;
int a=100;

method level

ReturnType methodName(
listof formal para. if any)
{
Block of stmt(s);
}

If we don't want to change the value of the variable then those variable values must be made as constants.

Def? of constant- A constant is an identifier who's values can not be changed during execution of the program.

In java programming to make anything as constant we use a keyword final.

final is a keyword which is placing an important role in java at three places, they are at ① variable level
② At method level
③ At class level.

① final at variable level- If we don't want to change the value of the variable then the variable value must be made as final. In other words if want to make any variable value as a constant then the variable value must be made as final.

Syntax-

final datatype V_1 = val1, V_2 = val2, ...
V_n = valn.

Ex.

```
final float PI = 3.1417f ;
PI = PI+1 ; // Invalid.
```

Once, the variable value is final, whoes value can not be changed.

Therefore final variable values can not be changed or modified.

Final in method level:-

We know that each and every operations of our programming languages must be performed with respect to functions/methods.

If we don't want to change the definition of the method then the method definition must be made as constant by using final keyword.

In another words whenever we develop another method which is common for many number of java programmers then such methods are highly recommanded to make them as final.

Syntax -

```
final returntype methodname
        (list of final params if any)
{
Block of statements;
}
```

```
final float simpleInterest(float P,
                    float T, float R)
{
  float SI = P*I*R;
  return (SI);
}
```

Once the method is final, which is not possible to redefine/overriden.

Therefore, final methods cannot be overridden.

Q. Define method overriding and method overloading?

Ans- Method overriding- The process of redifining the original method for performing multiple operations is known as method overriding.

OR.

Method overriding = method heading is same + method body is different.

ex.

```
void operation (int x, int y)
{
  int z = x+y;
}
```

original method

```
void operation (int x, int y)
{
  int z = x-y;
}
```

re-defined / overridden method

Hence operation method is known as overridden method.

Method overloading-

Defⁿ- A method is said to be overloading method if and only if method name is same

but it's signature is different.

Signature represents the following

(a) No. of parameters
(b) Type of parameters
(c) Order of parameters

} At least one thing must be differentiated.

Ex.-

```
Sum(10,20,30) ; ——①
Sum(100,200) ; ——②
Sum(10.75f,10.25F); ——③
Sum(100,10.75F) ; ——⑤
Sum(10.75f,100) ; ——⑥
```

The above sum function is known as overloaded function.

3/2/2011.

Final ~~values~~ at class level-

If we don't want to give the features of base class to the derived class, then the base class definition must be made as final.

Syntax for final class.

```
final class <class name>
{
==
}
```

ex. final class personal
{
}
class others extends personal // Invalid
{
}

Once the class is final whose properties will not be inherited from base class to derived class.

Final class is never participated in inheritance process

Note- Final variable values cannot be modified.

Final methods cannot be overriden.

Final class cannot be inheritable/reusable/extendable.

OOPS Features or concepts-

IT - Prog. lang
- Procedure oriented prog. lang:- C, Pascal, COBOL, ADA, ...
- Object oriented prog. lang:-
  - Small Talk ✗
  - Lisp —
  - Object Pascal
  - Object cobol
  - *C++ → Partial object oriented because:
    1) Every C++ program can be written like C prog.
    2) 50% of OOPs principles not satisfied
  - JAVA
  - .Net etc

3) Friend function-

In IT we have two model of languages they are;

1) Procedure oriented programming langs
2) Object oriented programming langs.

When we store the data by using procedure oriented programming languages that data can be manupulated by any unauthorised user.

According to industry experts, procedure oriented programming languages provides 0% security so that it is not recommanded for developing organization level applicatns. like banking, supermarket, transportation etc.

exn: C, Pascal, COBOL, ADA etc.

To eliminate the limitations of procedure oriented programming languages we have another model of language called object oriented programming language.

In order to say any programming language is object oriented, it has to satisfy all the complete principles/concepts of object oriented system/structure invented by OMG group. (Object Management group).

If we develop any application with object oriented programming languages then the data of that application can not be manipulated by any unauthorised users, so the basic aim of object oriented

programming is that to provide effective security to the data but all the programming languages which are comes under object oriented may not be providing 100% security.

examples-

Smalltalk, lisp, object pascal, object COBOL, C++, java, .Net etc.

Even though C++ listed under object oriented programming, according to industry experts opinion C++ is one of the partial object oriented programming language because of the following reasons;

(a) Every C++ program can be exactly written like a C program.

(b) A C++ program may or may not satisfy the complete principles of OOPS.

(c) C++ is having a concept called Friend Functions which is toying to access private data which is not recommanded.

The languages/technology like java or .Net are treated as object oriented because they satisfies completely all the OOPS principles and no Friend Functio concept is allowed.

Object oriented programming contains 8 principles they are:

1) classes
2) Objects
3) Data Abstraction
4) Data Encapsulation
5) Inheritance
6) Polymorphism
7) Dynamic binding
8) Message passing.

Fri, 4/2/2011

Class-

Class concept is always used for developing user/ programmer defined defined datatypes.

To developconcept of class we use a keyword class

Without class concept there is no java program.

Definition- The process of binding the data members and associated methods in a single unit. This single unit is known as class.

The data members of the class are also known as attributes or properties.

The methods of the class are also known as behaviours or accessories.

In object oriented programming we have 2 categories of methods.

They are,

a) Member method

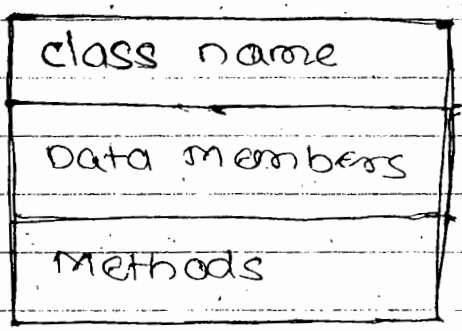
b) Non member method.

A member method is one which is available within the scope of the class. and it can access the data of the class.

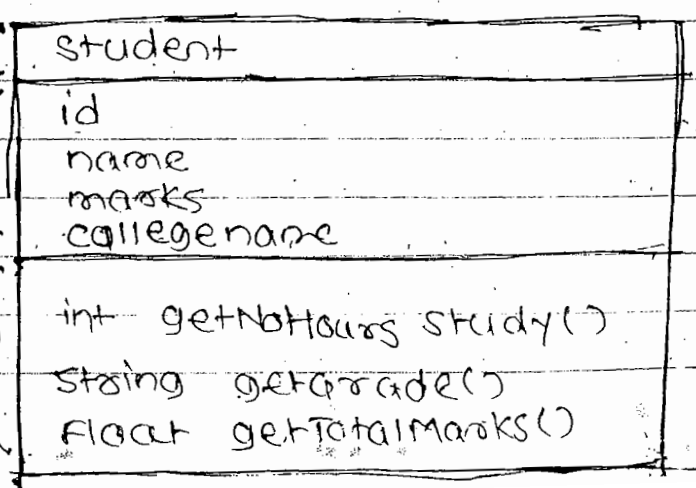
A non member method is one which is not available within the scope of the class and it can not access the data members of the class.

In java programming, member method concept is allowed but there is no concept called non-member methods.

In real world applications the discription of the class can be expressed in the form of class diagrams & whose structure is given below.



Ex. Define or express a class student with suitable data members & ~~student~~ methods.



The above diagram is common for all the object oriented language programmers & whose syntax is differing from one object oriented programming language to other language.

syntax for defing a class-

```
class <classname>
{
 variables declaration;
 Methods definition;
}
```

Explanation- In the above syntax;

1) Class is a keyword used for developing user/programmer defined datatype.

2) CLS names represents a java valid variable name treated as name of the class. All the class names in java are treated as user/programmer defined data types.

*3) Whenever we define a class, there is no memory space for data members of the class. Memory space will be created for the data members of class when we create an object.

4) In java programming we use all the class names for creating objects with respect to the class. that is

with respect to a single class definition, one can create multiple objects.

5) Variable declaration represents data members of the class.

6) Methods definition represents the type of methods we use to perform some specific operations by making use of data members of the class.

7) All the methods of java must be defined inside the class only i.e. java programming never allows to define the methods outside the class, and the metho of java are known as member methods

8) The definition of a class may or may not be terminated by semicolon.

9) Each and every class in java have logical existance(no memory space is occupied) where as every object in java will have physical existance(memory allocated).

ex. Define a class student -

```
class student
  {
    int id;
    string name;
    float m1,m2,m3;
    string colname;

  int getNoHourstudy()
  {
    return(n);
  }
```

```
String getGrade()
{
return "Distinction";
}

float getTotalMarks()
{
return (m1+m2+m3);
}
} // Student.
```

Sat, 5/2/11

Q. Define a class for computing sum of two numbers.

Sum
a b c
void input() void add() void drop()

```
class sum
{
  int a,b,c;
void input()
{ a=10;
  b=20;
}
void add()
{
}c = a+b;

void drop()
{
System.out.println(
  "val of a="+a);
System.out.println(
  "val of b="+b);
```

Whenever we define a class, there is no memory space for data members of the class. Memory space will be created for the data members of the class when we create an object.

In other words, if we want to enter the values for the data members of the class, then first we have to create the memory space by creating an object with respect to the class.

Without object creation data processing of a java application is not possible.

Definitions of object -

1. Instance of a class is known as an object (Instance is nothing but allocating sufficient amount of memory space for the data members of the class.,
2. Each and every class variable is known as an object.
3. Grouped item is known as an object. (Grouped item is a variable or identifier which allows us to store multiple values either of same type or different type or both)
4. value form of class is object.
5. Blue print of a class is known as an object.
6. Real world entities are known as object.
7. Logical runtime entity is known as an object.

Object creation -

When we create an object, we get the sufficient memory space for the data members of class.

Creating an object in java is nothing but following dynamic memory allocation by making use of new operator.

The new operator is known as dynamic memory allocation operator.

The operator new internally performs two types of operations, they are;

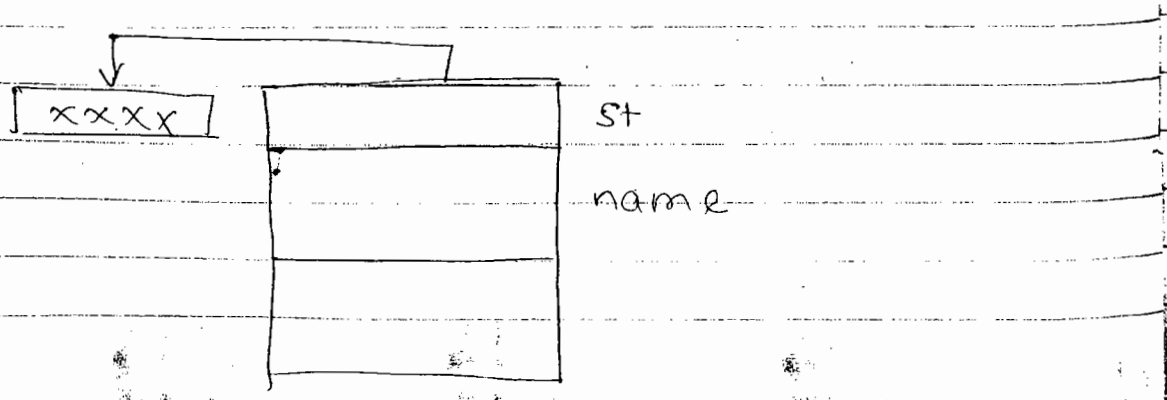
a) Allocating sufficient amount of memory space for the data members of the class.

It takes an address of the class which is loaded in the main memory and places into l.h.s. variable i.e. object variable.

In java programming, to create an object, we have two syntaxes,

i. <class name> object name = new <class name()>

Student So = new student();

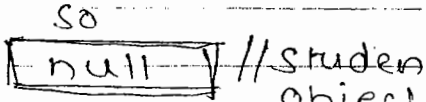


ii. <classname> = objectname; ——①
objectname = new <classname()>; ——②

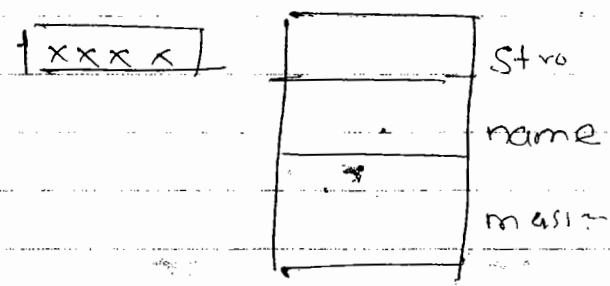
Statement ① represents object declarat when the object is declared, whose default value is null since there is no memory space for the data members of the class.

Statement ② represents object referencing. when the object is referenced, memory space is created for the data members of the class and the object defaut value is not null.

Ex. Student so;



So = new Student(); // Student object referencing.



Q. Define class loader subsystem.

→ A class loader subsystem is a program developed by sun microsystem added as a part of JVM & it is used for loading .class files from secondary memory (hard disk) to main memory.

Q. What are the differences between class and object?

→

class	object.
1. The process of binding data members & methods (associated) in a single unit is known as class.	1. The class variable is known as object.
2. When we define a class there is no memory space for the data members of the class.	2. When we create an object, we get the memory space for data members for the class.
3. All the classes in java will have logical existance.	3. All the objects in java will have physical existance.
4. The definition of a particular class will exists only once.,	4. With respect to one class, once can create multiple objects.
5. When we execute the java program, class will be loaded in the main memory, ii) After loading the class in the main memory, object creation can be possible.	5. After loading the class in the main

Types of data members -

Each & every java program must be written with respect to the class concept.

We know that a class is containing data members & methods.

The data members of the class are classified into two types, they are;

1) Instance/non-static data members -
2) static data members -

Instance / non static data members	static data members
1) Instance data members are those whose memory space is creating each and every time whenever an object is created.	1) Static data members are those whose memory space is creating only once when the class is loaded in the main memory irrespective of number of objects are created.
2) If our class is containing collection of data members and we are placing specific values then those data members are recommanded to take as instance data members.	2) If our class is containing collection of data members which are containing common values or fixed values then the data members of the class are recommaned to take as static data members.
3) programatically, to make any data member as	programatically, to make any data member as static

instance data member	data member
whose declaration should not be preceded by a keyword 'static.'	whose declaration must be preceded by a keyword static.
4) Syntax- datatype $v_1, v_2, \ldots, v_n$;	4) Syntax- static datatype $v_1, v_2, \ldots, v_n$;
*5) Each & every instance data member must be accessed with respect to object name. Objname. IDM (Instance data member Name)	*5) Each & every static data member must be accessed with respect to class name. classname. SDM (Static Data Member name)
6) The values of instance data members are not sharable.	6) The values of static data members are always sharable.
7) Instance data members are also known as object level data members because they depends on object and independent from classname.	7) static data members are also known as class level data members because they depends on class & independent from object name.
8) example- int stno, pin.	8) ex static string college_name, branch_name;

Types of methods in class -

A class is containing collection of data members and methods.

The methods of the class can be divided into two types,

1) Instance / non-static methods
2) Static methods.

Instance Methods	Static Methods
1) Instance methods are those which are recommended to perform repeated operations such as reading records from the file, reading records from the database etc.	1) Static methods are those which are recommanded to perform one time operations such a opening a file, obtaining the database connection etc.
2) Programmatically instance method's definition should not be preceded by a keyword static.	2) Programatically, static method's definition must be preceded by a keyword static.
3) Syntax — returntype methodname (list of formal params if any) { Block of statements; }	static returntype methodname (list of formal params if any) { Block of statements; }
4) Each & every instance method must be accessed with respect to objectname. objname.instance method();	4) All the static methods in java must be accessed with respect to class name. classname.static methodname;

5) The result of instance method is not sharable	5) The result of static method are always sharable.
6) Instance methods are known as object level methods. because they depends on object & independent from classname.	6) Static methods are also known as class level methods because they depends on classname and independent from object.

example-
void gettotal()
{
tot = m1+m2+m3;
}
}
	Public static void main (String arg[])
	{
	S1.gettotal();
	S2.gettotal();
	}

Note - A class can contain instance data members & static data members, instance methods & static methods.

Instance data members & instance methods must be accessed with respect to object name and static data members and static methods must be accessed with respect to class name.

System.out.println()
System.out.print()

The above statements are used for displaying the result of the java program on the console. either by line by line or in the same line.

The technical meaning of the above statements is given below.

println and print methods are two predefined overloaded instance methods, presenting a predefined class called printstream

To access these methods we need an object of printstream class but it is not available freely outside.

An object of printstream class is called out created in another predefined class called System as a static data member. Hence println & print method can be accessed as follows.

```
System.out.println();
System.out.print();
```

The following sample code gives the development aspect of the above statements

```
class System
{
Static PrintStream out=
                    new PrintStream()
  ---
  ---
  ---
}
```

```
class PrintStream
{
public void println(int)
  {
  ==
  }
public void print(--)
  {
  ==
  }
}
```

Examples -

① "Java world"

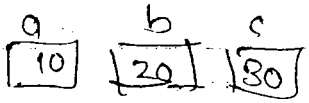
```
System.out.println("Java world");
```

②

```
int a = 10
sop(a); // 10
sop("val of a = "+a); // val of a = 10
sop(a+" is the value of a"); // 10 is value of a
```

③

```
int a = 10, b = 20;
int c;
c = a+b;
```




```
Sop("sum ="+c); // sum = 30
Sop(c+" is no sum"); // 30 is the sum
Sop("sum of "+a+" & "+b+" ="+c);
        // sum of a & b = 30.
```

Structure of the java program-

When we write any program in any programming language, it is mandatory for the programmer to follow the standard structure.

As a java programmer to write any java program, we follow the following standard structure;

```
Package details;
Class <class name>
{
    Data members

    User defined Methods

Public Static void main( String arg[])
{
  Block of Stmt(s)
}
}
```

In the above structure;

① package details represents collection of classes, interfaces & subpackages etc. If we use any predefined classes and interfaces as a part of our java program then it is mandatory to java programmer to specify in which package those classes and interfaces present.

② Each and every java program must

starts with concept of class i.e. without class name there is no java program.

③ CLS name represents a java valid variable name treated as class name, we know that whenever we define a class, there is no memory space for a data members & methods of the class. All the class names in java are defined as user/programmer defined datatypes and they can be used for creating objects.

④ Data members represents either instance or static & they are selected based on the class.

⑤ user defined methods represents either instance or static developed by the java programmer to perform specific operation according to the user requirement & these type of methods are known as beisiness logic methods.

⑥ Each and every java program execution starts from main method hence this method is known as program driver.

⑦ Since main method is not return any value so that its return type must be void.

⑧ since main method is executing only once throughout the life of

the entire java program, the nature of the main method must be static.

Since main method can be accessed universally whose access specifier must be public (Universal Access).

Each and every main method must take array of objects of String class

Block of statements represents represents set of executable statements which inturns calls user defined methods.

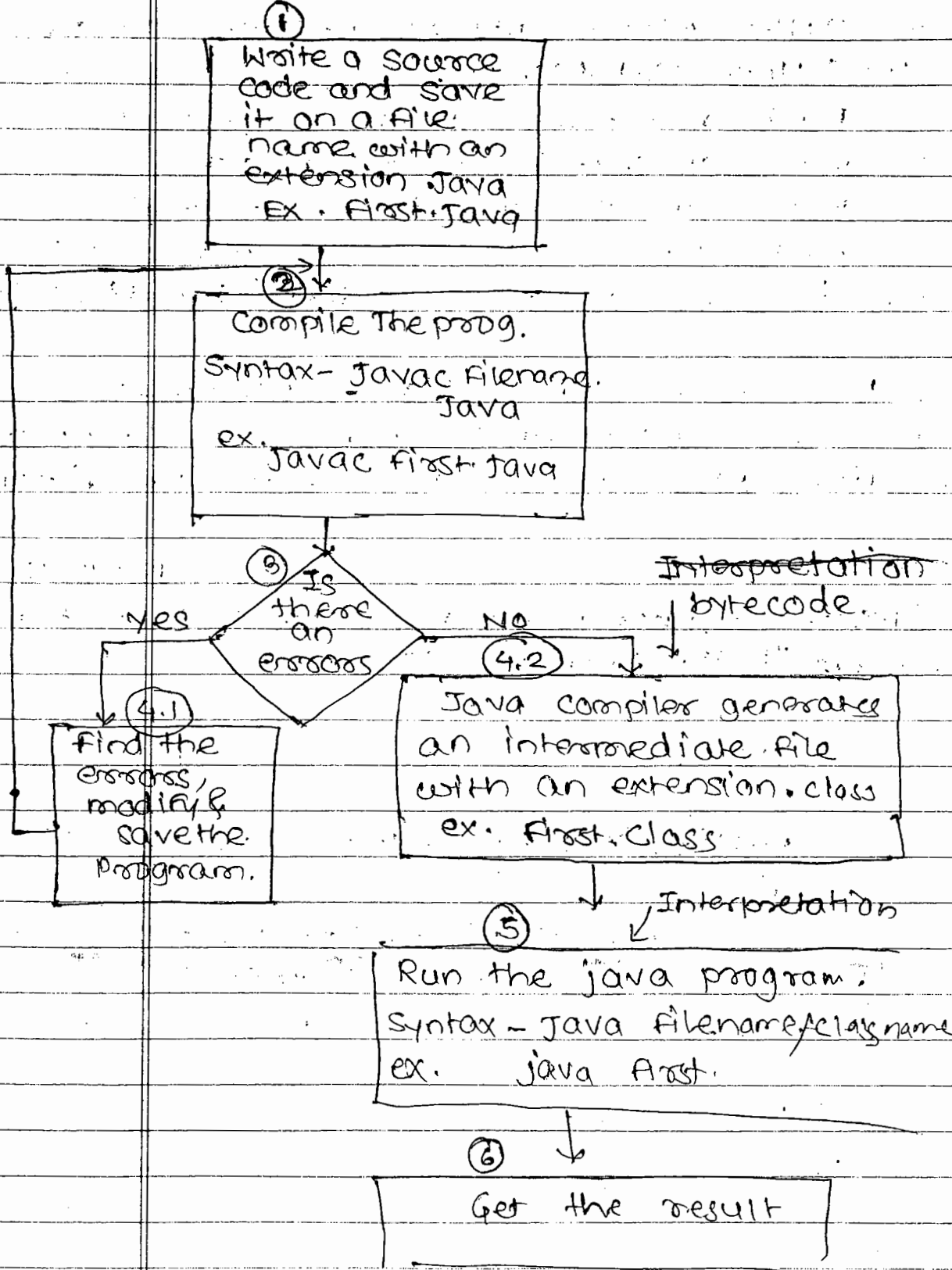
File naming convention - whichever class contains main method, that class must be given as filename with extension .java.

10/2/2011

Write a java program which will print a message "hello java world".

```
class First
{
public static void Main (String k[])
{
  System.out.println(" HELLO java world");
  S.o ——— " ——— (" This is my
                  First program");
}
}
```

Digramatic representation for compiling & running the java program.



Note- Requirement for java application or project development.

Java Software requirements
www.sun.com
Version- 1.5, 1.6.

OS requirements - any.

IDE - { Edit PLUS - not recommanded in industry.
Netbeans
MyEclipse
WSAD (Websphere Appl? develop? mens?

Q. Define IDE.

An IDE is the software developed by third party vendors for writing or developing a java program.

IDE. (Integrated DevelopmentEnvironment)

Hungerian Notation-

It is one of the naming convension followed by Sunmicrosystem for developing predefined classes/ interfaces, methods and data members.

Hungerian Rule I for classes and interfaces-

If a class name or interface name contains either one word or more than one word then all the word's first letters must be capital.

examples-

Normal Notation	Hungerian Notation
system	System
actionevent	ActionEvent
Numberformat-exception	NumberFormatExcept
Array index out of bound exception	ArrayIndexOutOf BoundException.

Hungerian Rule 2 for methods-

If a method name contains either one word or more than one word then first word, first letters must be small and rest of the subsequent word's first letters must be capital.

Normal Notation	Hungerian Notation,
println	println()
actionperformed	actionPerformed()
ItemStatechanged	itemStateChanged()

Hungerian rule 3 for data members-

If we use any data members of predefined classes/ interfaces then all those data members must be used in capital letters.

If a data member contains more than one word then that data member must be seperated by underscore (_).

Normal Notation	Hungerian Notation
pi	PI
maxvalue	MAX_VALUE
scrollbarverticalonly	SCROLLBAR_VERTICAL_ONLY

Hungerian notation is mandetory for the java programmer to be followed at the time of using predefined classes/ Interfaces / methods & data members.

Hungerian notation is optional to the java programmer to be followed for user defined classes / interfaces, methods & data members.

In real world applications, it is highly recommanded to follow hungerian notation even for user defined classes/ interfaces, methods & data methods.

11/2/2011

Write a java program which ~~list~~ illustrate the concept of instance methods & static methods

```
// Third.java

class Third
{
  void f3()
  {
  System.out.println(" f3-instance..");
  }

  void f1()
```

```
{
  f3(); // rule-2
  Third.f2(); //rule-4
System.out.println("f1-instance..");
}

static void f2()
{
System.out.println("f2-static..");
}

Public static void main(String[] args)
{
System.out.println("i am from main-beg");
  Third t1 = new Third();
  t1.f1(); //rule-3
   f2(); // or Third.f2(); ... rule-1
   System.out.println("i am from main
                        -end");
}
}
```

In the above program

Rule-1 : One static method can call another static method directly provided both static methods belongs to same class.

Rule-2: One instance method can call another instance method directly provided

both the methods must belongs to same class.

Rule-3: One static method can call another instance method with respect to an object provided wheather they belongs to same class or different class.

Rule-4: One instance method can call another static method with respect to class name provided wheather they belongs to same class or different class.

Note :- As a part of the class what are all the methods defined by the programmer are known as business logic methods and the block of statements available in the business logic methods is known business logic.

Q. Define a business logic method.

→ Business logic methods are those which are defined by the java programmer by writing the business logic ~~by pen~~ for providing solution to a client requirement.

Write a java program which will compute sum of two numbers.

```
// SumDemo.java
Class Sum
{
  int a,b,c;
  Void assign()
    { a=10;
      b=20;
```

```
    }
    void add()
    {
    c = a+b;
    }
    void display()
    {
  System.out.println("Val of a="+a);
  Syst ——— " ——— ("Val of b="+b);
    ——— " ——— ("Sum="+c);
    }
  }// sum- Business logic class

class SumDemo
 {
 public static void main(String[] args)
 {
System.out.println(" i am from main-
                        beg");
    Sum so = new Sum();
      so.assign();
      so.add();
      so.disp();
 System.out.println(" i am from
                    main--end");
 }
}// sumDemo-- execution Logic class
```

In the above program, the main method contains the block of statements which are

calling business logic methods hence those block of statements are known as execution logic.

Whichever method contains execution logic, that method is known as execution logic method. In general, main method is always known as execution logic method.

Whichever class contains execution logic method, that class is known as execution logic class.

In the java program execution, execution starts from execution logic class (main method) and execution ends with execution logic class.

In middle of program execution and program termination, lot of business logic methods are called or executed.

~~For~~ One java program can contain multiple business logic classes and recommanded to have single execution logic class.

The limitations of the above program are;

In order to get different results, we do the following steps:

1. Go to the java program
2. change the values
3. Save the program
4. Recompile program.

As and when to get different results, we must perform the above cycle of

operations which is not a
recommanded process.

It is highly recommanded to the java programmer to accept the values dynamically but not within the program by passing fixed values such programming is known as static programming.

14/2/2011

Length -

Length is one of the implicit variable created automatically by the JVM and supplied to every java program for two purposes.

(1) To find the number of elemnts or size of the array.

(2) To treat the fundamental arrays as objects because java is one of the pure object oriented programming language.

Syntax -

Arrayname.length.

Example -

(1) int a{10, 20, 30}
SOP ("No of elements =" +a.length);

(2) String S[] = {"abc", "PQR", "JAVA"};

SOP ("No of Names=" + S.length); //3

for(int i=0; i<S.length; i++)

{ sop(s[i]);
}

Accessing the data dynamically :-

Accessing the data dynamically is nothing but entering the values to the java program at runtime.

In java programming to access the values dynamically ; we have three approaches they are

1) By using command line arguments
2) By using keyboard
3) By using properties file. (Resource bundle file) -

1) By using command line arguments -

(Q. Why java main method takes array of objects of String class as a parameter?)

→ Every main method types array of objects of string class because to accept command line arguments.

The number of values which we are passing to the java program from the command prompt are known as command line arguments.

ex.

e:\1am j2se> java sumdemo 60 70

command Prompt — Java Program — command line arguments

1) When we enter any command line arguments to the java program, they are by default treated as String data.;

2) All the command line arguments are by default passing to main method.

3) All the command line arguments must be held in main method by taking a String parameter.

4) Since command line arguments are unlimited/ dynamic, main method is holding all the command line arguments in a array variable in a string type.

5) While we are using array variable of string type in the main method, one can not specify the size of the array.

6) Since java is pure object oriented programming, string data which is available in main method must be represented by using a predefined class called String.

7) Therefore, every main method must take array of objects of String class for holding command line arguments.

Write a java program to print or to accept command line arguments.

```
class DataPoint
{
 Public static void main(String arg[])
 {
 System.out.println("no. of values="
                                +arg.length);
 for(i=0; i<k.length ; i++)
  {
  System.out.println(k[i]);
  }
 }
}
```

Note- When we execute the java program jvm always look for that main method which is taking array of objects of string class but not for other main methods which are taking different types.

Converting String data into fundamental or numerical data- 15/2/2011

When we pass any command line arguments to the java program, by default the command line arguments are treated as string data. We know that on string data one can not perform numerical manupulations hence it is highly desirable to convert string data into numerical or fundamental data.

→ The following table gives ^ datatype corresponding wrapper class and conversion method from string data to fundamental data.

Datatype	Wrapper class name	conversion method from string to fundamental datatypes.
byte	Byte	Public static byte Parsebyte(String)
short	Short	—"— Short parseShort(String)
int	Integer	—"— Int parseInt(String)
long	Long	—"— Long parseLong(String)
float	Float	—"— Float parseFloat(String)
double	Double	—"— Double parseDouble(String)
char	Character	—"— Char parseChar(String)
boolean	Boolean	—"— Boolean parseBoolean(String)

In general each & every wrapper class is containg the following generalized predefined method for converting string data into fundamental data.

Wrapper class

↓

Public static xxx parse xxx (String)

Here xxx represents any fundamental datatype.

example-

```
String S[]={"10","20"};
Sop(S[0]+S[1]); //1020
```

```
int x = Integer parseInt(s[0]);
int y = Integer. parseInt(s[1]);
int z = x+y;
SOP(" sum =" +z); //30
SOP(" Sum =" +x+y);// 1020
```

Write a java program which will perform interchanging of two values.

```
class swapdemo
{
P S V main(string
int a,b;
void set(int x, int y)
{
a=x;
b=y;
}
void interchange()
{
int t=a;
a=b;
b=t;
void disp()
{ System.out.println("val of a="+a);
System.out.println("val of b="+b);
}
} // swap ~ BLC.
```

```
class swapdemo
{
PSV main(String k[])
{
int x = Integer.parseInt(k[0]);
int y = Integer.parseInt(k[1]);
Swap so = new Swap();
so.set(x,y);
System.out.println("original values");
so.interchange();
so.disp();
}//main
}//swapdemo ---- ELC
```

① Javac swapdemo.java

program flow in j2se applications-

① compile the program.
② Run the java program by passing command line arguments.
③ Whatever the command line arguments we are passing, those values will be sent to execution logic class.
④ Execution logic class sends those values to execution logic method. (main method) and the data is organized in the form of string.
⑤ convert the string data into

numerical or fundamental data.

⑥ Create the objects of business logic classes.

⑦ Invoke or call business logic methods of business logic classes.

⑧ Once the business logic methods are invoked, control goes to business logic class, executes business logic methods & gives result back to execution logic method (main).

⑨ Program execution terminated within the execution logic method.

Write a java program which will generate or display multiplication table for given number.

```
class mul
{
 int n;
 void assign(int x)
  {
 n=x;
  }
void table()
  {
For( int i=1; i<=10; i++)
  {
System.out.println(n+" * "+i+" = "+n*i);
  }
}
}//mul
```

```
class mulDemo
{
public static void main(String[] args)
{
if(args.length!=1)
System.out.println("plz enter
                    one value..");
}
else
{
  int x = Integer.parseInt(args[0]);
  if(x<=0)
  {
System.out.println(x+" is invalid..");
  }
  else
  {
    mul m = new m();
    m.assign(x);
    m.table();
  }
}
} // mulDemo.
```

16/2/11

Write a java program to calculate a factorial of a given number.

```
class fact
{
 int n;
 void set(int x)
```

```
{
  n=x;
}
void disp()
{
int res= factcal();
System.out.println("factorial="+res);
}
int factcal()
{
   int f=1;
    while(n>0)
    {
     f=f*n;
     n=n-1;
    } return f;
  }// factcal-BLC

class FactDemo
{
  public static void main(String k[])
  {
    if(k.length!=1)
     {
      System.out.println("plz enter one value");
     }
     else
      {
    int x= Integer.parseInt(k[0])
     if(x<0)
      {
  System.out.println(x+" is invalid.");
      }
```

```
else
  {
  Fact fo = new Fact();
  fo.set(x);
  fo.~~Fact~~ disp();
 } //else
}//else
}//main
}// Factdemo.
```

Write a java program to decide whether the given number is prime or not.

```
class prime
{
int n;
void set (int x)
{
 n=x;
}
String decide()
 {
 for(i=2; i<n; i++)
   {
     int r=n%i;
     if(r==0)
      {
        break;
      }
 }// for
```

```
    if (i==0)
        return "PRIME";
    else
        return "NOT PRIME";
    }// decide
}// Prime ... BLC

class primedemo
{
  Public static void main (String [] args)
  {
   if (args.length!=1)
   {
System.out.println(" enter one value ");
   }
   else
   {
    int x= integer.parseInt (arg[0]);
    if(x<=1)
    {
    System.out.println (x+" is invalid..");
    }
    else
    {
    Prime  p0 = new Prime();
     p0.set(x);
     String res= p0.decide();
     SOP ("+res);
    }
```

In java programming a method is not only returning fundamental values as a

return type but also returns class names as a return type. (user/ programmer defined datatypes).

Q. Define a wrapper class? Explain it's purpose.

Ans. Wrapper classes are the predefined classes which are existing for each and every predefined datatype.

The basic purpose of wrapper classes is that converting the data from string into fundamental, (numerical).

Write a java program which will convert an ordinary number into roman numbers.

Note- control structures are those which are executing the block of statements either only once (conditional statements) or repeatedly until the condition evaluation becomes false.

1000-M
900-CM
400-CD
500-D
100-C
50-L
40-XL

2838
1000
1838

1000-M — W
900-CM — if
500-D — if
400-CD — if
100-C — W
90-XC — if
50-L — if
40-XL — if
10-X-W
9-IX — if
5-V — if
4-IV — if

```
class Roman
{ int n;
  void set(int x)
  {
    n=x;
  }

void convert()
  {
   if(n<=0)
   {sop(" No Roman eq ");
   }
   else
   {
   while(n>=1000)
    {
    sop("M");
    n=n-1000;
    }
    if(n>=900)
    {
    s.o.pfint("cm");
    n=n-900;
    }
   if(n>=500)
    {
    soprint("D");
    n=n-500;
    }
   if(n>=400)
    {
    soprint("CD");
    n=n-400;
   }
   while(n>=100)
    {
    soprint("C");
   } n=n-100};
```

```
if(n >= 90)
{
S.O.print("xc");
n=n-90;
}
if(n>=50)
{
S.O.print(L");
n=n-50;
}
if(n>=40)
{
S.O.print("XL");
n=n-40;
}
while(n>=10)
{
S.O.print("x");
n=n-10;
}
if(n>=9)
{
Soprint("Ix");
n=n-9;
}
if(n>=5)
{
Soprint("v");
n=n-5;
}
if(n>=4)
{
Soprint("IV");
n=n-4;
}
while(n>=1)
{
```

```
        SOPrint("I");
        n=n-1;
      }
    }//else
  }// convert
}// Roman

class RomanDemo
{
 PSVmain(String k[])
 {
  If(k.length!=1)
  {
   SOP("PLZ enter one value");
  }
  else
  {
   int x= Integer.parseInt(k[0])
    Roman r= new Roman();
     r.set(x);
     r.convert();
  }//else
 }// main
}Roman demo.
```

Write a java program which will convert a Roman number into an ordinary number

Ex —

MMX — 1000+1000+10 = 2010
valid

MCA — Invalid becoz A is not

valid roman characters.

2) Write a java program to decide wheather the given number is perfect or not.

(perfect no. = sum of the factors of the original number)

6 :: 1+2+3 = 6 is perfect

8 :: 1+2+4 = 7 - 8 is not perfect.

3) Write a java program to print the given number in words.

ex- 486 = four Eight six

(or) four hundred Eighty six.

* Constructors in java

A constructor is a special member method which is automatically calling by the JVM for placing our own values whenever an object is created without placing the default values.

The basic purpose of constructors in java is to initialize an object.

Advantages of constructors -

1) constructors elliminates in placing default values,

2) constructors elliminates in calling ordinary methods explicitly.

Rules/ properties / characteristics of constructors -

1) When we use the constructors as a part of our java program, we need to follow the following rules.

2) Constructor will be called automatically whenever an object is created, the name of the constructor must be similar to class name.

3) Custructors should not return any value even void also. (If we write void as a return type of the constructor then it is treated as ordinary method).

Program for to decide wheather the given number is perfect or not.

```
class perfect
{
  int n;
  void set (int x)
  {
   n=x;
  }
  void decide ()
  {
    int b, i, j=0;
   for (i=1; i<=n/2; i++)
    {
     b = n%i;
     if (b==0)
     {
     j=j+i;
     } // if
   } // for
```

```
if(j==n)
  System.out.println("NO is perfect");
else
  System.out.println("NO is not perfect");
} // decide
} // class
class perfectDemo
{
public static void main(String args[])
{ int y = Integer.parseInt(args[0]);
  perfect p=new perfect();
  p.set(y);
  p.decide();
} // main
} // class
```

4) constructor should not be static because constructors must be called each & and every time as and even object is created.

5) constructors will not be inherited from one class to another class because every constructor of a class must be used for initializing data members of the same class but not other class data members.

6) A constructor can be private provided an object of that class can not be created in the context of some other classes. A constructor cannot be private provided an object of one class can be created in the context of

some other classes.

Types of constructors-

In java programming, we have two types of constructors, they are;

1) Default/ parameterless/ zero argument constructor.
2) Parameterised constructor.

Default constructor -

```
Test t1 = new Test()
```

A default constructor is one which never takes any parameters.

Default constructor is highly recommanded to take, to place same values in different objects of same class.

Syntax -

```
class <classname>
{
  ---
}
<classname>() // Default Constructor
{
  Block of Statements
}
}
```

Write a java program which list the concept of default constructor.

```
Class Test
{
 int a,b;
 Test()
 {
SOP("Test-DC");
a=10;
b=20;
S.O.P(val of a="+a);
S.O.P(val of b="+b);
 }
}// Test

class TestDemo
 {
  public S V m(String args)
  {
  Test t1= new Test();
  }
  }
```

Rule1 - Whenever we create an object only with default constructor, defining the default constructor is optional. If we are not defining default constructor then jvm will call it's own constructor called System defined default constructor and it is placing default values

depends on data members of class. If we define our own default constructor then JVM will call user / programmer defined default constructor for placing our own values.

Parameterised constructor -

A parameterised constructor is one which always takes parameters.

If we want to place different values for different objects of same class then we must use parameterised constructor.

Syntax -

```
class <classname>
{
  —
  <classname> ( list of formal
                parameters)
  {
    Block of statement(s)
  }
}
```

Write a java program which illustrate the concept of parameterised constructor.

```
class Test
{
  int a,b;
  Test( int x, int y)
  {
    S.O.P( "Test-PC");
```

```
        a=x;
        b=y;
        SOP("val of a="+a);
        SOP("val of b="+b);
        }
    }//Test

    class TestDemo
    {
    public static void main(String[] args)
    {
    Test t1= new Test(1,2);
    Test t2= new Test(10,20);
    Test t3= new Test(100,200);
    }
    }
```

Rule 2 -

Whenever we create an object with parameterised constructor, defining the parameterised constructor is mandatory.

Imp for SCJP

** Rule 3 -

Whenever we create multiple objects with both default & parameterised constructors, it is mandatory for the java programmer to define both the constructors otherwise we get compile time error.

Overloaded constructors:-

A constructor is said to be overloaded if and only if constructor name is same but its signature is different.

Signature represents one thing

(a) No. of parameters.
(b) Type of parameters
(c) Order of parameters

} At least ^ must be differenciated.

example-

```
Test t1 = new Test(10,20); —(1)
Test t2 = new Test (100); —(2)
Test t3 = new Test (10.25f, 20.75f) —(3)
Test t4 = new Test (100, 10.75f) —(4)
Test t5 = new Test (10.75f, 200) —(5)
```

In the above example, the constructor Test is known as overloaded constructor.

Object Parameterized constructor-

An object parameterized constructor is one which always takes objects as parameter. object parameterized constructor is used for copying the content of one object into another object provided both the objects must belongs to same type.

Let us consider the following,

```
test t2 = new test (10,20) —(1)
test t4 = new test (t2); —(2)
```

t2: a = 10, b = 20

t4: a = 10, b = 20

The statement ② represents object parameterized constructor i.e. it copies the content of t2 into an object t4. The objects t2 and t4 belongs to same type i.e. Test.

Therefore in java programming to copy the content of one object into another object, we use the concept of object parameterized constructor. Which exactly functioning like copy constructor of C++.

Write a java program which illustrate the concept of default parameterized, overloaded and object parameterized constructor.

```
class Test
{
 int a,b;
 Test()
 {
  SOP("Test - Default");
  a=1;
  b=2;
 SOP("Val of a="+a);
 SOP("Val of b="+b);
}

Test(int x)
{
 SOP("Test-SPC");
```

```
    a=x;
    b=x;
    SOP(" val of a=" +a);
    SOP(" val of b=" +b);
  }

  Test(int x, int y)
  {
    SOP (" Test-DPC");
    a=x;
    b=y;
    SOP (" val of a=" +a);
    SOP (" val of b=" +b);
  }

  Test( Test T)
  {
    SOP (" Test-DPC");
    a = T.a;
    b = T.b;
    SOP (" val of a=" +a);
    SOP (" val of b=" +b);
  } // Test

class TestDemo
{
  psvm (String args[])
  {
    Test t1 = new Test();
    Test t2 = new Test(10,20);
    Test t3 = new Test(100);
    Test t4 = new Test(t2); // object Parameterized constructor
  }
} // class.
```

The default parameter passing mechanism in java is call by reference whereas in C, C++ the default parameter passing mechanism is call by value.

In a class if we are not defining constructors then JVM will place by default a constructor on its own, known as system defined default constructor.

If any constructor of a class is private then that constructor will not be shown as a part of profile.

In general private features of a class can not be shown as a part of the profile, profile of a class is nothing but data members, constructors and various types of methods.

Programatically a class of java can contain a single default constructor and multiple parameterised constructors (overloaded constructors).

To see the profile of a class/ interface, we use javap. javap is one of the application program or exe file or tool used to see the profile of the class and it is supplied as a part of jdk1.5/bin folder.

Syntax - javap name of the class.

example -

javap Test

⇒ class Test extends java.lang.object{
int a;
int b;
Test ();
Test (int);
Test (int, int);
Test (Test);

24/2/2011

24/2/11

This keyword -

This is one of the implicit keyword/ object created by the JVM and supplied to every java program for two purposes. They are,

1) It points to current class object

2) It whenever formal parameters and datamembers of the class are similar then JVM gets an ambiguity (not having clarity to identify which are data membe of the class and which are formal parameters).

In order to differenciate between formal parameters and datamembers of the class, as a java programmers data members of the class must be preceded by a keyword "this".

Syntax -

this. current class data member.

ex. Write a java program which illustrate the concept of 'this' keyword.

```
class sample
{
  int a,b;
  Sample(int a, int b)
  {
    this.a=a;  } → Assigning formal parameter values to data members of the class.
    this.b=b;  }
    a=a+1;     } → Modifying the values of the formal parameters.
    b=b+1;     }
    this.a=this.a+1;  } Modifying the values of the data members of the class.
    this.b=this.b+1;  }
  SOP("val of a(fp)="+a);
  SOP("val of b(formal Parameter)="+b);
  }
  void disp()
  {
  SOP("val of a="+this.a);  } // Writing this keyword is optional
  SOP("val of b="+this.b);  }
  }
}// Sample
class Sdemo
  {
    psvm(String args)
    {
    Sample S1=new Sample(10,20)
    S1.disp();
  }
}
```

2. Write a java program which will compute sum of two objects.

```
class Test
{  int a,b;
   Test(int a, int b)
    {
      this.a = a;
      this.b = b;
    }
   Test sum(Test T)
    {
      Test t11 = new TEST(); // local object
      t11.a = this.a + T.a;
      t11.b = this.b + T.b;
      return (t11);
    }
   void disp()
    {
      SOP(" val of a="+a);
      SOP(" val of b="+b);
    }
~~}// Test~~
    Test()
    {
// a=b=0;
    }
}// Test

class objectsum
  {
    P S V main(String[] args)
     {
     int x1 = Integer.parseInt(args[0]);
     -"- x2 = -"- -"- (args[1]);
     -"- x3 = -"- -"- (args[2]);
     -"- x4 = -"- -"- (args[4]);
```

Here optional

```
Test T1 = new Test (x1,x2)
Test T2 = new Test (x3,x4)
Test t3 = new Test();
// t3 = t1 + t2 ..... invalid stmt
 SOP(" t1values");
 t1.disp();
 SOP (" t2 values:");
 t2.disp();
 SOP (" object sum:");
 t3.disp();
 }
}
```

In java programming operator overloading facility is not available, but we can solve all the operator overloading kind of problems with a simple concept of methods.

A method is not only returning fundamental values and predefined classes but also returns user defined class names as a return type.

Fri, 25/02/2011

this(), this(...)

We know that one method can call another method either with respect to object or with respect to class.

i.e. we can establish the communikath between multiple methods.

Similarly, in order to establish the communication between constructors

of the current class, we must use two types of functions. They are

ⓐ this()

ⓑ this(...)

a) this() :- It is used for calling current class default constructor from current class parameterised constructors.

b) this(...) :- It is used for calling the current class parameterised constructors from other category constructors of same class.

Rules :-

1) When we are using either this() or this(...) in the current class constructors they must be used always as a first executable statement otherwise we get compile time errors.

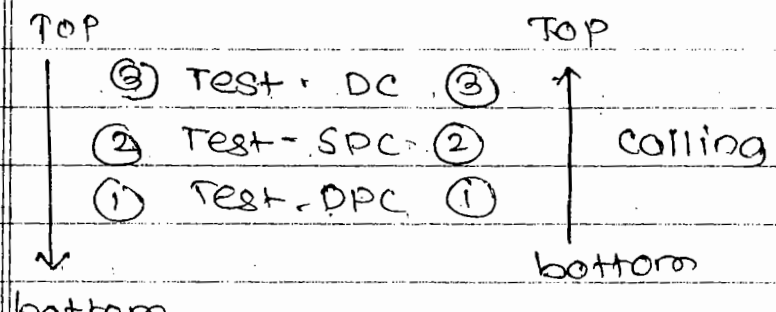
2) No cycle to be formed but recommand to form only path when we are using either this() or this(...) in the current class constructors.

3) No ~~two~~ consecutive this(), this(...) to be used in the current class constructors because every this(), this(...) want to have as a first executable statement in the current class constructors.

Write a java program which illustrate the concept of this() & this(...).

```
class Test
{
 int a,b;
 Test() //-----(2)
 {
  this(1000);// control goes to (3)
  SOP("Test-DC");
  a=1;
  b=2;
 SOP("val of a=" +a);
 SOP("val of b=" +b);
 }
 Test(int x) //------(3)
 {
  SOP("Test-SPC");
    a=x;
    b=x;
  SOP("val of a=" +a);
  SOP("val of b=" +b);
 }
 Test(int x, int y) //---(1)
 {
  this();// control goes to (2)
  SOP("Test-DPC");
  a=x;
  b=y;
  SOP("val of a=" +a);
  SOP("val of b=" +b);
 }
} //class

class TestDemo
 {
```

```
    p s v m(String[] args)
    {
    Test t1 = new Test(100,200); // control goes to (1)
    }
}
```



In the above program, constructors are calling in the order of ①②③ (increasing order) and executing in the order of ③, ②, ① (decreasing order).

In general, in any java program, constructors are calling from bottom to top and executing from top to bottom.

Note:- Whenever we use two consecutive this() & this(...) in the current class constructor syntactically which is wrong but it can be solved programatically by using the above rule.

Note:- Whenever we want to have multiple values in the same object, it is recommanded to establish the communication between multiple constructors of the same class by making use of this() and this(...).

Date:- 28-Feb-2011

* Inheritance *

- Inheritance is one of the distinct feature of object Oriented Programming.
- If we use Inheritance concept, as a part of our Java programming we get various advantages.

* Definition -

① The process of objetaining the data members and methods from one class to another class is known as Inheritance.

② The process of creating a new class from the old class is known as Inheritance.

- The class which is giving data members and methods to some other class is known as Base or Super or Parent Class.
- The class which is getting data members and methods ~~for~~ ~~som~~ from some other class is known as Derived or Sub or Child class.
- The data members & methods of a class is known as features of Class.
- The concept of Inheritance is also known as Sub-classing or Extendable Classes or Derivation or Re-usability.

* Advantages -

When we use inheritance concept as a part of our java program, we get the following advantages

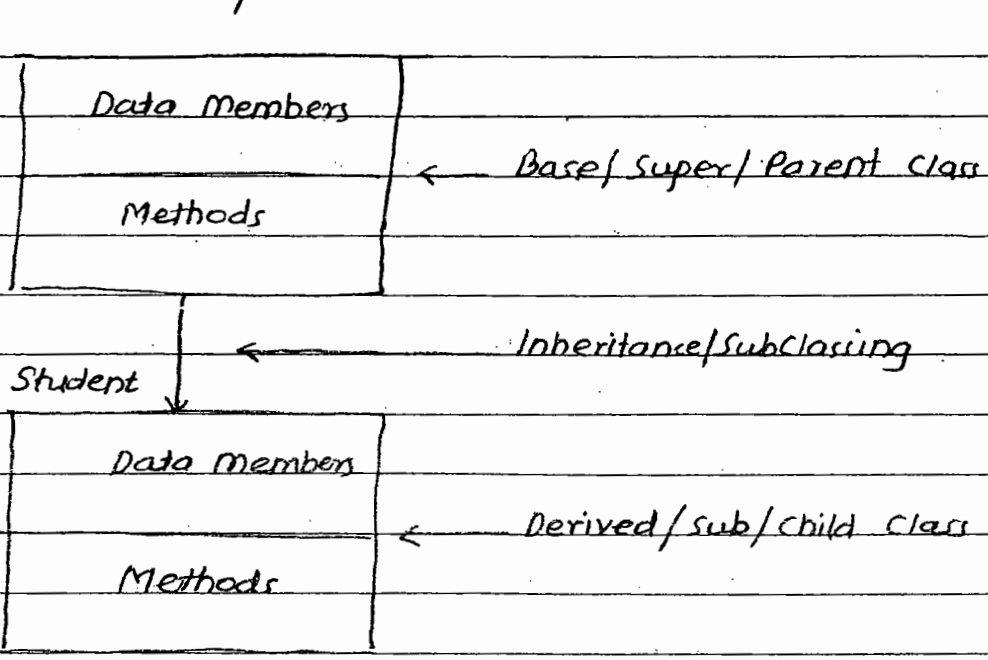
i) Application Development time is less.

ii) Application Memory space is less.

iv) The execution time of an application is less.

v) The redundancy (Repeatation) of the code is reduced. So that we get less memory cost and consi-
~~So th~~-stent result.

vi) On overall, we can achieve the slogan of Java i.e. WORA (Write-Once Reuse Anywhere).



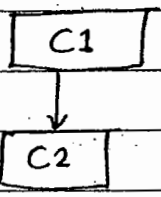
* Types of Inheritance - (Reusable Techniques) -

The number of ways we inherit the features of various classes into derived classes are known as Re-usable Techiniques or Inheritance Types.

In Java Programming, we have five inheritance types. They are.

1) Single Inheritance -

single base class and single Derived class



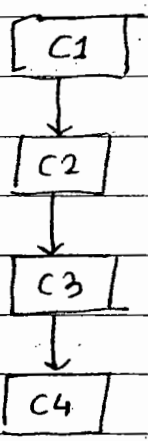
* Inheritance Path -

C1 → C2

2) Multilevel Inheritance -

Multilevel Inheritance is one in which there exist single base class, single derived class and multiple intermediate base classes.

* Diagram -



* Inheritance Path -

C1 → C2 → C3 → C4

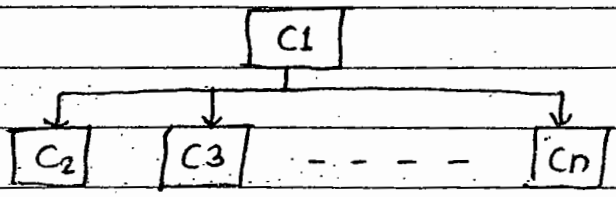
* Define Intermediate Base Class -

An intermedicde base class is one, in one context it access base classes and in another context it access derived class.

Date- 01-03-2011

3) Hierarchical Inheritance -

Hierarchical Inheritance is one in which there exist single Base Class and multiple Derived Classes



Inheritance Path -

C1 → C2

C1 → C3

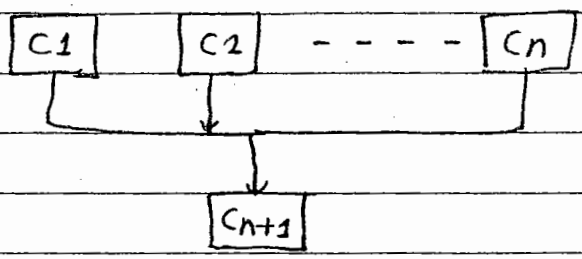
⋮

C1 → Cn

4) multiple Inheritance -

Multiple Inheritance is one in which there exist multiple Base Classes and single Derived Class

* Diagram -



* Inheritance Path -

C1 → Cn+1 , C2 → Cn+1 , - - - - ,

Cn → Cn+1

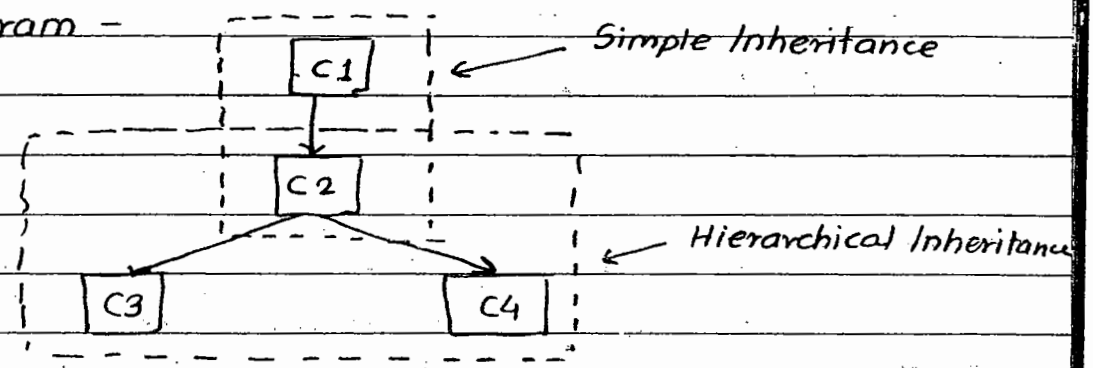
* Note: Java Programming never support the concept of multiple Inheritance with respect to Classes. But it can be supported by Java with a concept of an

5) Hybrid Inheritance -

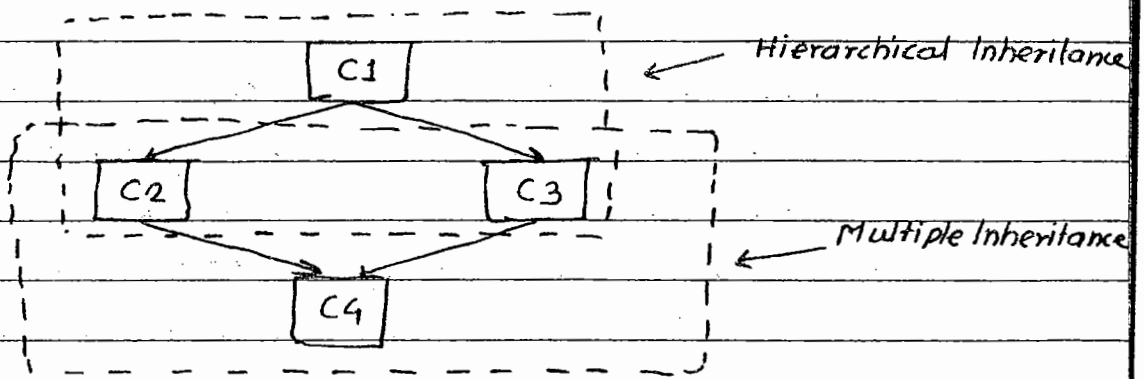
Hybrid Inheritance = Combination of Any available Inheritance Types.

In the combination, if one of the combination is multiple Inheritance then the entire combination is not supported by the concept of Classes. But it can be supported by the concept of Interfaces.

* Diagram -



Valid by both Classes and Interfaces Concept.



Invalid by Classes Concept.
Valid by Interface Concept.

* Syntax for Inheriting features of Base Class to Derived Class -

class <classname 2> extends <classname 1>
{
Variable Declarations;

* Syntax Explanation –

In the above syntax. –

i) <classname-1> & <classname 2> represents the name of the Base and Derived Classes respectively.

ii) Extends is a keyword used for inheriting the Features of base class into derived class plus it is improving the functionality of derived classes.

iii) One derived class can always extends only one base class.

iv) Whenever we develop any inheritance applications, it is highly recommended for the Java Programmer to create an object of bottom most derived class Because it contains all the features of top most super class and multiple intermediate base classes

v) Whenever we create an object of bottom most derived class, First we get the memory space for data members of top most base class, second, we get the memory space for data members of intermediate base classes and at the last we get the memory space for data members of bottom most derived class.

vi) If we don't want to give the features of base class to the derived classes, then the definition of the base class must be made as Final. Hence final base classes, never participates in inheritance process

vii) If we don't want to give some of the features of base class to the derived class then those features of the base class must be made as private. Hence, private features of the base class never participates in the inheritance. In other words, the private features of base class cannot be accessed in derived class context.

viii) The constructors of the base class will not be inherited into derived class, because constructors are meant for initializing its own class data members but not other class data members.

ix) An object of base class always contains details about its own class. But an object of base class never contains details about special features of derived class.

x) For each & every class in Java, there exist an implicit predefined super class known as java.lang.Object class. Because java.lang.Object provides grarbage collector program to its subclasses for collecting unused memory space for improving the performance of java applications.

Prog#16- Write a Java Program which illustrate the concept of Inheritance.

→

```
Class Base
{
```

```
        void display()
        {
            System.out.println("Display : Base");
        }
    } // End of Base

class Derived extends Base
{
    int b;
    void show()
    {
        display();
        System.out.println("Show : Derived");
        a = 100;
        b = 200;
        System.out.println("Value of A = " + a);
        System.out.println("Value of B = " + b);
    }
} // End of Derived

class InheritanceDemo
{
    public static void main (String args[])
    {
        Derived d = new Derived();
        // d.display();
        d.show();
    } // End of main
} // End of InheritanceDemo
```

Date - 02 - MAR - 11

Types of Relationship in Java -

Types of relationships always makes us to understand how to get the features of one class into another class. In other words, types of relationships allows us to share the data between multiple classes.

In Java programming we have three types of relationships. They are -

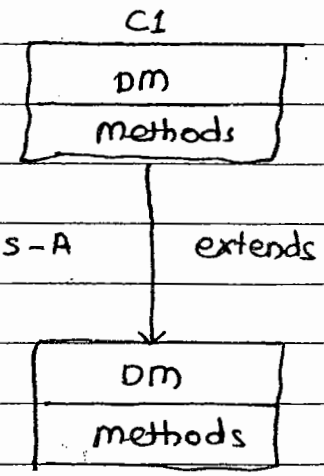
i) Is-a Relationship
ii) has-a / A-part-of relationship
iii) Uses-a Relationship.

1) Is-a Relationship -

Is-a relationship one in which the features of one class is obtaining from another class with the concept of Inheritance by using extends keyword.

Example: -

```
class C1
{
    =
}

class C2 extends C1
{
    =
}
```



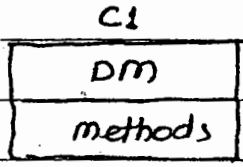
2) Has-a Relationship -

Has-a relationship is one in which an

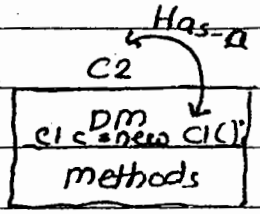
Some other class either as an Instance

Example -

```
class C1
{
    =
}
```




```
class C2
{
    C1 co = new C1();
    =
}
```

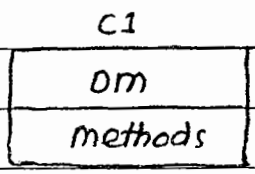


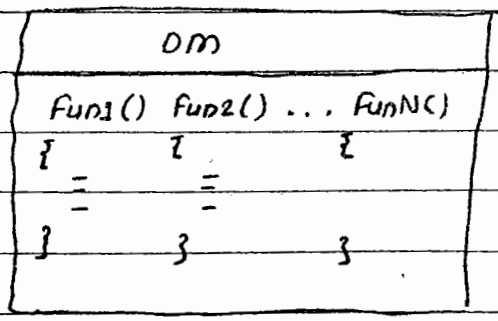
3) Uses - a Relationship -

A Uses - a relationship is one in which a method of one class is using an object of another class.

Example -

```
class C1
{
    =
}
class C2
{
    =
    void Func()
    {
        C1 co=new C1();
        =
    }
}
```





Date - 03-MAR-2011

1) Super at Variable level -

Whenever we inherit the data members of base class into the derived class, there is a possibility that those class members are similar to derived class data members. JVM is unable to differentiate between base class data members and derived class data members.

In order to differentiate between base class data members and derived class data members, in the context of derived class, base class data members must be preceded by a keyword 'super'.

Syntax: -

super.<Base class Data member Name>;

Prog #17 Write a java program which illustrate the concept of 'super' keyword.

→

```
Class ~~Idemo2~~ Base
{
    int a;
} // End of Base

Class Derived extends Base
{
    int a, c;
    void set(int x, int y)
    {
        super.a = x;
        a = y;
    }
```

* Note -

i) The default relationship in java is IS-a relationship. Because each & every class in java there exist an implicit predifined super ~~call~~ class called java.lang.Object.

ii) System.out is the universal example For Has-a relationship. (Out is object of PrintStream Class and it is created as Static data member in another class called system.)

iii) "Each & every execution logic method of execution logic class is making use of Busines logic Class objects." - Comes under Universal Example of Uses-a relationship.

* Super keyword *

Whenever we inherit the base class features into derived class, there is a possibility that base class features may be similar to derived class features. In this context, JVM is unable to differentiate between base class features & derived class features.

In order to differentiate between base class features and derived class features, the base class features must be preceded by a keyword 'super'.

In other word, Super is a keyword which will differentiate base class features with derived class features.

The keyword Super is placing an important role in three places in Java. They are -

i) at variable level ... iii) at contructor level.

```
    void sum()
    {
        c = super.a + this.a;
    }
    void display()
    {
        System.out.println("Value of a From Base = "+
                                                super.a);
        System.out.println("Value of a From Derived = "+a);
        System.out.println("sum = "+c);
    }
} // End of Derived.

class IDemo2
{
    public static void main(String args[])
    {
        int x = Integer.parseInt(args[0]);
        int y = Integer.parseInt(args[1]);
        Derived d = new Derived();
        d.set(x, y);
        d.sum();
        d.display();
    } // End of main
} // End of IDemo2
```

2) Super at method level -

whenever we inherit the methods of base class into a derived class, there is possibility that base class methods are similar to derived class methods So that JVM is unable to differentiate between base

In order to differentiate between base class methods and derived class methods, in the context of derived class, base class methods must be preceded by a keyword 'super'.

Syntax: super.BaseClassMethodName();

Prog #18 Write a java program which illustrate the concept of 'super' keyword at method level.

→

```
class Base
{
    void display()
    {
        System.out.println("Base : Display()");
    }
} // End of Base

class Derived extends Base
{
    void display()
    {
        super.display(); // will call Base Class Display().
        System.out.println("Derived : Display()");
    }
} // End of Derived

class IDemo3
{
    public static void main(String args[])
    { Derived dc = new Derived();
```

In the above program, if we eliminate 'super' keyword then display method of Derived class called by inifinitely. Which leads to bad recursion.

In Java programming if same function called by itself infinitely then we get an error at runtime. known as StackOverFlowError. (This known as Exception.)

Q] How do you call a base class method from the context of derived class when both methods are same.

→ super.BaseClassMethodName();

3) Super at method overriding -

Method Overriding = Method Heading is same + Method Body is different.

The process of redefining the original method into various derived classes by inheriteoing the original method from base class for performing various operations.

Prog # 19 Write a Java program which illustrate the concept of 'super' keyword at method overriden.

→

```
class Base
{
    System.void show()
    {
        System.out.println("show() - Original");
    }
} // End of Base
```

```
class IBase etxtends Base
{
    void show()
    {  super.show();
       System.out.println("show() - Redefined First");
    }
} // End of IBase

class Derived extents IBase
{
    void show()
    {
       super.show();
       System.out.println("show() - Redefined second");
    }
} // End of Derived.

class IDemo4
{
    public static void main(String args[])
    {
       Derived d = new Derived();
       d.show();
    } // End of main
} // End of IDemo4
```

Output:

```
show() --- Original
show() --- Redefined First.
show() --- Redefined Second.
```

In Java programming, original methods always presents in base classes, and overriden methods always presents in derived classes.

Q] How do you call an original method of base class? From the context of overriden method of derived class?

→ super.OriginalMethodName();

The above statement must be written within the context of overriden method of derived class.

Date - 04 - MAR - 2011

4) Super at Constructor Level -

① Whenever we develop any inheritance application, it is highly recommend for the Java programmer to create an object of bottom most derived class. Because it is getting features from intermediate base classes and its top most base classes.

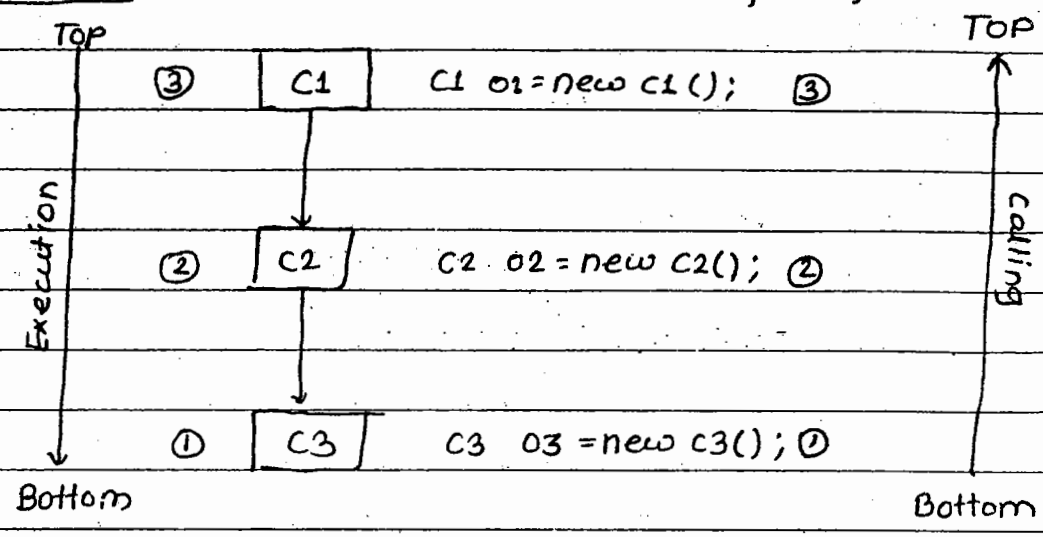
② When we create an object of bottom most derived class, first we get the memory space for base class data members and second we get the memory space for intermediate base class and at last we get the memory space for derived class.

③ In whichever the order memory space is created in the same order initialization process has to be taken class.

④ we know that initialization of the data members must be done by constructors.

The above four points makes us to understand,

Example: Let us consider the following diagram.



In the above diagram, the default constructor of C3 is automatically calling default constructor of C2 and default constructor of C2 is automatically calling default constructor of C1. Constructor of C1 will be executed first and C2 default constructor will execute second and at last default constructor of C3 will be executed.

In general, in any inheritance applications the way of calling the constructors is from bottom to top (Increasing order) and they are executing in the order Top to Bottom (Decreasing Order).

Prog #20 Write a Java Program which illustrate the concept of constructors are calling from Bottom to Top and executing from Top to Bottom.

→

```
class C1
{
    C1() //- - - - - - -> ③
```

```
class C2 extends C1
{
    C2() //<------------②
    {
        super(); // Controls goes to ③ --Optional
        System.out.println("C1 - Intermediate Base");
    }
} //End of C2

class C3 extends C2
{
    C3() //<------------①
    {
        super(); // Control goes to ② --Optional
        System.out.println("C3 - Bottom most Derived");
    }
} // End of C3

class IDemo5
{
    public static void main(String args[])
    {
        C3 O3 = new C3(); // control goes to ①
    } // End of main
} // End of IDemo5
```

In order to establish the communication between base class constructors with derived class constructors, we have two functions. They are -

i) super()

ii) super(...)

① super() -

It is used for calling base class default constructor from derived class constructors.

When we are super(), in the context of derived class, JVM will automatically call base class default constructor. The usage super() in the context of derived class constructor is optional. Because Base class contains only one default constructor.

② Super(---) -

It is used for calling Base class parameterized constructor from derived class constructors.

The usage of super(---) is mandatory in the derived class constructors.

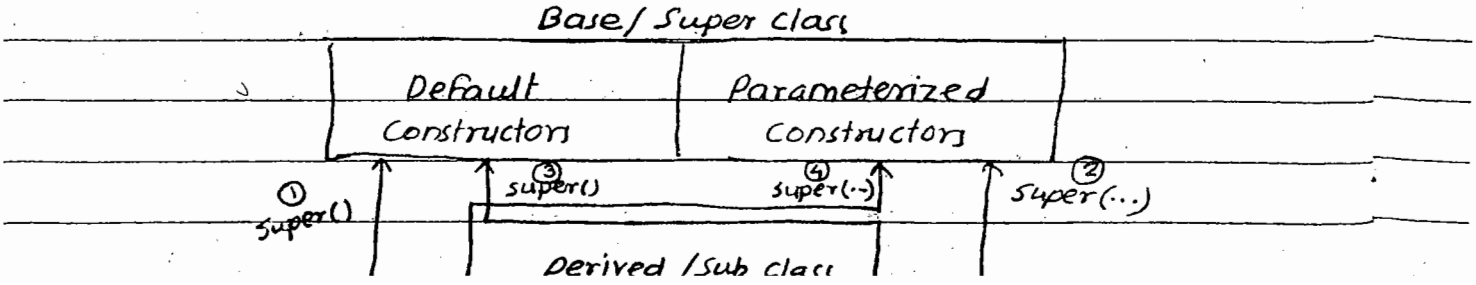
* Rules -

Whenever we use super(), super(---) in our program then the programmer must Follow the following rules.

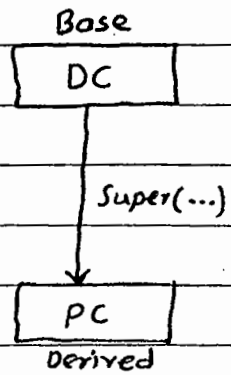
① Whenever we use super() or super(---) in the derived class constructor then they must be used as first executationable statement.

② No two consecutive super() and super(---) will be used in the context of derived class constructors

③ Possibilities of using super(), super(---) are

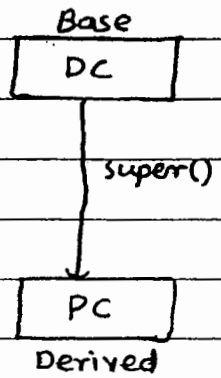


Prog#22 Write a Java Program which illustrate rule ② of Figure-1.




```
class Base
{
    int a;
    Base (int a) // ----------> ②
    {
        System.out.println("Base : PC");
        this.a = a;
        System.out.println("Value of a = " + this.a);
    }
} End of Base

Class Derived extends Base
{
    int b;
    Derived (int a, int b) // ----------> ①
    {
        ~~System.out.println(~~
        Super(a); // Control goes to ② -- Mandatory.
        system.out.println(" Derived : PC");
        this.b = b;
        System.out.println(" Value of b = " + this.b);
    }
} // End of Derived.

class Sdemo2
{
    public static void main (String args[])
    {
        Derived d = new Derived (10,20); // control goes to ①
    } // End of main
```

Prog #23 Write a Java Program which illustrate Rule ③ of Pre Figure.




```
class Base
{
    int a;
    Base()  ------> ①
    {
        System.out.println("Base: Default");
        a = 1;
        System.out.println("value of a = "+a);
    }
} //End of Base

class Derived extends Base
{
    int b;
    Derived(int b) //  ------> ①
    {
        //super(); // Control goes to ② -- Mandatory Optional.
        System.out.println("Derived: Parameterized");
        this.b = b;
        System.out.println("value of b = " + b);
    }
} // End of Derived.

class Sdemo3
{
    public static void main (String args[])
    {
        Derived d = new Derived (2); // Control
                                     // goes to ①
    } // End of main
} // End of Sdemo3
```

Rules ① & ③ -

Whenever the constructors of the derived class want to call default constructor of base class then we must be use super() in the context of derived class constructors.

The usage of super() in the derived class constructors is optional.

Rules ② & ④ -

Whenever we want to call parameterized constructor of Base class from the context of derived class constructors then we must use super(...) in the derived class constructors.

The usage of super(...) in the derived class constructor is mandatory. Because a base class may contain multiple parameterized constructor.

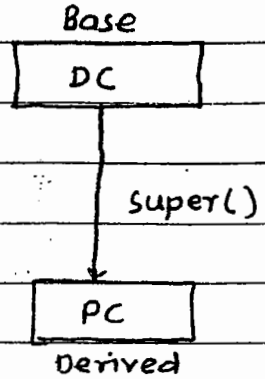
~~Date: 05 MAR-11~~

Q] What are the differences between this(), this(---) and super(), super(---) ?

→ This(), this(---) are used for establishing the communication between current class constructors. Whereas super(), super(---) are used for establishing the communication between superk class constructors with subsclass constructors.

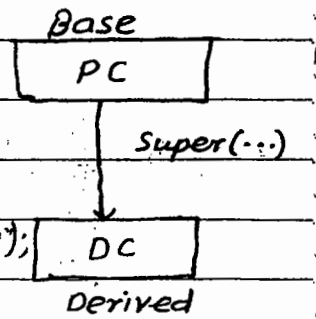
Date- 05-MAR-2011

Prog #21 Write a java program which illustrates the Rule ① Pr
of Figure-1.




```
class Base
{
    int a;
    Base() // ----------> ②
    {
        System.out.println("Base: Default");
        a=10;
        System.out.println("Value of a From Base = "+a);
    }
} // End of Base

class Derived extends Base
{
    int b;
    Derived () // ----------> ①
    {
        //super(); // Control goes to ② -- Optional.
        System.out.println ("Derived : Default");
        b=20;
        System.out.println ("Value of b = " + b);
    }
} // End of Derived.

class sdemo1
{
    public static void main (String args[])
    {
        Derived d = new Derived (); // Control goes to ①
```

Prog #24 write a Java program which illustrate Rule ④ of Figure - ①




```
class Base
{
    int a;
    Base (int a) //————————— ②
    {
        System.out.println("Base: Parameterized");
        this.a = a;
        System.out.println ("Value of a "=" + this.a);
    }
} // End of Base.

class Derived extends Base
{
    int b;
    Derived () //————————— ①
    {
        super(1000); // Control goes to ② -- Mandatory
        b = 200;
        System.out.println("Value of b = " + b);
    }
} // End of Derived

class Sdemo4
{
    public static void main (String args[])
    {
        Derived d = new Derived (); // Control goes
                                    // to ①
    } // End of Main
} // End of Sdemo4
```

In the above programs, constructors are calling 1, 2 and executing in the order of 2, 1.

Prog #25 Write a Java program which illustrate the concept of this(), this(...), super() and super(...).

→

```
class Base
{
    Base()   // ⟶ ⑤
    {  this(100);
       System.out.println("X");
    }
    Base(int n)  // ⟶ ⑥
    {
        System.out.println("Y");
    }
} // End of Base

class IBase extends Base
{
    IBase()  // ⟶ ④
    {
        super(); // Control goes to ⑤
        System.out.println("P");
    }
    IBase(int n)  ⟶ ③
    {
        this(); // Control goes to ④
        System.out.println("Q");
    }
} // End of IBase.
```

```
class Derived extends IBase
{
    Derived ()  //  ------------> (2)
    {
        super(10); // Control goes to (3)
        System.out.println("M");
    }
    Derived (int n)  //  ------------> (1)
    {
        this(); // Control goes to (2)
        System.out.println("N");
    }
} // End of Derived

public class StDemo
{
    public static void main (String arg[])
    {
        Derived d = new Derived (200); // Control
                                       goes to (1)
    } // End of main.
} // End of StDemo.
```

In the above program, constructors are calling in increasing order (1, 2, 3, 4, 5, 6) and executing in decreasing order (6, 5, 4, 3, 2, 1).

Date - 07-MAR-11

* Polymorphism *

Polymorphism is one of the distinct facility of the object oriented Programming.

Definition -

"The process of representing one form in the multiple form" is known as Polymorphism.

① In Java programming the concept of polymorphism can be implemented programatically by one concept of method overloading.

② We know that method overriding is nothing but method binding is same & whoose boys is different.

In other words, the process of the original method into multiple forms for performing multiple operations is known as Method overriding.

According to OMG principle of polymorphism is classified into ~~too~~ two types. They are. -

i) Static or Compile time Polymorphism.

ii) Dynamic or Runtime Polymorphism.

i) Static Polymorphism -

The process of binding the method with an object at compile time is known as Static Polymorphism

* Drawback of Static Polymorphism -

Utilization of resources (memory), CPU Time) etc. are very poor.

To eliminate the above problem, we use an the concept of Dynamic Polymorphism.

by using Function overloading & operator overloading then that programs satisfies static polymorphism principle.

In real word, C++ environment, C++ progra-mmer never follows static polymorphism principle by only follows Dynamic Polymorphism.

2) Dynamic Polymorphism. -

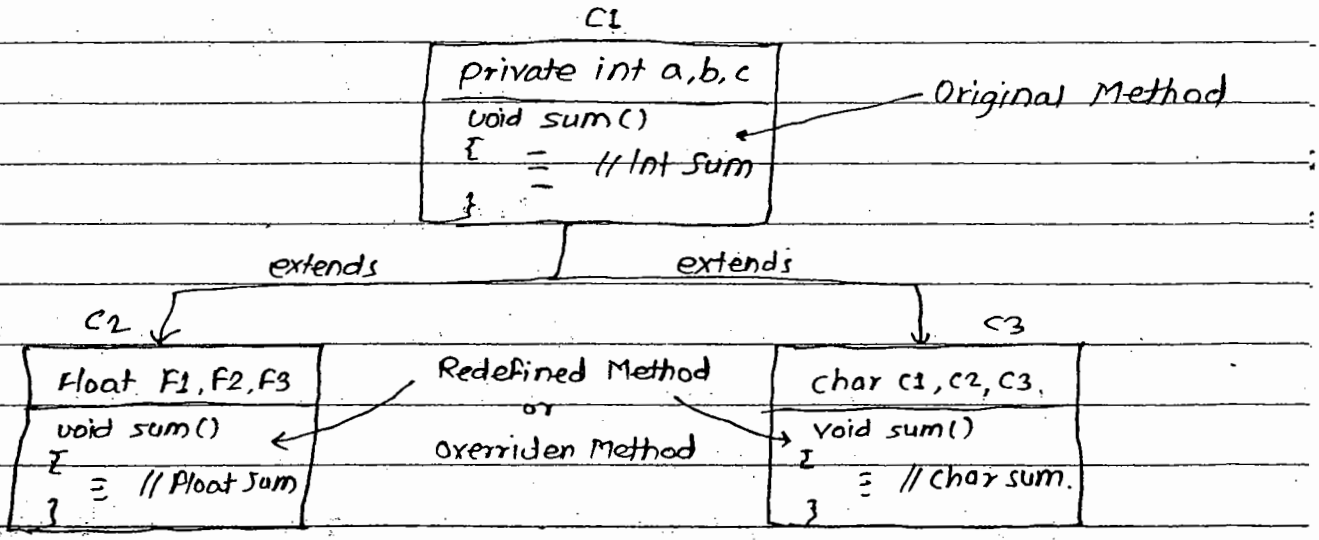
A Dynamic Polymorphism is one in which methods will binds with an object at run time.

Java programming never follows static polymorphi-sm but it always follows Dynamic Polymorphism. The basic advantages of Dynamic Polymorphism is to utilize the resources efficiently.

Q] Define workspace -

→ The amount of temporary memory space created by O.S. in the stack memory for temporary execution of the method is known as workspace.

* Consider the following diagram.



Date - 08-MAR-2011

Most of the realtime applications will be developed with the concept of Polymorphism along with the method overriding concept.

To execute the polymorphism applications, it is highly recommended to use dynamic binding concept (Execution Logic).

* Dynamic Binding -

Dynamic Binding is one of the distinct feature of Object Oriented Programming.

i) Dynamic Binding always says that "Don't create the objects of derived classes but create the objects of base classes".

ii) Dynamic is alway used to execute polymorphism applications.

iii) The basic advantage of dynamic binding concept is that to minimize the application memory space, So that performance of an application is improved.

Definition -

"The process of binding appropriate versions (overiden methods) of the derived classes which are inherited from base class with base class object is known as Dynamic "Binding.

while we are executing polymorphism applications by making use of dynamic binding, internally JVM consider two points.

when we create an object indirectly then JVM consider internally two points.

ii) what type of address reference object contains?

* Let us consider the following set of executable state-ments for executing the problem which is represented in the Figure. 1

```
class DBDemo
{
    public static void main(String []args)
    {
        ① C1 O1 = new C1();
        ② O1.sum();
        ③ O1 = new C2();
        ④ O1.sum();
        ⑤ O1 = new C3();
        ⑥ O1.sum();
    }
}
```

In the above program -

In statement no. 1, 3, 5, the object O1 contains addresses of multiple classes. (C1, C2 & C3). Hence the object O1 is known Polymorphic Object.

Originally, the method is available in one form (original form) but it is represented in multiple form (overriden methods). Hence some method sum() method is known as Polymorphic method.

In the statement no. 2, 4, 6 'sum() method is actually a single statement but it gives different sums (int sum, Float sum, char sum). Hence, O1.sum() is known as Polymorphic statement.

Hence, in most of realtime application development

applications must be executed with the concept of Dynamic Binding.

If we use these principles as a part of any object Oriented Programming language, whoose application will get advantages of there principles

* Abstract Classes *

- Each and every ~~elas~~ program in Java must be written with the respective to the concept classes.
- In Java Programming, we hava types classes. They are -
  a) Concrete Classes.
  b) Abstract Classes.
- A concrete class is one which contains Fullydefined methods.
- Fully defined methods of class are alsom known as implemented / concrete methods.
- One the classes is concrete, we can instantiate (create) an object of that class directly.

Ex.

```
Class C1
{
    void show();
    {
        System.out.println("C1.show");
    }
    void display()
    {
        System.out.println("C1.display");
    }
} // End of C1
```

Here, C1 is a concrete and hence who's object can be instantiated directly.

Ex.

```
C1 O1 = new C1();
O1.show();
O1.display();
```

- An abstract class is one which contains some defined & un-defined methods.
- Undefined methods of a class are also known as unimplemented / abstract methods -

* Definition -

A Abstract Method is one which does not contains a body or definitions but it contains only prototypes / declarations of a method.

In order to make, any undefined methods a abstract method, the prototype of that method must be preceded by a keyword call 'abstract'.

* Syntax -

```
abstract <ReturnType> <Method_Name> (list of
        formal params if any);
```

* Example -

```
abstract void sum();
abstract void operation(int x, int y);
```

- Abstract methods always makes use to understand the following

@ What a method can do ?

- All the methods of Java must be placed in a class including abstract methods.
- Whicherer class contains abstract methods such class is known as Abstract class.
- To make class as 'abstract', the definition of the class must be preceded by a keyword 'abstract'

Syntax:

```
abstract class <classname>
{
    variable declarations;
    methods definitions /declarations;
}
```

Example -

```
abstract class Op
    {
        abstract void operations(int x, int y);
        void show()
        {
            System.out.println("show Op");
        }
    }
```

Here, the class Op is an abstract class so that, creating an object of op classes directly not possible. But it can possible indirectly.

Ex.

```
Op O1 = new Op(); // Invalid.
```

Hence, in Java with respect to an abstract class one can not instantiate an object directly. But it can instarstiated indirectly.

Date: 11-Mar-2011

When we use concrete classes in our application development, there is a possibility that memory space is wasted, and it may leads to less performance of the application.

If we use abstract classes as a part our application development, there is a possibly that application memory space can be reduced and it results in higher performance.

* Points to be remember -

1) Abstract Classes are always reusable / Inheritable / extendable.

2) Abstract Classes must always contains common reusable methods.

3) IF the group of programmers are using same methods then such method is known as Common Method and those common methods must be taken as Abstract Method in an Abstract Class. (IF a method is used by only one programmer then such method is known as Specific Method.). All the common methods are recommended to write in abstract classes but not in concrete classes.

4) Abstract Classes should not be Final because they are always reusable.

5) Abstract Classes can be declared but they are not

ex. Op O1 = new Op(); ✗ // Invalid.
Op O1; // Valid Declaration.

An object astract class can not created directly but it can be created indirectly.

① An object of Abstract Class = Abn object of its Subclass

② An object of Abstract Class = An object of that class which extends that abstract class.

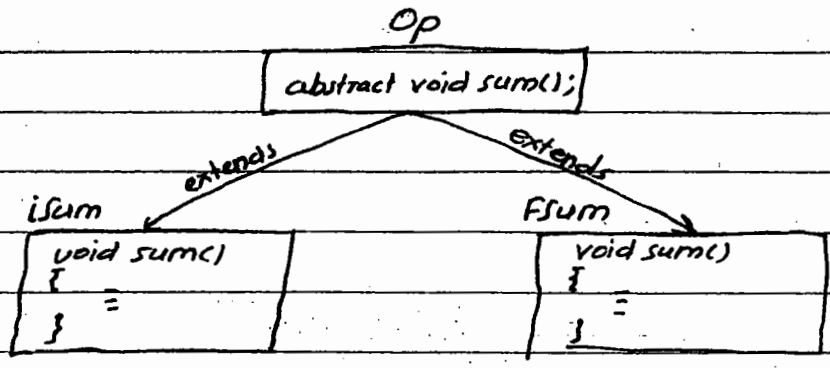
③ An object of subclass of an abstract class is nothing but an object of abstract class.

Prog # Write a java program which will compute sum of two integers and floats.

→

```
abstract class Op
{
    abstract void sum();
} // End of Op

class iSum extends Op
{
    int a,b,c;
    void sum()
    {
        a=10;
        b=20;
        c=a+b;
```

```
class FSum extends Op
{
    float a,b,c;
    void sum()
    {
        a = 10.25f;
        b = 20.75f;
        c = a+b;
        System.out.println("Float Sum = " + c);
    }
} // End of FSum

public class ACDemo
{
    public static void main(String args[])
    {
        //Op o = new Op(); Invalid Since 'op' is Abstract Class
        Op op1; // declaration
        op1 = new iSum();
        op1.sum();
        System.out.println("W.r.t. to iSum");

        op1 = new FSum();
        op1.sum();
        System.out.println("w.r.t to FSum");

    } // End of Sum

} // End of ACDemo.
```



Here, sum() method of iSum & FSum classes are specific.

Abstract Classes concept of Java is always using polymorphism alongwith method overriding for application development (Business logic) and dynamic binding for executing the polymorphic application (execution logic).

* Abstract Base Class & Abstract Derived Class *

An abstract base class is one which is always containing physical representation of abstract methods.

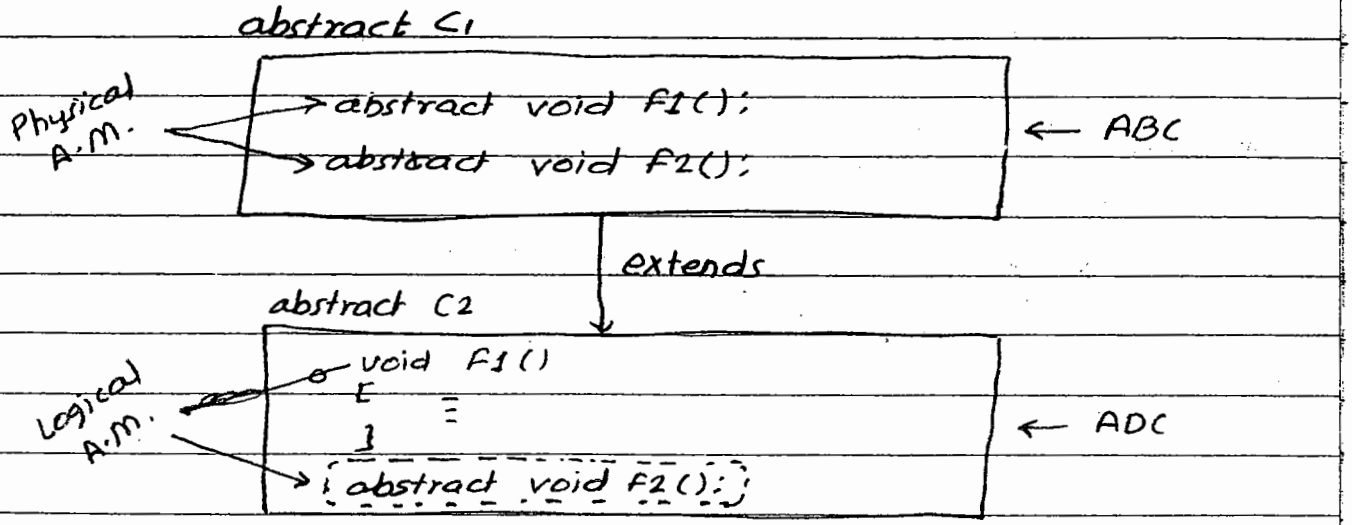
An abstract derived class is one which is containing logical representation of abstract methods which are inherited from abstract base class.

If the Derived class inherits 'n' no. of abstract methods from the abstract base class and if the derived class is not defining at least one abstract method. then the derived class is known as abstract derived class and whose definition must be made as abstract by using 'abstract' keyword.

If the derived class is defining all the 'n' number of abstract methods which are inherited from an abstract class then the derived class is known as concrete derived class.

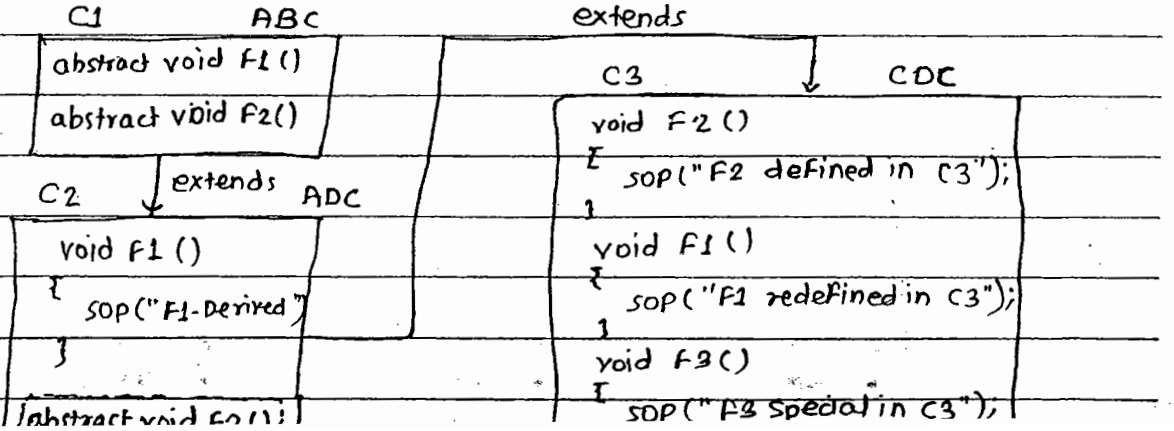
With respect to both ABC and ADC, one can not instantiate an object directly but we can instantiate indirectly. Or

Both ABC & ADC classes are reusable / inheri-table.



Date - 12-MAR-2011

Prog # Write a Java Program to implement the Following diagram



```
→ abs. class C1
  {
      abstract void F1();
      abstract void F2();
  } // End of abstract C1   (ABC)

  abstract class C2 extends C1
  {
      void F1()
      {
          System.out.println("F1 defined in C2");
      }
  } // End of C2      (ADC)

  class C3 extends C2
  {
      void f2()
      {
          system.out.println("F2 defined in C3");
      }
      void F1()
      {   super.F1(); // will call super class F1().
          System.out.println("F1 is redefined in C3");
      }
      void F3()
      {
          System.out.println("F3 is specilized method in C3");
      }
  } // End of C3      (CDC)
```

```
class ABDemo2
{
    public static void main (String args[])
    {
        System.out.println(" W.r.t C3 - Inheritance");
        C3 o3 = new C3();
        o3.F1();
        o3.F2();
        o3.F3();
        System.out.println (" w.r.t C2 - Dynamic Binding");
        C2 o2 = new C3(); // Indirect object creation.
        o2.F1();
        o2.F2();
        //o2.F3(); Invalid since F3() does not exists in C2 class.

        System.out.println ("w.r.t. C1 - Dynamic Binding");
        C1 o1 = new C3();
        o1.F1();
        o1.F2();
        //o1.F3(); Invalid since F3() does not exists in C3 class.
    } // End of main
} // End OF ABDemo2
```

With respect to ABC & ADC class, we can call those methods which are available in the same classes but objects of ABC & ADC classes does not contains the details about those methods which are specially defined in derived classes.

* Consequences of Abstract Classes -

1) If any classes containing set of different methods with no block of statements (Null body methods) then such classes are highly recommended to make as abstract. Because there is no use of calling those null body methods with respective that particular class.

In very short, the class which is containing null body methods is highly recommended to make as abstract & such class is known as Abstract Concrete Class.

Ex.

```
abstract Class Father
{
    void goingToCollege() //Null Body Method
    {
    }
}
```

2) If any class is containing set of defined methods with a block of statements which are of no use then there is no use of calling such method. Hence, it is highly recommended to make these type of concrete classes as @abstract.

Ex.

```
abstract class Op
{
    void sum()
    {
        System.out.println("I am from sum()
        without doing any sum");
    }
```

Q] Can we make a concrete class as abstract.

→ Yes,

See consequence no. 1 & no. 2

Q] Define Null Body Method.

→ A Null Body Method is one which is containing a definition / body with no block of statements.

Null Body methods are always present in abstract classes & they are always suppose to be overriden in the derived classes.

3) It is highly recommended to override abstract methods but not defined methods because overriding abstract methods takes only ~~abstr~~ construction cost and overriding of defined methods takes construction as well as destruction cost. Hence method overhead cost of abstract method is less. And method overhead cost of defined methods is more.

Date: 13-MAR-2011

* Interfaces *

Interfaces is one of the distinct programming concept in java for providing various advantages.

* Limitations of Abstract Classes -

- When we use abstract classes as a part of our application, there is possibility of getting following limitations.

① Abstract Classes never participle in multiple Inheritances.

ex. class c1 extends A1, A2 // Invalid
{
=
}

② The features of abstract classes are by default common reusable, but not by default universal common reusable. i.e. Abstract classes feature are not by default public.

③ We know that abstract class contains both defined & abstract methods. We know that overriding of the abstract methods are recommended but not recommended to overrided defined method. Whichever defined methods are not recommended to override those methods are present in abstrad classes. So that reusing the features of abstract classes into further classes are not recommended.

To eliminate the above limitations, we must use the concept of interfaces.

- Interface concept is used for developing User/Programmer

All the interface names in Java are treated as User/ programmer defined datatypes.

- To develop the concept of interfaces we use a keyword called 'interface'.

* Advantages of Interfaces over Abstract Classes -

When we use interfaces concept as a part our application, we get the following advantages.

i) All interfaces in Java participates in multiple Inheritance.

ii) All the features of interfaces are by default belongs to Universal common reusable features.

iii) Interfaces contains only collection of abstract methods but not defined methods.

* Syntax For Defining an Interface -

```
interface <intf_name>
{
    Variable declaration cum initialization;
    Methods Declaration;
}
```

* Explanation -

In the above syntax.

- 'Interface' is a keyword used for developing user/ programmer defined datatypes.
- <intf name> represents a java valid variable name. And it is treated as name of the interface.
- With respect to an interface one cannot instantiate an

- Variable declaration in the interface represents data members of an interface and we know that every data member of an interface is accessed universally. In general, all the univeral data members must have some fixed values. so that everybody can use without changing those values.
- Each & every data member or variable of an interface must be initialize otherwise we get compile time error.
- By default programmatically all the data members of an interface belongs to

  public static final xxx datamembers.

  Here, xxx represents data type, variable name & variable value.
- All the methods in the interface are belongs to universal reusable methods for performing universal common operations & those methods must be declared

We know that any method which is declared, such method are known as undefined methods. To make undefined methods as Abstract methods explicitely, we need not to use any keyword 'abstract' before the declaration.

To make the abstract method as universal abstract methods, we need not to use public keyword explicitely. Hence, by default all the methods of interfaces belong to

public abstract methods

* Definition of Interface -

1) An interface is a collections of public static final xxx datamembers and public abstract methods.

reusable datamembers and universal common reusable methods.

Prog # Write an interface by taking universal common reusable data members a, b with 10 & 20 and take the suitable methods.

→

```
interface i1
{
    int a=10; // public static final int a=;
    int b=20; // public static final int b;
    void f1(); // public abstract void f1();
    void f2(); // public abstract void f2();
}
```

Compile above program.

cmd> javac i1.java

Ensure that i1.class file must be generated.

In order to see the profile of i1 interface use the following.

cmd> javap i1 ↵

```
interface i1 {
    public static final int a;
    public static final int b;
    public abstract void f1();
    public abstract void f2();
}
```

* Points to be remember -

1) All the interfaces & their features are always uni-

2) The definitation of the interfaces should not be final. because interfaces are reusable always.

3) Interfaces definition is by default abstract i.e. it is optional for the java programmer to write abstract keyword before the interface method & the interface definition.

4) Constructors are not permitted in the interfaces because of the following reasons.
   (a) The data members of interfaces are alread initialized.
   (b) We know that constructors are defined special methods & such defined methods are not permitted in Interfaces.

5) The definition of an interface should not contain any main method because all the main methods in java must be defined. And the defined methods are not permitted in Interfaces.

6) For each & every class in java, there exist an implicit predefined super class called java.lang.Object. But each & every interface, there are no super classes.

Date - 14-MAR-2011

* Inheriting the features of Interfaces -

In order to inherit the features of interfaces to the classes, we have three approaches or syntaxes.

+ Approach 1/Syntax-1 (Inheriting the features of interface

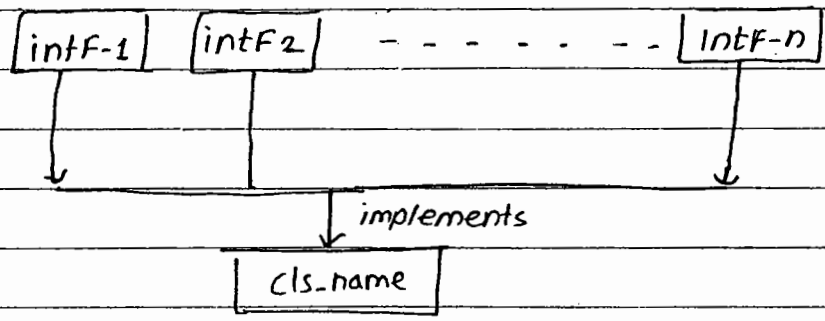
[abstract] class <cls_name> implements <intF-1>, <intF-2> .... <intF-n>

{

variable declaration;

Method definition /declaration;

}



* Explanation -

In the above syntax -

→ <clsname> represents name of the derived class.

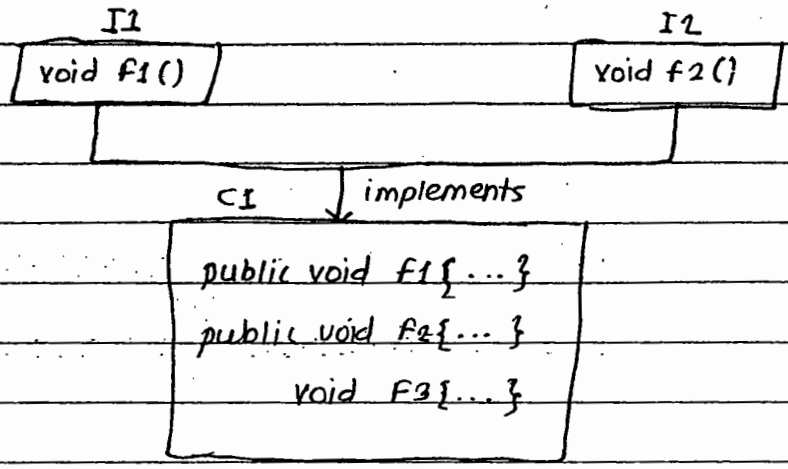
→ <intF-1> .... <intF-n> represents list of base interfaces.

→ implements - It is keyword which is used for inheri-ting the features of interface(s) to the derived class. It is improving the functionality of derived class.

In Java programming one ~~can~~ class can extends only one base class whereas one ~~can~~ class can implements either one or multiple interfaces.

If the derived class is inheriting 'n' number of abstract methods from multiple interface(s) & if the derived class is not defining at least one abstract method then the current derived class is known as Abstract and whoose definition must be made as abstract by using 'abstract' keyword.

Write a Java program to implement the following diagram.




```
interface i1
{
    void f1(); // public abstract void f1();
}

interface i2
{
    void f2(); // public abstract void f2();
}

class c1 implements i1,i2
{
    public static void main(String args[])
    {
        Sop(" W.r.t.
    public void f1()
    {
        sop("f1-defined - c1");
    }
    public void f2()
```

```
    void F3()
    {
        sop("F3 - Special Method - C1");
    }
}

public class InDemo
{
    public static void main(String args[])
    {
        sop("w.r.t C1 - Inheritance");
        C1 o1 = new C1();
        o1.F1();
        o1.F2();
        o1.F3();
        sop("w.r.t i1 - Dynamic Binding");
        // i1 i = new i1(); Invalid since it is abstract.
        i1 io1 = new C1();
        io1.F1();
        io1.F2();
        io1.F3();   // Invalid because they does not exist in i1

        sop("w.r.t i2 - Dynamic Binding");
        i2 io2 = new C1();
        io2.F1(); // Invalid since F1() does not exist in i1.
        io2.F2();
        io2.F3(); // Invalid since F3() does not exist in i2.
    } // End of main
} // End of InDemo.
```

An object of an interface cannot be created directly

An object of an Interface = An object of its subclass

or

An object of an interface = An object of that class which implements that interface.

or

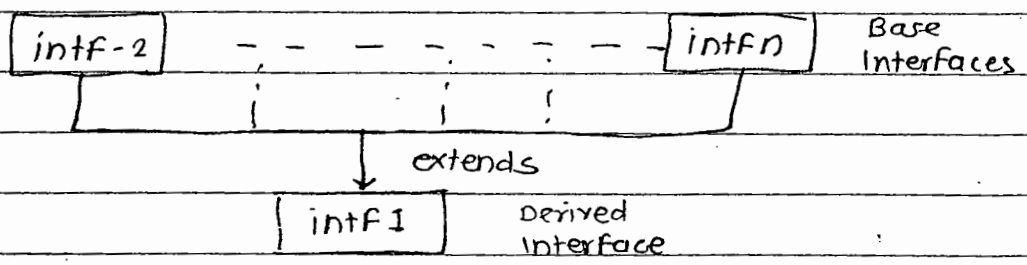
An object subclass of interface is nothing but an object of an interface.

Interfaces concept of Java makes use of polymorphism alongwith method overriding (Business logic) for developing application and dynamic binding for execution of the application (execution logic).

* Approach 2 / Syntax-2 - (Interface Inheritance)

The process of inheriting the features of interface(s) to another interface is known as Interface Inheritance.

```
interface <intf-1> extends <intf-2> ------<intfn>
{
    variable declaration cum initialization;
    method declaration;
}
```

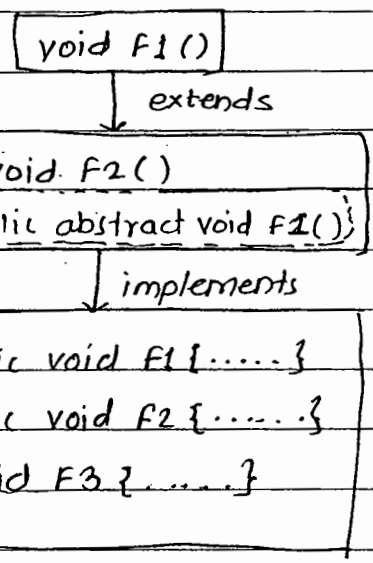


In the above syntax -

- <intf-1> represents name of the derived interface.

- 'extends' is a keyword which is used for inheriting the features from interfaces to the another interface + it is improving the functionality of derived interface.
- One class can extends only one class whereas one interface can extends either one or more than one interface because java does not support multiple inheritance through the concept of classes but it can be supported by the concept of interface(s).

Write a Java Program which implements the following diagram.



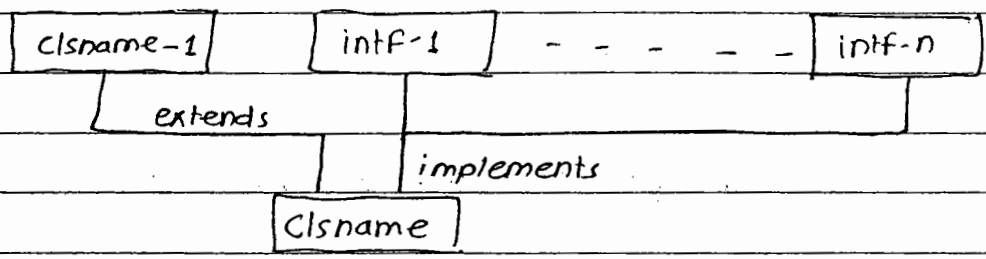

```
interface i1
{
    void f1(); // public abstract void f1();
}

interface i2 extends i1
{
    void f2(); // public abstract void f2();
}
```

```
class C1 implement i2
{
    public void F1()
    {
        SOP(" F1 - defined in C1");
    }
    public void F2()
    {
        SOP(" F2 - defined in C1 ");
    }
    Void F3()
    {
        SOP ("F3 - Special Method ");
    }
}

public class InDemo2
{
    public static void main(String args[])
    {
        SOP(" W.r.t. C1 - Inheritance ");
        C1 co1 = new C1();
        co1. F1();
        co1. F2();
        co1. F3();
        SOP(" W.r.t i1 - Dynamic Binding ");
        i1 io1 = new C1();
        io1. F1();
        SOP("w.r.t i2 - Dynamic Binding ");
        i2 io2 = new C1();
        io2. F2();
```

An object of an interface contains the details about those methods which are available in that interface only but it does not contain details about those methods which are available into some other interfaces and the special methods which are defined in some other derived class.

* Approach 3 / Syntax-3 - (Derived class is inheriting the features of both from base class & interface(s))

[abstract] class <cls.name> extends <cls.name 1>
implements <intF-1> - - - <intF-n>
{
variable declaration;
method Definition or Declaration
}

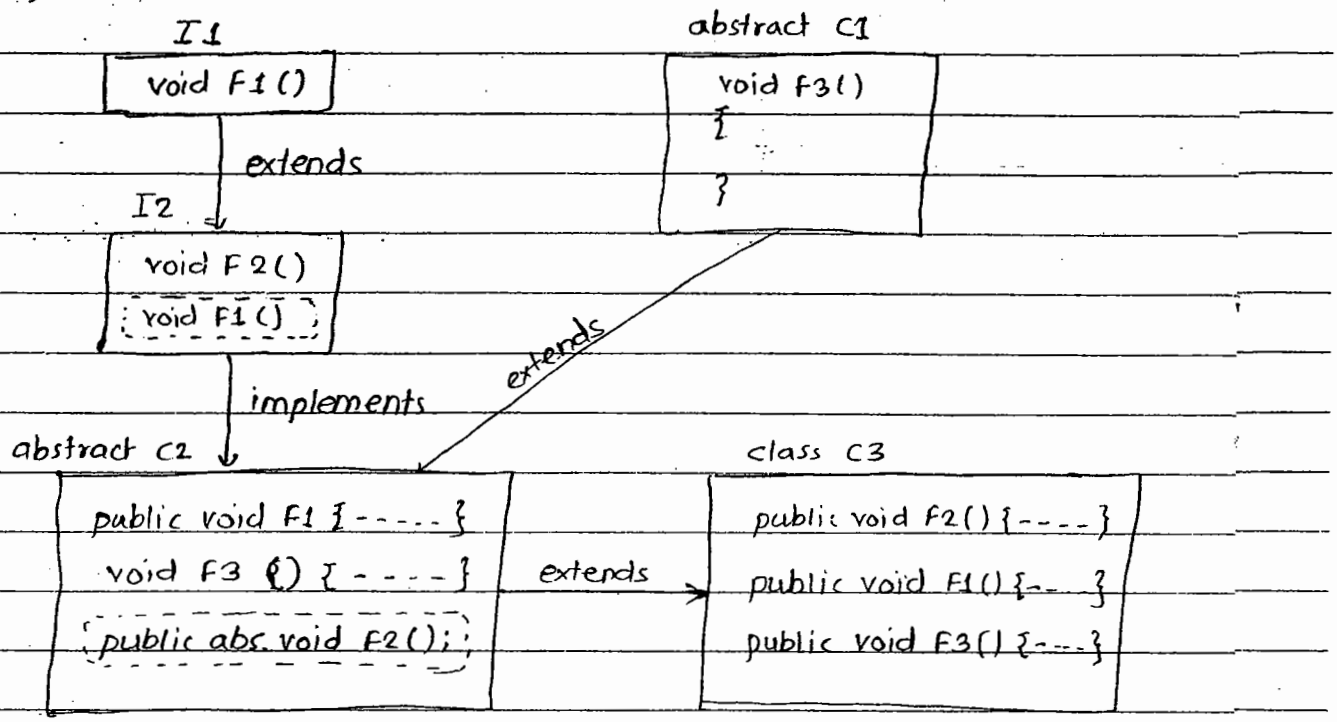


In the above syntax -
- <clsname 1> represents the name of base class.
- <intF-1> - - - <intF n> represents name of the base interface(s)
- < Clsname> represents the name of derived class

Whenever we use both extends & implements keywords in a single syntax, it is mandatory for the java programmer to use extends keyword first and later

Date: 15-MAR-2011

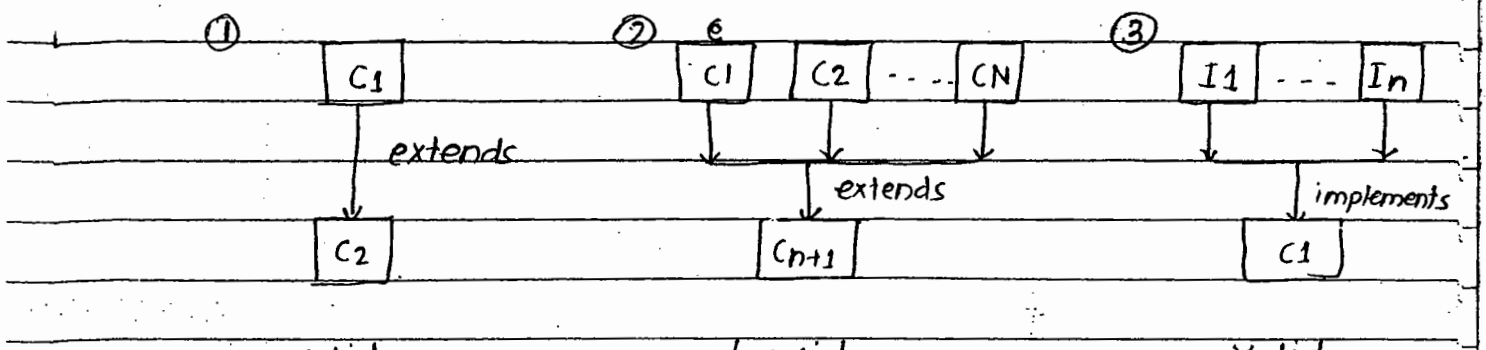
Prog # Write a Java Program which will implement the following diagram.

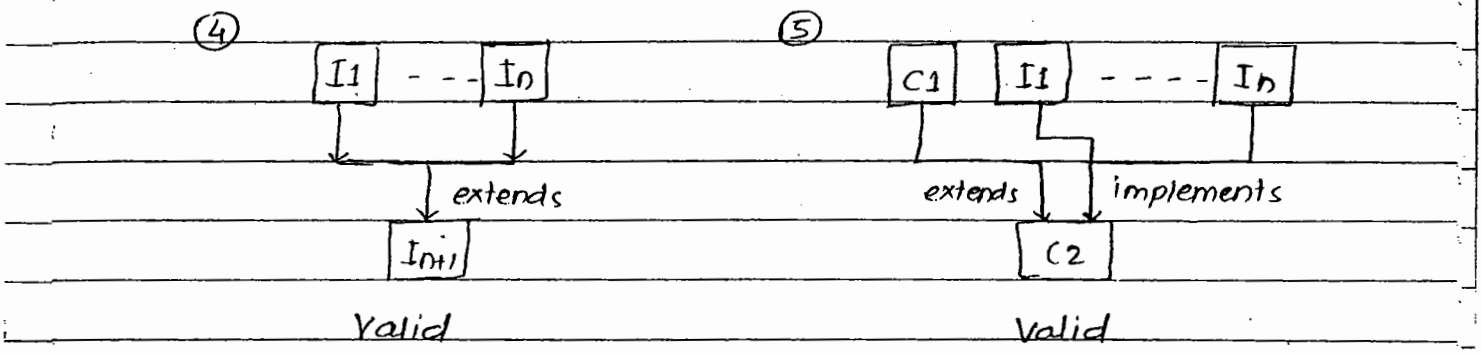


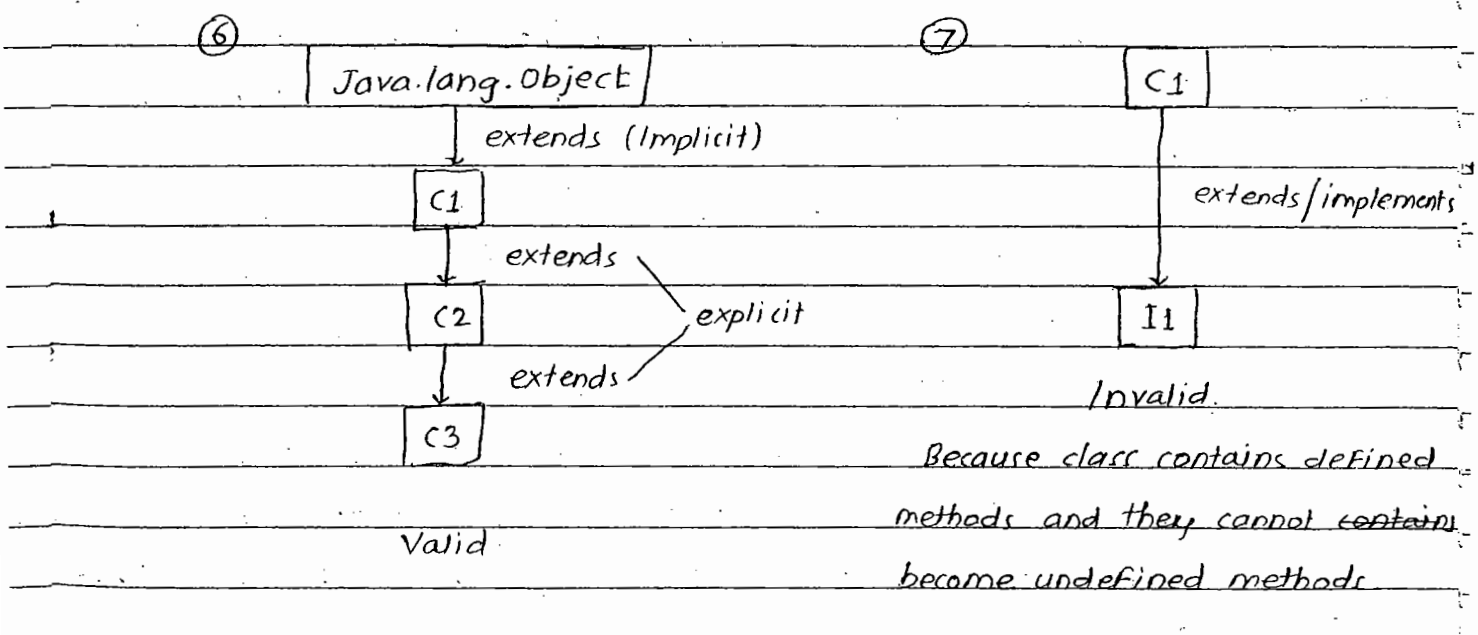

```
interface i1
{
    void F1(); // public abstract void F1();
}

interface i2 extends i1    // Interface Inheritance
{
    void F2(); // public abstract void F2();
}

abstract class C1
{
    void F3()
    {
    }
```

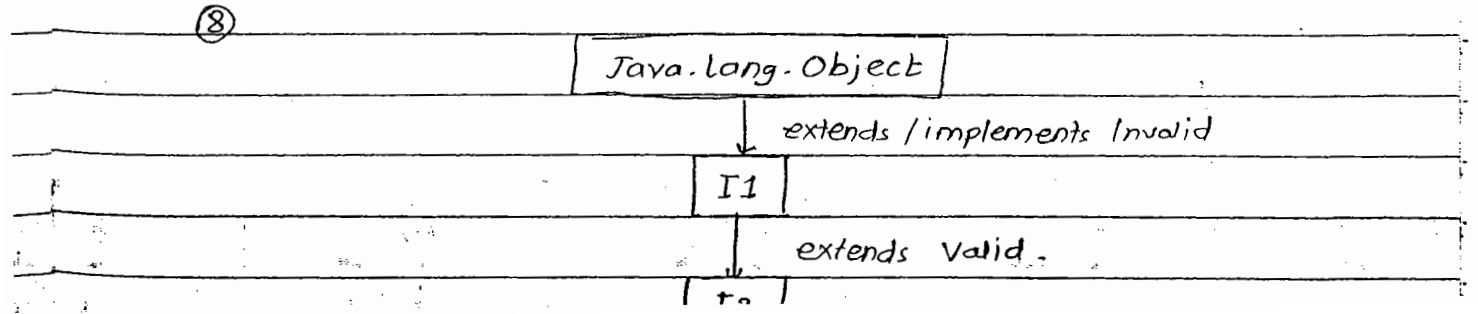
```
abstract class c2 extend c1 implements i2
{
    public void F1 ()
    {
        System.out.println("F1 - defined in c2");
    }
    void F3()
    {
        System.out.println("F3 - redefined in c2");
    }
} //End of C2

class C3 extends C2
{
    public void F2()
    {
        System.out.println("F2 - defined in C3");
    }
    pubic void F1()
    {
        System.out.println("F1 - Redefined in C3");
    }
    public void F3()
    {
        System.out.println("F3 - Re-redefined in C3");
    }
}

public class InDemo3
{
    public static void main (String args[])
```

```
System.out.println(" w.r.t C3 - Inheritance");
C3 o3 = new C3();
o3.F1();
o3.F2();
o3.F3();
System.
System.out.println(" w.r.t. C2 - Dynamic Binding");
// C2 o2 = new C2(); Invalid because C2 is abstract
C2 o2 = new C3();
o2.F1();
o2.F2();
o2.F3();

System.out.println("w.r.t. C1 - Dynamic Binding");
// C1 o1 = new C1(); Invalid because C1 is abstract
C1 o1 = new C3();
//o1.F1();
//o1.F2();   } Invalid because doennot exist in C1
o1.F3();

System.out.println("w.r.t I2 - Dynamic Binding");
// I2 io2 = new I2();
I2 io2 = new C3();
io2.F1();
io2.F2();
// io2.F3();

System.out.println("w.r.t I1 - Dynamic Binding");
// I1 io1 = new I1();
I1 io1 = new C3();
io1.F1();
```


①
C1
extends
C2
Valid
②
C1
C2
- - -
CN
extends
Cn+1
Invalid
③
I1
- - -
In
implements
C1
Valid


④
I1
- - -
In
extends
In+1
Valid
⑤
C1
I1
- - - -
In
extends
implements
C2
Valid


⑥
Java.lang.Object
extends (Implicit)
C1
extends
C2
extends
explicit
C3
Valid
⑦
C1
extends/implements
I1
Invalid
Because class contains defined methods and they cannot ~~contain~~ become undefined methods


⑧
Java.lang.Object
extends/implements Invalid
I1
extends Valid.

~~You are~~

9) Give or write the profile for following.

```
Class C1 implements I1
{
    =
}
```

Profile:-

```
cmd> javap C1
   > class C1 extends java.lang.Object implements I1
     {
        =
     }
```

* Note -

When you are overriding abstract methods of interfaces, in the context of derived class, we must write public access specifier before the abstract methods of interfaces. Other-wise we get a compile time error. (weaker access Privilege).

* Data Conversion *

In order to convert a Fundamental value into the String data, we must use the following predefined method which is present in a predefined class called String Class

String

public static Final String valueOf (byte)

—— " —— (int)

(char)

In general, valueOF(-) method can be expressed as

```
public static Final String valueOF (xxx);
```

Here, xxx represents any Fundamental datatype value.

Example -

① byte b = 20;
String s = String.valueOF (b);

② int n = 40000;
String s1 = String.valueOF (n);

* Possible Conversions of Data -

① String Data ⟶ Fundamental Data
② Fundamental Data ⟶ String Data.
③ Strind Data ⟶ Wrapper Class Object
④ Wrapper Class Object ⟶ strind Data.
⑤ Fandamental Data ⟶ wrapper Class Object
⑥ Wrapper Class Object ⟶ Fundamental Data.

Date: 16-MAR-2011

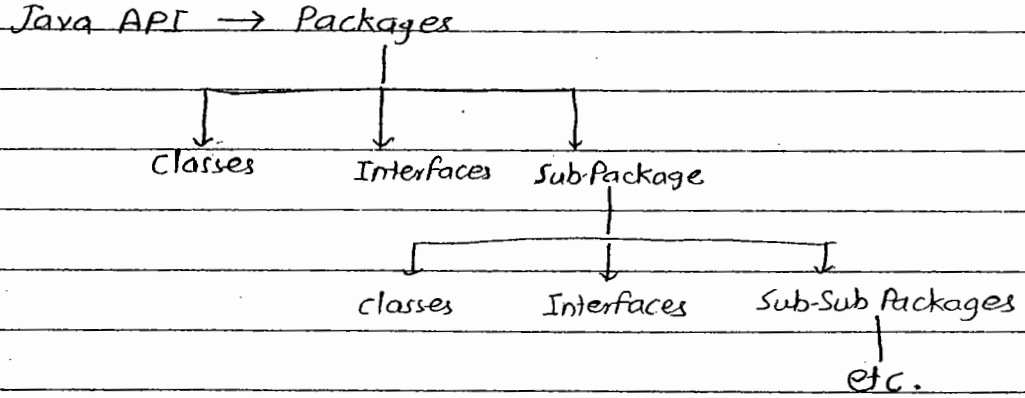
* Packages in Java *

Learning about C language is nothing but learning about its concepts + Library. We know that Library of C language is a collection of pre-defined header files. and a header file is a collection of pre-defined functions.

Similarly, Learning about Java is nothing but the Learning about concepts + OOPs features + API. We know that API of java is a collection of ~~Classes~~, packages.

* Definition of a Package -

A package is a collection of Classes, Interfaces and Sub-packages. A sub-package in turns contains collection of Classes, Interfaces and sub-subpackages etc.



If we any classes or interfaces are common for many no. of Java programmers then those classes & interfaces are recommended to place in a package.

In other words, all the common classes & common interfaces must placed in the package for getting more advantages

* Advantages of Packages -

When we use the concept of package as a part of our Java program, we get the following advt.

① Application Development time is very less.

② Memory Space of an application is very less

③ The performance of an application is increased.

④ Redundancy (Repeatition) of the code is reduced, so that we get consistent result & less memory cost

⑤ Execution of time of the application is very less

⑥ An overall performance of the project is improved

Hence, with the concept of packages one can acheive the slogan of Java i.e. WORA. (Write-Once-Reuse-Anywhere)

Q] What is the difference between inheritance & packages?

→ The concept of inheritance makes us to understand data sharing can be achieved within the program between class to class, interfaces to interface and interface to class but not accross the program.

The concept of packages makes us to understand data sharing can be possible within the program and accross the programs between Class to Class, Interface(s) to interface & interface(s) to class.

* Types of Packages in Java -

In Java programming packages are classified into two types. They are

1) Pre-defined / Built-in Packages
2) User / Programmer / Secondary / Custom defined Packages

1) Pre-defined Package -

Pre-defined packages are those which are developed by sun Microsystems and supplied as a part of Jdk to deal with Universal Requirements

2) User-defined Packages -

User-defined packages are those which are developed by Java Programmer & supplied as a part of their project to deal with common specific requirements

* Type of Pre-defined Packages -

According to industry standards pre-defined packages of Java are classified into three types. They are

1) J2SE / Core Packages
2) J2EE / Advanced Packages
3) Third Party Packages

J2SE / core packages are those which are used in the industry for development of client-side applications.

All the four core packages in java starts with -

`java.<packname>`

J2EE packages are those which are used in the industry for developing serve-side applications. All the J2EE packages starts with

`javax.<packname>`

ex. javax.sql.*;
javax.servlet.*;
javax.servlet.http.*;

Third party packages are those developed by third party software vendors (Oracle, IBM, Hibernate) supplied some packages for strengthening and providing support for J2SE & J2EE packages for effective application development.

ex. oracle.jdbc.driver.OracleDriver;
ibm.jdbc.driver.DB2Driver;
satya.jdbc.driver.SatyaDriver;

Third party packages never starts with java and javax. and most of third party packages starts with their company name.

Date: 17-MAR-2011

* List of J2SE Pre-defined Packages -

All the predefined packages of J2SE are compressed in a single file called rt.jar. (runtime.JavaARchive). The location of rt.jar file is Jdk1.5/jre/lib folder.

In J2SE, we have 8 predefined essential packages. They are -

~~i) java.lang.*;~~

1) java.lang.* -

obtaining language functionalitie or the basic services of the language.

Some of the language functionalities are.

a) Displaying the data on the console.

b) Accepting command line arguments.

c) Performing data conversions

d) Obtaining garbage collector.

e) Development of Multi Threading Applications etc.

This is the package which is by default import into every Java Program. Hence, it is known as default package.

2) java.awt.* - (Abstract Windowing Toolkit)

This is one of the predefined package which is used for developing GUI applications / Front-end Appl^n.

In order to develop any GUI application we require various GUI components / widgets and all GUI components are given in the form of predefined classes such as Button, Label, Textfield, TextArea etc.

In other word, java.awt.* package allows us to create the components.

Creating any GUI component is nothing but creating an object appropriate predefined class.

ex. Create a button with a name "Save"

```
Button b1 = new Button("save");
```

3) java.awt.event.* -

Event is the subpackage of awt package. The purpose of this package is to provide the functionality / life / behaviour to the GUI components.

In order to develop a complete GUI application we require to import. java.awt.* (Component Creation)

* Definition of Event -

The change in the state of the object is known as Event

example: Button is Clicked.
Checkbox is checked
Scrollbar is adjusted vertically or Horizontally

4) java.io.* -

This is one of the pre-defined package contains set of classess & interfaces which are used for perf-orming file handling operations. Some of the file handling operations are.

a) Opening a file either in read mode or write mode
b) Inserting a record into the file.
c) Deleting a record from the file.
d) Modifying a record in a file.

5) java.applet.* -

This is one of the predefined package developed by sun microsystem for building distributed applications. In the earlier days, of Sun MicroSystem (1990-1995) there was a concept called applets whose basic aim is to develop distributed applications.

The concept of applets developed by Sun Micro System & released to the industry on the name of a predefined class Applet. And they kept it into a pre defined package java.applet.*. This package contains only one class called Applet & whose Fully qualified name java.applet.Applet.

If we write any java program with the concept of applets then the java program is known as Applet

* Definition of an Applet -

An Applet is a Java program which runs in the context of browser and whose results are sharable accross the Globe.

* Note -

Because the drawbacks of applets, industry programmers are not using ~~for~~ applet for development of distributed application. ~~for~~ Servlet & JSP are the current technologies used for developing distributed applications in the industry.

Date: 18-MAR-2011

6) java.net.* - (Networking)

This is one of the pre-defined package used for developing networking or Client-Server applications. We know that client-server application contains two types of programs. They are -

i) Client Program

ii) Server Program

A Client program is one which always makes a request to get the services from server. In order to develop client programs, we use the pre-defined classes and an interfaces. which are present in java.net.*.

A Server program is one which always receives client request, process that client request & gives response back to multiple clients. To develop the server side program we use set of predefined classes & interfaces which are present in java.net.*.

With java.net.* package, we are able to develop only intranet applications for small scale organisations & unable to develop internet application for large scale organisations. (To develop internet applications, we make

7) java.util.* - (Collection Framework) -

Java.util.* package is used for improving the performance of Java & J2EE applications. In other words, this package contains Collection of classes & interfaces which are used as a part of our Java application for strengthening the performance of Java & J2EE applications.

Java.util.* package is also known as collection Framework. Collection Framework is an additional service or add-on service to the library of Java for strengthening the java projects.

* Definition of Collection Framework -

Collection Framework is the standardized mechanism of Java which allows use to integrate or group or gather the multiple values either they belongs to same type or different type or both to a single variable. This single variable is known as Collection Framework variable.

Q] What are the differences between arrays & collection Framework ?

→

Array	Collection Framework
i) An array is a collective name given to a group of memory locations which are all refered by similar type of elements.	i) Collection Framework is a standardised mechanism of java which allows us to group both similar & different types of elements or both in a single variable.
ii) The concept of arrays allows only homogeneous elements.	ii) The concept of collection Framework allows both homo-geneous & heterogeneous ele-

ii) Concept of arrays is having fixed size in Nature.

iii) Concept of Collection Framework is having dynamic in nature.

8) java.text.* -

This package contains collection of pre-defined classes & Interfaces which are used for formating dates, time, numerical values etc. at the time of generating reports in the organisation.

Generating reports is ~~kno~~process is also known as Text Processing.

Date: 19-MAR-2011

* User/Programmer/Custom Defined Packages *

User Defined Packages are those which are developed by Java programmer & supplied as a part of their project to deal with common specific requirements.

If we develop any classes & interfaces for group of java programmers for common accessibility, then these classes & interfaces are recommended to place in a package. In other words, common classes and interface, we always keep in the packages.

Creating a user-defined package is nothing but creating the package name as directory or Current ~~working machine~~ Folder within current working machine.

* Syntax for Creation of Package-

```
package pack1[.pack2[.pack3 . . . . [packN]]];
```

* Explanation-

i) Package is a keyword used for developing User-defined Packages.

ii) pack1., pack2 ... packN represents valid variable names of java treated as User-defined package names.

iii) pack2, pack3 - - - packN represents sub-packages & whose specification is optional.

iv) pack1 represents an outer package & whose specification is mandatory.

Example:

```
package P1 ;
package p1.p2 ;
```
} Package Statement

* Rule -

Whenever we keep any class or any interface within the package, the package statement keyword always used as a first executable statement.

* Steps or Guidelines for development User-defined Packages.

In order to develop any User defined packages as a part of our project, the Java programmer must Follow the following guidelines.

i) Choose the appropriate package name alongwith package keyword & ensure that it should be first executable statement.

ensure whoose modifier must be public.

iii) The modifier of the constructor of the class must be public. This rule is not the case with Interfaces.

iv) The modifier of the methods of class / Interface must be public. (This rule is optional in the case interfaces)

v) Whichever class / Interface name is placing in a package, that class name / interface name must be given as a File name with an extension .java.

vi) At any point of time, either we must write a class definition or interface definition in a package but not both the definitions at a time.

* Create a package tp & place a class Test and choose the suitable method & constructors.

```
// Test.java  - - - - - (5)
public class Test  // - - - (4)
package tp;  // - - - - (1)

public class Test  // - - - (2)
{
    public Test()  // - - - - (3)
    {
        System.out.println("Test : DC");
    }
    public void disp()
```

```
        System.out.println("Test : Disp"); // ---④
    }
} // Main
} // Test
```

* Create a package tp & choose a place an interface ITest and choose the suitable methods.

→
```
// ITest.java  // ---⑤
package tp;  // ----①

public interface ITest  // ---②
{
    public void show(); // ----④
}
```

* Syntax for Compiling Package classes & Interfaces -

```
javac -d . <filename>.java
```

i) -d . In the above syntax -

i) -d represents an option or switch which gives an indication to the JVM saying that goto Filename.java, take the package name create it as directory or folder provided.

ⓐ No errors present in <Filename>.java.

ⓑ Package name should not be created as directory earlier.

The create directory is known as Current directory.

ii) Automatically <Filename>.java will be compiled and we must ensure Filename.class file must be generated

iii) . refers current directory and it is used for implicitely copying currently generated .class file (filename.class) into the current directory which was created in step-I (.class files are copied in the package.)

* Example -

① `javac -d . Test.java ↵`

tp
→ Test.class

② `javac -d . Itest.java ↵`

tp
→ Test.class
→ Itest.class.

* Write a Java program which makes use of tp.Test class.

→
```
// packDemo.java
// import tp.Test;

class PackDemo
{
    public static void main(String args[])
    {
        tp.Test t = new tp.test(); // Fully Qualified Name
        t.disp();                  // Approach.
    } // Main.
} // End of PackDemo
```

Date: 21-MAR-2011

* Number of ways to refer to the package classes & interfaces -

In order to refer all the classes & interfaces of the specific package, in java programming we have two approaches. They are -

i) import statement
ii) Full qualified name approach -

1) By using import Statement -

Import statement is used for refering either all the classes & interfaces of specific package or a specific class / interface of a specific package in our current Java program.

Import statement is having two syntaxes.

a) Syntax - I -

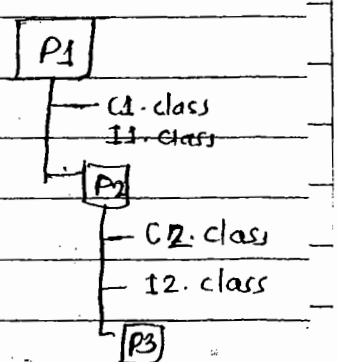
```
import pack1[.pack2 ----]. * ;
```

By using the above syntax one can refer all the classes & interfaces of a specific package.

Import is a keyword used for the refering classes and interfaces Pack, Pack2 .... PackN represents list of packages '*' is one of the wild character which gives an indication to the JVM saying that bring all the classes and interfaces a specific package to the current Java program.

Example -

```
import p1.*;        ①
import p1.p2.*;     ②
import p1.p2.p3.*;  ③
```



Statement ① always refers all the classes & interfaces of package P1 only but not their subpackages classes or interfaces.

Statement ② represents the classes & interfaces of package p2 but not its subpackage and its super packages details.

Statement ③

b) Syntax - II -

import pack[.pack2 ... [.pack2]]].<classname>/*/intfname>;

This syntax refers either one class or one interface of a specific package but not all.

Ex:

import p1.C1; ①
import p1.p2;I2; ②

Statement ① refers only class C1 of package P1 but not all the classes or interfaces of P1.

Statement ② refers interface I2 of package P2 but not all the classes & interfaces of package P2.

2) Fully Qualified name approach -

This is one of the alternative technique for import statement for refering classes & interfaces of a specific package.

Using fully qualified name approach a Java programmer can refer either one class or one interface at a time of a specific package but not all like import statement.

Ex:

```
P1.C1 o1 = new P1.C1();          ①
P1.P2.I2 io2 = new p1.p2.p3.C3(); ②
```

In statement the interface I2 of package P2 of P1 is implemented by class C3 of pack3 of pack2 of pack1.

* Write a java program which illustrate the concept of package which makes use of ITest interface of tp package.

→

```
// PackageDemo.java
// import tp.Test;

Class C1 implement tp.Test
{
    public void show()
    {
        System.out.println("Show Defined - in C1");
    }
} // C1

class PackageDemo
{
    public static void main(String args[])
    {
        System.out.println("w.r.t. Inheritance");
        C1 o1 = new C1();
        o1.show();
        System.out.println("w.r.t. Dynamic Binding");
        tpTest tp.ITest o2 = new C1();
        o2.show();
    } // main
```

* Skeletop - 1 - Examples -

This skeletop makes us to understand creation of an independent package & giving to multiple java programmers. The packages used in skeleton one are isolated packages.

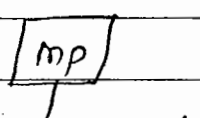
* write a java program to create a package mp and place a class called multiplication & take the suitable methods for generating multiplication table for a given number.

→ //Mul.java

```
package mp;
public class mul
{
    int n;
    public void set(int n)
    {
        this.n = n;
    }
    public void table()
    {
        for(int i=1 ; i<=10; i++)
            System.out.println(n+" * "+i+" = "+(i*n));
    }
} // End of mul.
```

compile the above program as -

cmd> javac -d . mul.java ↵



```
// Main Program -
// PackDemo
// This program makes use of mp.Mul to generate multi-
// plication table
import mp.Mul;
class PackDemo
{
    public static void main(String args[])
    {
        int n = Integer.parseInt(args[0]);
        Mul m = new Mul();
        m.set(n);
        m.table();
    } // Main
} // PackDemo
```

Note: If any class is containing a main method then such classes are not recommend to place in a package because main methods are defined by various pro-grammers for their specific purpose but not for common hence the class contains main method is not sharable.

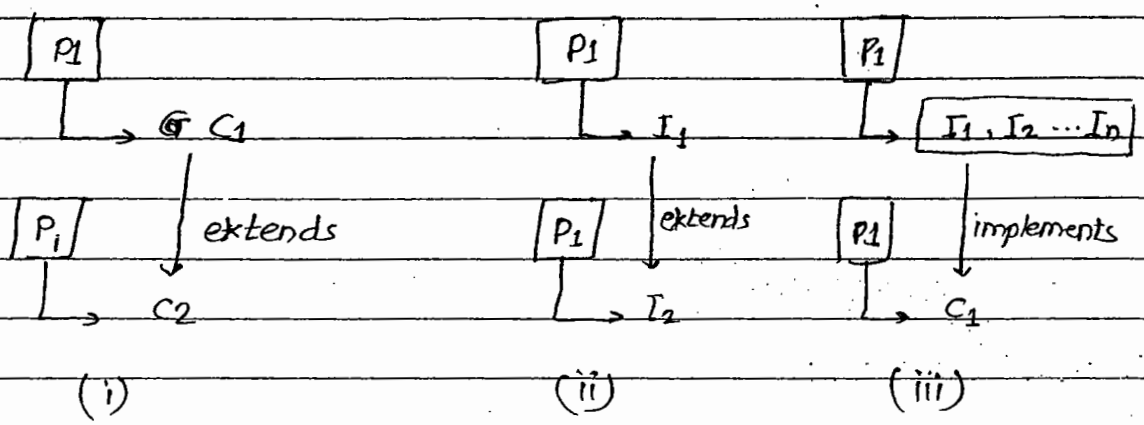
If any class or interface is not containing main method then those class & interfaces are highly recommended to place in a package.

Date: 22-MAR-2011

* Skeleton - 2 Examples -

(Accross the program & within the package)

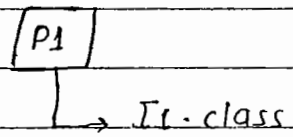
Let us consider the following diagram.



* Implement Figure (iii) by writing the Java program

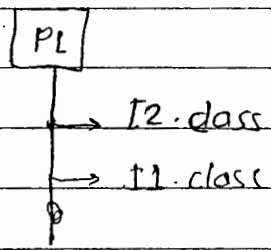
→ // I1.java

```
package P1;
public interface I1
{
public void F1();
}
```



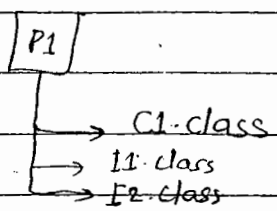
// I2.java

```
package P1;
public interface I2
{
    void F2();
}
```



// C1.java

```
package P1;
public class C1 implements I1, I2
{
    public void F1()
    {
        System.out.println("f1 - defined in C1");
```



```
    public void F2()
    {
        System.out.println("F2() defined in C1");
    }
} // C1
```

⇒ Compile the above program.

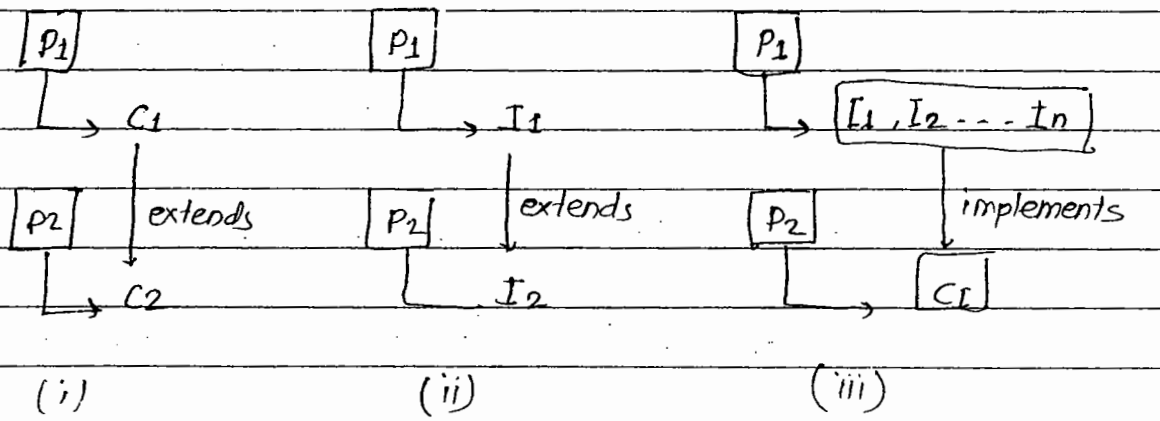
cmd > javac -d . C1.java.

```
// PackDemo.java.
// This program makes use of p1.C1, p1.I1 & p1.I2
class PackDemo
{
    public static void main(String args[])
    {
        System.out.println("w.r.t C1 - Inheritance");
        P1.C1 o1 = new P1.C1();
        o1.F1();
        o1.F2();

        System.out.println("w.r.t C I1 - DB");
        P1.I1 io1 = new P1.C1();
        io1.F1();
        // io1.F2(); // Invalid

        System.out.println("w.r.t. I2 - DB");
        P1.I2 io2 = new P1.C1();
        // io2.F1(); Invalid.
        io2.F2();
    } // main
} // PackDemo
```

If any class or interface is extending or implementing the base interfaces or base classes & all of them belongs to the same package then base classes or base interfaces need not to be qualified by the package name.

* Skeleton - 3 Example -

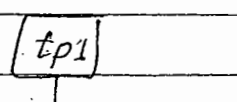
(Accross the programs & accross the packages)



* Note: If a derived class or derived interface is extending or implementing the base interfaces & base classes & if they are belongs to different packages, then the base classes & interfaces must be qualified by the package name, & otherwise it is a compile time error.

* Implement figure (iii) using Java program.

```
// I1.java
public package tp1;
  public interface I1
  {
     void F1()
  }
```



Compile the above java program as

```
// I2.java
package tp2;
public interface I2
{
    public void F2();
}
```

Compile the above java program as
cmd> javac -d . I2.java (Using tp.I2)

```
// C1.java
package tp3;
public public class C1 implements tp1.i1 , tp2.i2
{
    public void F1()
    {
        SO.P("F1 - defined - C1");
    }
    public void F2()
    {
        SOP("F2 - defined - C1");
    }
} //C1
```

Compile the above program as
javac -d .

```
// PackDemo.java
import tp1.I1;
import tp2.I2;
import tp3.I3;

class packDemo
```

```
public static void main(String args[])
{
    SOP("w.r.t C1 - Inheritance");
    C1 o1 = new C1();
    o1.F1();
    o1.F2();

    SOP("w.r.t I1 - DB");
    I1 io1 = new C1();
    io1.F1();
    // io1.F2(); Invalid

    SOP("w.r.t. I2 - DB");
    I2 io2 = new C1();
    // io2.F1(); Invalid.
    io2.F2();
} // main
} // PackDemo
```

* Access Specifiers in Java *

1) Access specifiers are the keywords in Java.
2) In java programming we have four types of access specifiers. They are.
    a) Private
    b) Default (Not a keyword)
    c) Protected
    d) Public
3) Access specified must be supplied before the data

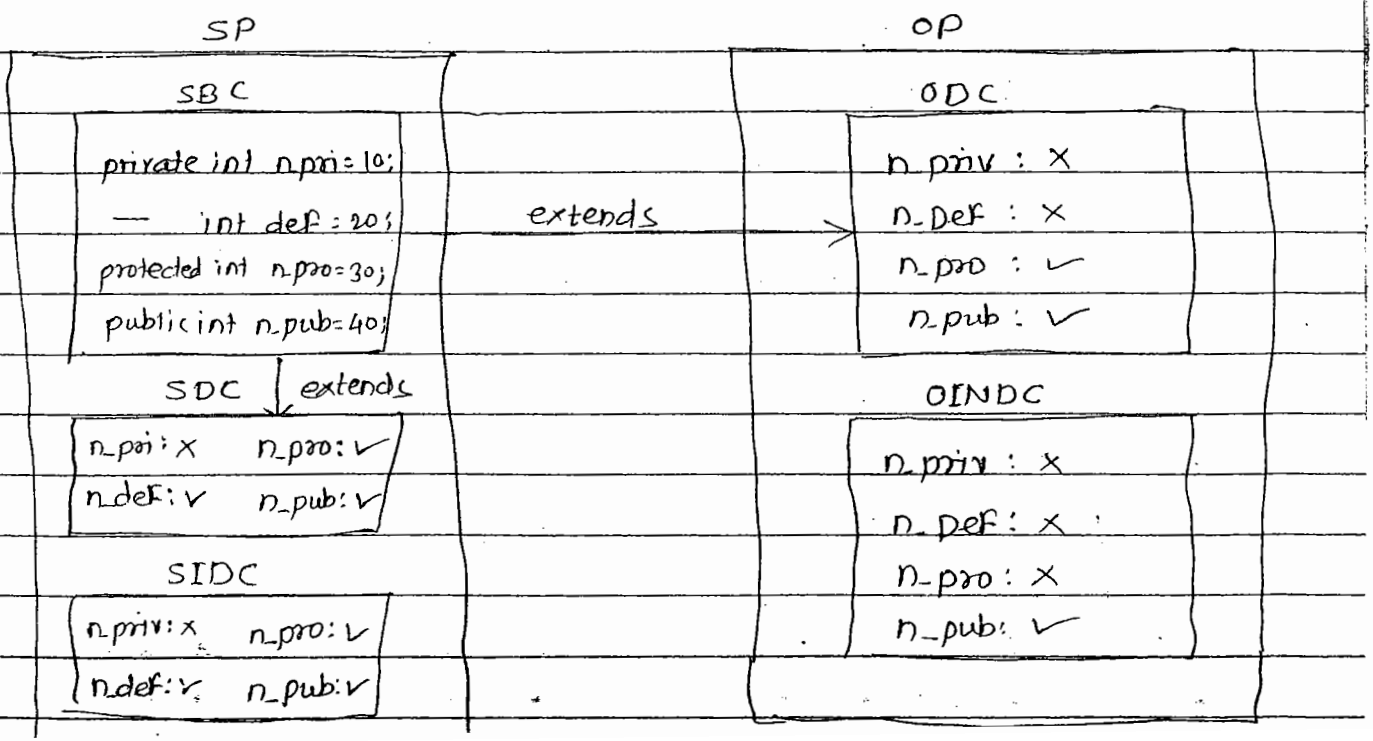
Definition: Access Specifier are those which allows us to access the data within the program, accross the programs within the package and accross the programs & accross the package between class to class, inteface(s) to interface and interface(s) to class.

In other words, access specifiers makes us to understand where to access the data & where not to access the data.

* If we are not using private, protected & public keywords before data members of the class then those data member of class are by default treated as default.

* Rules for the Access Specifiers -

The following diagram gives rules & regulations for access specifiers.



Access Protection Matrix

Type of Access Spec. \ Types of class	Same Package BC	Same Package DC	Same Packag SINDC	Other Package DC	Other Package INDC
Private	✓	×	×	×	×
Default	✓	✓	✓	×	×
Protected	✓	✓	✓	✓	×
Public	✓	✓	✓	✓	✓

✓ means Accessible or visible

× means Unaccessible or invisible

Date: 23-MAR-2011

Note: i) Private access specifier is also known as Native Access Specifier.

ii) Default access specifier is also known as Package access specifier.

iii) Protected access specifier is also known as Inherited access specifier.

iv) Public accers specifier is also known as Universal Access Specifier.

Q] What are the package Access Specifiers?

→ i) Private

ii) Default

iii) Protected

iv) Public

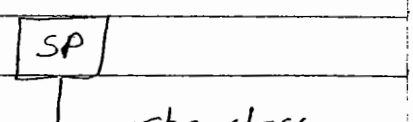
Q] Which access specifier is known as Package Access Specifier

→ Default.

* Write the Java Programs which illustrate the concept of Access Specifiers Rules.

→ // Sbc.java

```
package sp;
public class Sbc
{
    private int n_pri = 10;
            int n_def = 20;
    protected int n_pro = 30;
    public int n_pub = 40;

    public sbc()
    {
        S.O.P("Value of N_pri = " + n_pri);
        S.O.P("Value of N_def = " + n_def);
        S.O.P("Value of N_Pro = " + n_pro);
        S.O.P("Value of N_pub = " + n_pub);
    }
} // Sbc
```




```
// output
// sbc s1 = new sbc();
// 10 20 30 40
```

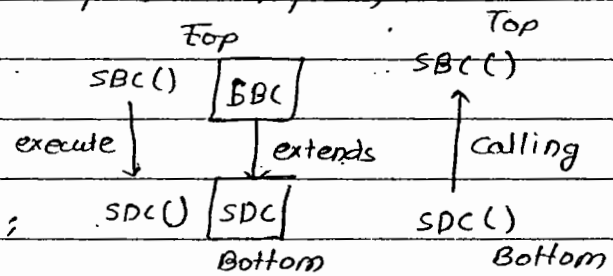
Compile the above program as

javac -d . Sbc.java.

Ensure that sbc.class must be generated.

```
// sdc.java
packag sp;
public class sdc extends sbc // Is-A Relation within the
```

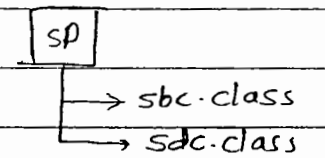
```
    public sdc()
    {
        // S.O.P ("Value of Pri = " + n_pri) ; // Not accessible
        S.O.P ("Value of N_def = " + n_def);
        S.O.P ("Value of N_Pro = " + n_pro);
        S.O.P ("Value of N_pub = " + n_pub);
    }
} // sdc
// Output
// Sdc s2 = new sdc();
// 10 20 30 40 20 30 40
```



// Compile the above program as

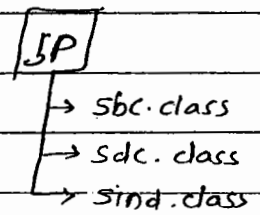
javac -d . sdc.java

Ensure that SDC.class must be generated.




```
// Sind.java
package sp;

public class Sind
{
    Sbc C1 = new Sbc(); // Has-A relationship within the package
    public Sind()
    {
        // S.O.P ("Value of n_pri=" + C1.n_pri); // Not Accessible
        S.O.P ("Value of n_def = " + C1.n_def);
        S.O.P ("Value of n_pro = " + C1.n_pro);
        S.O.P ("Value of n_pub= " + C1.n_pub);
    }
} // Sind
```

// Sind s3 = new Sind();
// 10 20 30 40 20 30 40



Compile the above program as
javac -d . Sind.java.
Ensure that Sind.class must be generated.

// SPDemo.java
class SPDemo
{
public static void main (String args[])
{
Sy.O.P ("w.r.t Sbc");
Sp.Sbc s1 = new sp.sbc(); // 10 20 30 40
S.O.P("w.r.t. Sdc");
Sp.Sdc s2 = new sp.sdc(); // 10 20 30 40 20 30 40
S.O.P ("w.r.t. Sind");
sp.Sind s3 = new sp.Sind(); // 10 20 30 40 20 30 40
} // main
} // Sind

Compile the above program as
cmd> javac spdemo.java. ↵
Ensure that spDemo.class File must be generated.
Run the Program as
cmd> java spDemo ↵

```
//Odc.java
package op;
public class odc extends sp.sbc // IS-A Relationship accross
{                                // the package.
    public Odc()
    {
    // S.O.P(" value of n.pri = " + n.pri); // Not accessible
    //S.O.P(" Value of n.def = " + n.def) ; //Not accessible
     S.O.P(" Value of n.pro = "+ n.pro);
     S.O.P(" Value of n.pub = " +n.pub);
    }
} // Odc
// Output : 10 20 30 40  30  40
```

op → Odc.class

Compile the above program as -

javac -d . Odc.java.

Ensure that odc.class must be generated

```
// Oind
package op;
public class Oind
{
    sp.sbc s1 = new sp.sbc(); //Has-A relationship accross
    public Oind()             // the package.
    {
      //S.O.P("Value of n.pri = "} + n.pri);
      //S.O.P(" Value of n.def = " + n.def);
      //S.O.P(" Value of n.pro = " + n.pro);
      S.O.P(" Value of n.pub "=") + n.pub);
    }
} // Oind
// Output = 10 20 30 40 40
```

op → Odc.class

```
// OPDemo.java
import op.odc;
import op.oind;
class OpDemo
{
    public static void main(String args[])
    {
        public OpDemo()
        { S.O.P("w.r.t odc");
          odc o1 = new odc(); // 10 20 30 40 30 40
          s.o.p("w.r.t oind");
          oind o2 = new oind(); // 10 20 30 40 40
    } // main
} // oind OpDemo
```

Compile the above program as -

cmd> javac OpDemo.java

Run the program.

cmd> java OpDemo

Q] How do you run a program in which a class placed in a package & contains main method?

→ We can run the program which is placed in a package & contains main method by using the following syntaxes.

Syntax-1: Useful of every OS.

cmd> java <packname>.<classname> (which contains main method.)

Syntax-2: works only on windows.

cmd> java <packname>/<classname>

* Creation of Jar File -

Date: 24-MAR-2011

* Exception Handling *

Exception Handling is one of the distinct Facility in Java Programming to build robusted applications.

When we write any program in any programming language we get two types of errors. They are -

i) Compile Time Errors.

ii) Runtime Errors.

Compile time errors will be resulted when the Java programmer is not following syntares of the language.

Runtime Errors are those which are generally listed or resulted when the normal user enters an invalid input at the time of running the program.

The language like C, C++, Pascal, COBOL etc are treated as weak Programming languages because these languages are not containing a powerful feature call-ed an Exception Handling.

In general, if we develop any project by using C, C++, pascal, COBOL and if the user of the project enters invalid input then these projects are by default displaying System/Technical Error Messages, which are not able to be understandable by the normal users of the project. It is highly recommended for these programmers to use their programming skills For converting System Error messages to User Freindly Error Messages

System Error Massages are those which are able to be understandable by the technical developers but not by the normal users of the application.

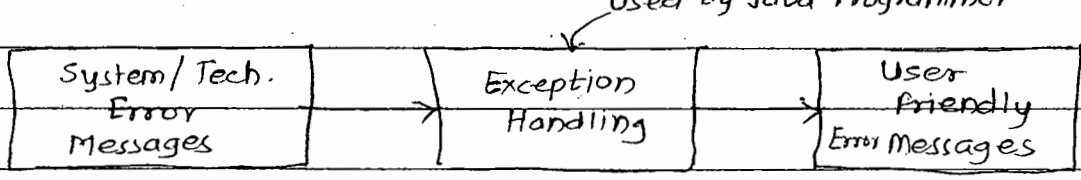
User Freiendly Messages are those which are

It is highly recommended for the industry programmer to use exception handling facility for converting System Error Messages into User Friendly messages for building Strong applications.

* Runtime Errors will be listed when we enter invalid input.
* Runtime Errors of Java are known as Exceptions.
  (In Information Technology, runtime error are also known as System Error Messages)
* If an exception occurs in Java program then program execution is abnormally terminated & control comes out of the program flow & by default generates System / Tech. Error Messages. which are not recommended in the s/w Industry.

* Definition of Exception Handling -

The process of converting System Error Messages into User Friendly Messages is known as Exception Handling.



Date: 25-MAR-2011

"Each & every operation / task / Action must be performed in Java with respect to an object."

Whenever an exception occurs in Java, program execution is abnormaly terminated, control comes out of the program & generates System Error Messages. We

So that we required an object., For creation of object we required class. F.

Finally, to generate a system Error message, JVM is creating an object of appropriate class & this class is known Exception related class.

* Types of Exceptions in Java -

According to Industry Standards, sun microsystems classified the exception into two types. They are -

i) Pre-defined / Built-in Exceptions
ii) User defined / Programmer / Secondary / Custom defined Exceptions

1) Pre-defined Exceptions -

Pre-defined exceptions are those which are developed by Sun microsystems & supplied as a part of Jdk to deal with all the Universal Problems -

Some of Universal Problems are -

a) Division By Zero
b) Invalid Bounds of the Array
c) Invalid Formats of the Numbers.

etc.

2) User Defined Exceptions -

User defined exceptions are those which developed by Java Programmer & supplied as a part their project to deal with common specific problems

Some of the common specific problems are

a) Invalid Salary of an employee
b) Invalid ages of the human beings

Date: 26-MAR-2011

* Types of Pre-defined Exceptions -

Pre-defined exceptions are divided into two types. They are

i) Asynchronous Exceptions
ii) Synchronous Exceptions.

Asynchronous Exceptions are those which are always dealing with "Hardware Problems & External Problems." Some of the Hardware & External problems are -

a) Mouse Failure, Keyboard Failure, Motherboard Failure etc.
b) Power Failures
c) memory Problems (java.lang.StackOverFlowError)

Currently, a java program never deals with asynchronous exceptions because asynchronous exceptions are not fully developed by Sun Microsystems i.e Asynchronous Exceptions are in R & D (Research & Development)

In Java Programming there exist one pre-defined superclass. For handling all asynchronous exceptions known as Java.lang.Error.

Synchronous Exceptions are those which are always dealing with "programmatic runtime errors". According to Sun Microsystems, Synchronous Exceptions are classified into two types. They are -

i) Checked Exceptions
ii) Unchecked Exceptions.

Checked Exceptions are those which the subclasses of java.lang.Exception class. (In other words, any kind of failure occures during hard disk checking process

Ex: i) IOExceptions
ii) FileNotFoundException
iii) ClassNotFoundException.

All the above classes are the subclasses of java.lang.Exception

Unchecked Exceptions are those which are the sub classes of java.lang.RuhTimeException.

Ex: ~~i) java.lang.RunTime~~

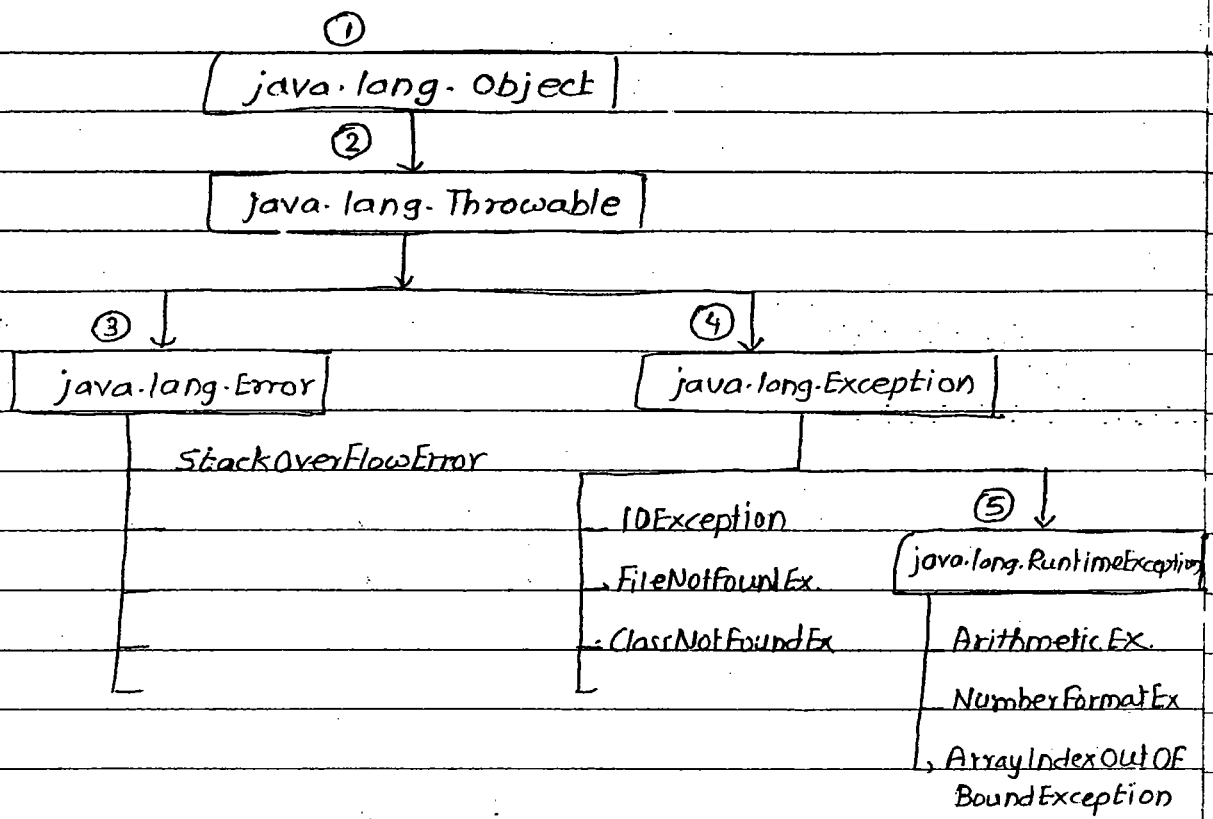
java.lang.Exception
|
java.lang.RunTimeException
- java.lang.ArithmeticException.
- java.lang.ArrayIndexOutOfBoundException
- NumberFormatException

Runtime Exception is one of the subclass of java.lang.Exception. But RuntimeException should not be included under checked Exception.

On overall the super class for all synchronous Exceptions is java.lang.Exception.

In the s/w development the java programmer always deals with synchronous Exceptions only.

* Exception Handling Hierarchy Chart *



In the above hierarchy chart.

1) java.lang.Object -

It is the superclass for all the classes in Java.

2) java.lang.Throwable -

It is the pre-defined super class for all the ex--ceptions in Java. The basic purpose of this class is to check which types of exception occurs in java program.

3) java.lang.Error -

It is a superclass for all the asynchronous exceptions in java.

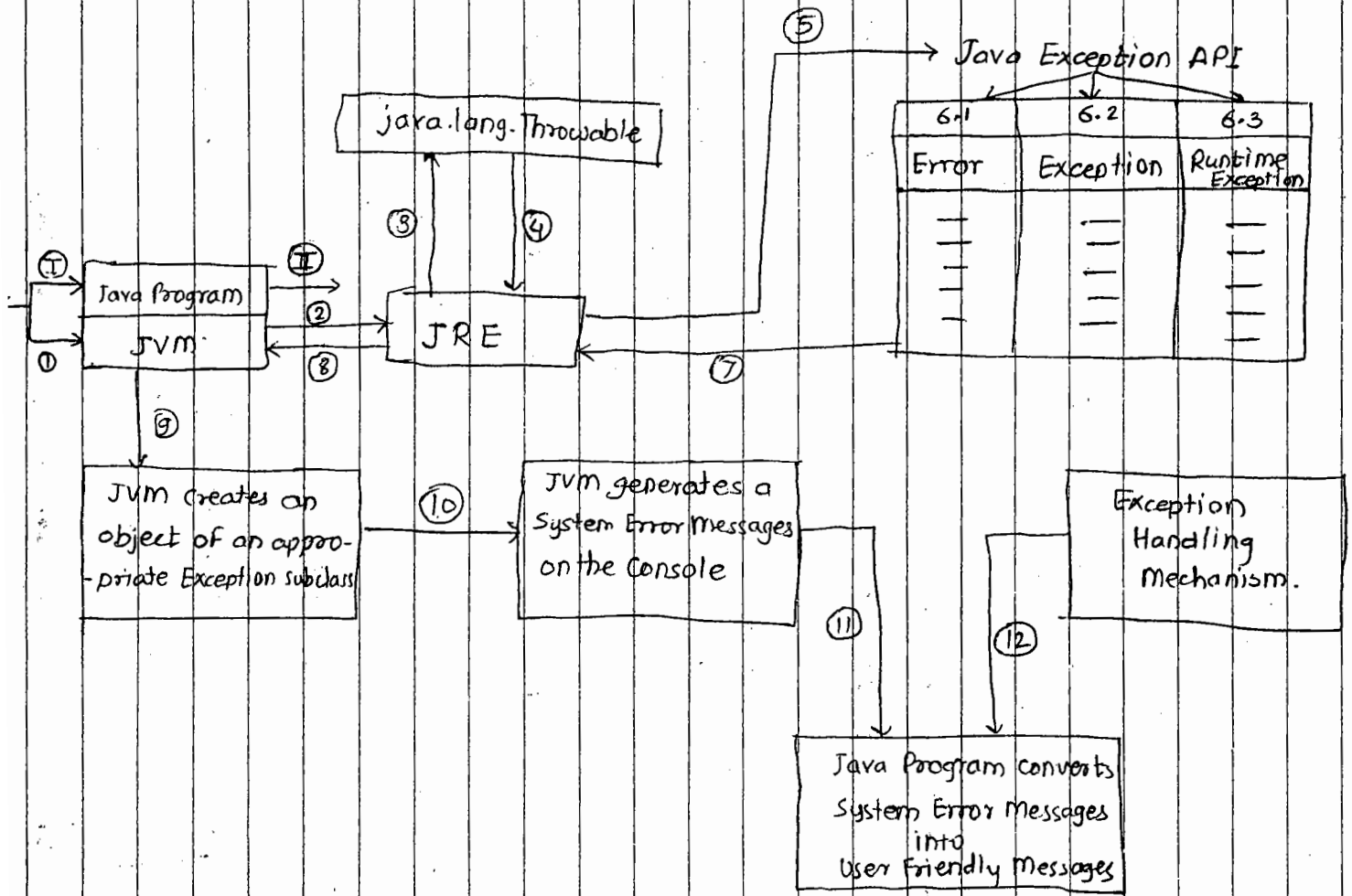
4) java.lang.Exception -

It is the superclass for all the synchronous exceptions (Specially for Synchronous Checked)

5) java.lang.RuntimeException -

It is the superclass for all synchronous

* Internal Flow of Exception Occurrence *



* Internal Flow of Exception Occurrence -

Step I - Java Programmer or Java Application user enters valid input to the Java Program.

Step II - JVM the process the Java program with valid input & gives valid output.

Step-1 - Java application user enters invalid input to the Java Program.

Step-2 : Java Program can not be processed by JVM with invalid input & JVM contacts to JRE (JRE = JVM + Java API)

Step 3: JRE contacts to java.lang.Throwable for Finding what type of exception occurs in the java program.

Step 4: java.lang.Throwable decides the type of exception which is occured in the Java program & gives to JRE.

Step 5: JRE is further contact into Java Exception API.

Step 6.1, 6.2, 6.3 : Java Exception API executes either 6.1 for asynchronous exceptions or 6.2 for synchro-nous checked or 6.3 for synchronous un-checked.

Step 7: Java Exception API gives an appropriate exception subclass to the JRE

Step 8: JRE gives the same exception subclass to JVM.

Step 9: JVM creates an object of appropriate exception subclass.

Ex. For division of two numbers program, if we enter 10 & 0 then JVM creates an object of ArithmeticException class.

Programmatically. ~~Arith~~

ArithmeticException ae = new ArithmeticException ("division by zero");

Step 10: JVM generates system Error Message on the console

Ex: After entering 10 & 0, JVM creates an object of ArithmeticException & we get the following message on console.

```
Exception in thread main java.lang.ArithmeticEx
        -ception : \ by zero
```

Generating System error messages are not recommended in the s/w industry.

Step 11 & 12: As a java programmer, we convert the system Error messages into User Friendly Error messages by making use of exception handling.

For above error message, the user friendly Error message will be:

"Don't enter zero for denominator."

* Technical Definition of Exception -

An exception is an object occurs at runtime which describes the nature of the message. The nature of the message can be either System Error Message or User

Ex:

ArithmeticException Ae = new ArithmeticException(
"Divide by Zero");
Nature of the Message

Date: 28-MAR-2011

* Handling Exceptions in Java *

When we write any Java Program, there is possibility of occuring many exceptions. We know that any exception occurs in Java then by default Java Environment displays System Error Messages which is not recommended in the industry.

Industry is always recommended to the Java programmers, convert the system error messages into user freendly Error messages.

To display the User Friendly Error messages instead of System Error messages we need to use the following Five keywords.

i) try
ii) catch
iii) Finally
iv) throws
v) throw

* Syntax for Exception Handling -

```
try
{
    Block of Statements which causes problems at
    Runtime
```

```
catch ( <TypeOfException> <var> )
{
    Block of Statement(s) which provides User Friendly
    Messages
}
catch ( <TypeofException2> <var> )
{
    Block of Statement(s) which provides User Friendly
    messages.
}
    :
catch ( <TypeofException n> <var> )
{
    Block of Statement(s) which provides User Friendly
    messages.
}
finally
{
    Block of Statement(s) which executes compulsorily.
}
```

* Try Block -

1) This is the block which is always containing the block of statements which causes problems at runtime.

2) If any exception occurs in try block then program execution is abnormally terminated & control comes out of the try block & excutes appropriate catch block.

3) After executing the appropriate catch block, control

4) Each & every try block must contain at least one catch block but it is highly recommended for the java programmer to write multiple catch blocks. for generating more no. of user friendly messages

5) Each & every try block must be followed by catch block i.e. No intermediate statements are allowed between try & catch blocks.

6) One try block can contain another try block i.e. Nested / Inner try blocks can be possible.

* Catch block-

1) This one of the block which will provide User friendly error messages by suppressing system error messages.

2) It is highly recommeonded for the Java programmer to write more than one catch block for generating multiple User Friendly messages.

3) At any point of time one catch block will execute out of multiple catch blocks. if an exception occurs.

4) In the catch block.

we declare an object of an appropriate exception

```
ArithmeticException ae ; // Declaration done by the Java
                         // Programmer.
```

Whenever ArithmeticException occurs, JVM will refe

// Referencing By JVM
ae = new ArithmeticException("/ by zero");

* Finally Block - Date: 29-MAR-2011

1) This is one of the block in which we write the block of statements which will relinquish (release or close) the resources (Files, databases) which are appending try block.

2) Finally block will execute compulsorily. ☺

3) writing Finally block is optional.

4) within the Finally block one can also write try-catch-blocks.

5) For per java program, it is mandatory for Java progr-ammer to write only one Finally block for releasing the resources which are appending try-block.

* Conclusion -

Case 1: If an exeception occurs then
- ⓐ A part of try block will execute.
- ⓑ An appropriate catch block will execute.
- ⓒ A Finally block will execute (if we write).

Case 2: If no exception occurs then
- ⓐ Complete try block will execute.
- ⓑ Finally block will execute (if we write)

* throws keyword -

Throws is a keyword which gives an indication to the calling function to keep the called function under try & catch blocks.

Calling functions are known as specific functions developed by Java Programmer & called functions are known as common functions developed by language vendors.

Throws keyword must be always be used as a part of method heading.

Throws keyword describes the number of exception occured in method body & it will establish the bridge between called function & calling function.

Exceptions are always raised in called functions & handled in calling functions for converting System error messages into User friendly error messages.

* Syntax :-

```
<returntype> <methodname> (<list_of_params>) throws
                    <exception1>, <exception2> ----<exceptionn>
{
    Block of Statement(s) ;
}
```

* Write a Java Program which illustrate the concept of throws keyword in the context of Pre-defined methods

→

java.lang. Integer

↓

public static int parseInt(String) throws

Here, parseInt(-) is not a normal method and it is one of the exception method defined by Sun Ms. Whoever is calling this parseInt(-) method, they must keep it into try-catch blocks for generating user friendly error messages.

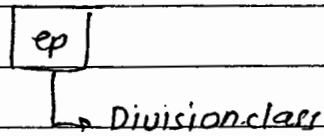
```
class Exception1
{
    public static void main(String args[])
    {
        try
        {
            String str = args[0];
            int n = Integer.parseInt(str);
            System.out.println(n);
        } catch(NumberFormatException nfe)
        {
            System.err.println("Please enter only integer values");
        }
    } //main
} //Exception1
```

* Write a Java programmer which illustrate the concept of throws keyword for User defined methods

→

```
package ep;
public class Division.
{
    public void divide(String s1., String s2) throws
                NumberFormatException, ArithmeticException
    {
```

```
        int z = x/y;
        System.out.println(" Division is : " + z);
    }
} // Division
```



Compile the above program as

```
javac -d . Division.java
```

Ensure that Division.class file must be generated.

```
// Exception2
import ep.Division;

public class Exception2
{
    public static void main (String args[])
    {
        String s1 = args[0];
        String s2 = args[1];

        Division div = new Division();
        try
        {
            div.divide (s1, s2);
        }
        catch ( NumberFormatException nfe)
        {
            System.err.println(" Enter integer values only");
        }
        catch ( ArithmeticException ae)
        {
```

Date: 30-MAR-2011

*. No. of Approaches to Find details about unknown Exceptions:

Whenever we develop any Java program there is a posibility that many no. of exceptions may occur but the Java programmer may not be aware of or may not be knowing about all the ~~exce~~ specific exceptions which are occuring in the Java program.

In order to knows the details about unknown exceptions, in Java programming we have three types of Approaches. They are.

i) By using an object of java.lang.Exception
ii) By using printStackTrace() method.
iii) By using getMessage() method.

1) By using an object java.lang.Exception -

An object of java.lang.Exception is containing the details about the all the exceptions which are occuring in java program. Because java.lang.Exception is the superclass for Synchronous checked & unchecked exception

When we use an object of java.lang.Exception in a catch block of our java program, we get the following two details.

a) Name of Unknown Exception
b) Nature of the Message.

Example:

```
=
try
{
    =
    int x = 100/0;
} catch (Exception e)
{
```

The above S.e.p statement will print the following message.

java.lang.ArithmeticException : / by zero
Name of Unknow Exception Nature of Message

The above code is executed when the Arithmetic Exception as follows

```
Exception e = new ArithmeticException("/ by zero");
```

Note: Whenever an exception occurs in Java Program Program execution is abnoramally terminated & before control goes out of the program, the exception details are pushed into stack.

Exception details are name of the exception, nature of message & line no. where an exception taken place.

2) By using printStackTrace() method -

printStackTrace() is one of the predefined method available in java.lang.Throwable class and it is further inherited into both java.lang.Error and java.lang.Exception.

When we use printStackTrace() method in our application it gives three details. They are

a) Name of the Unknown Exception
b) Nature of the Message
c) Line number where the exception has taken place.

* The Prototype of printStackTrace() method -

java.lang.Throwable

A java programmer should not call printStackTrace() method as a part of S.O.P or ~~S.e.~~ System.err.println()

Example:

```
try
{
    int n = 10/0;
}catch (Exception e)
{
    //System.out.err.println(e.printStalkTrace());
    e.printStackTrace();
}
```

The above e.printStackTrace() method will print the following message

java.lang.~~Exce~~ArithmeticException : / by zero
(Name of Unknown Exception) (Nature of Message)

at line no: X
(line no. where Exception raised.)

3) By using getMessage() method -

getMessage() is one of the predefined method available java.lang.Throwable class & it is further inherited into both java.lang.Error & java.lang.Exception. If we use this method as a part of our application, we get the Nature of the message which is occured because of the exception.

* The prototype of getMessage() method -

Example:-

```
=
try
{
  =
  int n = 10/0;
  =
}
catch (Exception e)
{
  System.error.println( e.getMessage());
}
```

The above e.getMessage() method will print the following message.

cmd> / by zero

Note: Whenever we develop any applications in Java, it is mandatory For the java programmer to write specific exceptions First in any order (ArithmeticException, NumberFormatException etc) and atlast we write a generalized exception called java.lang.Exception. Because it can hold all the exception which are occuring in java program which are not matching with specific exceptions. If the above order is revered we get the compile time error.

* err is an object of PrintStream class created in System class as a static data member & it is used to display user Friendly error messages.
* Out is an object of PrintStream class created in

* in is an object of InputStream class created as static datamember in a system class & it is used for getting the control of keyboard.

Date: 31-MAR-2011

User / Programmer / Custom defined Exception :-

User defined exceptions are those which are developed by Java Programmer & supplied as a part of their project to deal with common specific problems.

Some of common specific problems are

a) Invalid Salary of an employee.
b) Invalid ages of human beings
c) Invalid number of password characters

* Guidelines / Steps for developing User defined Exceptions -

If we want to develop any user defined exceptions as a part of our java project then the java programmer must follow the following sequence of steps

Step 1: Choose an appropriate package & ensure whose it should be first executable statement for specifying User Defined Exception Subclasses

Step 2: Choose an appropriate User Defined Class name & ensure that whose modifier must be public & this class is going to be called as User Defined Exception Subclass. (It is one which extends either java.lang.Exception or java.lang.RuntimeException)

Step 3: whichever class is selected in step 2 must extends either java.lang.Exception or java.lang.RuntimeException for gaining the power of exception to stop the program execution abnormally.

Step 4: Each & every User Defined Exception subclass must contain a parameterized constructor by taking string as a parameter. Here, string parameter represents nature of the message which is resulting when the exception has taken place.

Step 5: Each & every user defined ~~subclass~~ Exception subclass parameterized constructor must call parameterized constructor of either java.lang.Exception or java.lang.RuntimeException (~~Because~~ By super(msg). Here msg represents nature of the message. (Because if the java programmer is not specifying specific exceptions in their program & ~~in~~ if specific exceptions are occuring in the java program, then the specific exceptions must be processed or taken by either java.lang.Exception or java.lang.RuntimeException)

Step 6: whichever the user defined exception subclass we are placing in the package that class name must be given as filename with an extension .java.

* Develop User Defined Exception Subclass For dealing with negative salary of an Employee.

→ // Nsal.java // - - → Step-6

```
package np; // - - → step 1

                step 2          step 3
                 ↓               ↓
public class Nsal extends Exception
{
    public Nsal (String s) // - - → step 4
    {
        super(s); // - - → step 5
    }
} // Nsal.
```

* Develop User Defined Exception Subclass to deal with positive salary of an employee (Complement the employee and stop the program execution)

→ // Psal.java

```
package pp;

public class Psal extends Exception
{
    public Psal (String s)
    {
        super (s);
    }
} // Psal
```

Compile the above programs as

javac -d . Nsal.java

javac -d . Psal.java

Date: 1-Apr-2011

* Write a java program which will check the salary of the every employee by making use of user defined exceptions.

→ //Emp.java

```
package org.hyd;
import np.NSal;
import pp.PSal;

public class Emp
{
    public void decideSal(String s) throws NSal, PSal,
    {                                      NumberFormatException
        int sal = Integer.parseInt(s);
        if (sal <= 0)
        {
            NSal no = new NSal("Invalid Salary");
            throw(no);
        }
        else
        {
            PSal po = new PSal("valid Salary");
        } throw(po);
    } // DecideSal
} // Emp


//SatyaEmp.java
import org.hyd.Emp;
import np.NSal;
import pp.PSal;

class SatyaEmp
```

```
public static void main(String args[])
{
    try
    {
        String sal = args[0];
        Emp eo = new Emp();
        eo.decideSal(sal);
    }
    catch(NSal no)
    {
        System.err.println("Don't enter negative salary");
    }
    catch(PSal po)
    {
        System.err.println("Ok, Valid salary");
    }
    catch(NumberFormatException nfe)
    {
        System.out.println("Enter integer salary");
    }
    catch(ArrayIndexOutOfBoundsException ae)
    {
        System.err.println("Please, enter the salary");
    }
    catch(Exception e)
    {
        System.err.println(e);
    }
    finally
    {
        System.out.println("I am From Finally");
```

In the above programs,

i) np.NSal & pp.PSal comes under development of Stage-I And they are known as User Defined Exception Sub class.

ii) Org.hyd.Emp is comes under stage-II development. This class is known as User defined generalized class. which makes use of the classes Np.Nsal & pp.PSal

iii) SatyaEmp is comes under Stage-III development and it is known as User defined Specific class. which makes use of Stage-I & Stage-II classes

In general in realtime projects whenever we develop any user defined exceptions, which can be developed in above three stages

* throw keyword -

throw is a keyword which is used for hitting / raising / generating / populating the exceptions which are occuring in method body.

In other words throw keyword will establish the bridge between method body to method heading.

The place of using throw keyword is in the method body. whenever we use throw keyword as a part of our method body, it is mandatory for the java programmer to use throws keyword as a part of method heading. Otherwise we get compiletime error.

Syntax:1

<class name1> ob1 = new <class name2>(msg);

OR

throw (new <class_name>(msg)) ;

Here,

* <class_name1> represents subclass of either java.lang. Exception or user Java.lang.RuntimeException
* msg represents the nature of the message.

Syntax : 2 (throw keyword alongwith throws keyword) -

<return_type> <method_name>(<list_of_params>) throws <clsname1>, <clsname2>, - - - - <clsnameN>

{

~~if(test cond~~

~~Block of~~

if (~~clsn~~ Test cond1)

{

<clsname> ob1 = new <clsname>(msg);

throw (ob1) ;

}

:

if (Test cond N)

{

<clsnameN> obn = new <clsnameN>(msg);

throw (obn);

}

}

Here,

<clsname1>, <clsname2> - - - - <clsnameN> represents list of subclasses of either java.lang. Exception or java. lang. RuntimeException.

Q) Differences between throws & throw keyword?

→

throw keyword	throws keyword.
i) throw is a keyword used for hitting or raising the exceptions which are occuring in the method body.	i) throws is an keyword which gives an indication to calling function to keep the called function in try-catch blocks
ii) throw keyword must be used always in method body.	ii) throws keyword must be used always in Method Heading
iii) whenever we use throw keyword as a part of the method body, it is mandatory for the java programmer to use throws keyword as a part of method heading	iii) When we use throws keyword as a part of method heading, it is optional for the java programmer to use throw keyword as a part of method body.

* Develop Stage I, Stage-II and Stage-III classes for validating whether the username & password is correct or not.

→ // LoginException.java

```
package lp;

public class LoginExcep extends Exception
{
    public LoginExcep(String s)
    {
        super(s);
    }
```

Compile the above program as

LP

javac -d . LoginExcep.java → LoginExcep.class

```
// LoginProcess.java.
package lp1;
import lp.LoginExcep;

public class LogicProcess
{
    public void isLogin(String user, String pwd) throws LoginExcep.
    {
        if( user.equals("Sathya") && pwd.equals("Test"))
        {
            System.out.println("User/pwd is correct, Proceed");
        }
        else
        {
            LoginExcep obj = new LoginExcep("Access Denied");
            throw(obj);
        }
    } // main
} // LoginProcess
```

Compile the above program as

LP1

// ~~Sathya~~ javac -d . LoginProcess.java. → LoginProcess.class

```
// LoginUser.java.
import lp1.LoginProcess;
import lp.LoginExcep;
class LoginUser
```

```
public void static void main(String args[])
{
    try
    {
        String user = args[0];
        String pwd = args[1];

        LoginProcess lp = new LoginProcess();
        lp.islogin(user, pwd);
    }
    catch (LoginExcep le)
    {
        System.err.println("Username / Password Invalid");
    }
    catch(ArrayIndexOutOfBoundsException ae)
    {
        System.err.println("Plz, enter username & password");
    }
    catch (Exception e)
    {
        system.err.println(e);
    }
} //main
} // LoginUser
```

i)

2)

Date: 2-APR-2011

* Data Conversions in Java *

In most of the real time applications, we may come accross various data conversions in Java Program. In Java we have six data conversions. They are

i) Converting String data into fundamental / Numerial Data. -

In order to convert string data into fundamental we use the following generalized pre-defined method which is present in each and wrapper class.

```
wrapper ClassName
        ↓
public static xxx parseXxx (String s) throws
                                NumberFormatException
```

ex: java.lang.Integer

```
public static int parseInt (String s) throws
                                NumberFormatException
```

Here, parameter string s represents any numerical value enclosed within double quotes

In place of string parameter, if we enter anything other than numerical value within the double quotes (" ") then we get a predefined exception called java.lang.NumberFormatException.

```
Ex.  String s1 = "40000";
     String s2 = "100.75";      } Numerical String Data

     int n1 = Integer.parseInt (s1);
     double d = Double.parseDouble (s2);      } fundament Data.
```

2) converting fundamental data into string data -

In order to convert any fundamental data to string data then we use the following pre-defined overloaded

public static final String valueOf (byte)
" " " (short)
" " " (int)
" " " (long)
" " " (Float)
" " " (double)
" " " (char)
public static final string valueOf (boolean)

In general string class is containing the following generalised value of method.

public static final string ~~xxx~~ valueOf (xxx)

Here, xxx represents any valid fundamental types

Ex int a = 40000;
String s = String.valueOf(a);

Float F = 10.75F;
String s2 = String.valueOf(F);

3) Converting fundamental data into object data -

In order to convert fundamental data into object form we use the following table

4)

Datatype	wrapperClass Name	Constructor.
byte	Byte	Byte (byte)
short	Short	Short (short)
int	Integer	Integer (int)
long	Long	Long (long)
Float	Float	Float (Float)
Double	Double	Double (double)

when we develop distributed applications, there exist set of client-side application & server side applications. when the data is visiting between client & server-side applications over the n/w, there is a posibility that the data can be trapped by the backers when we send in the form of fundamental data. It is highly recommended for the programmer to convert the fundament values into corresponding object form. Because by default object data cannot be modified

In general each & every wrapper classes containing the following generalized pre-defined constructor for converting fundament data wrapper class object.

ex. Wrapper Classname ↓

wrapperClassName(xxx)

Here, xxx represents any fundamental data type

① int i = 10;
Integer io = new Integer(i);

② Float F = 10.75f;
Float Fo = new Float(F);

4) Object Data into Fundament Data -

In order to convert object data into fundamental data we use the following table.

Datatype	Wrapper Class Name	Conversion method
byte	Byte	pubpublic byte byteValue()
short	Short	public short shortValue()
int	Integer	public int intValue()

Datatype	wrapper ClassName	Conversion Method
Float	Float	public Float floatValue()
double	Double	public double doubleValue()
char	Character	public char charValue()
boolean	Boolean	public boolean booleanValue()

In general each & every wrapper class is containing the following generalised pre-defined instance method for converting object data into fundatamental data

wrapper Classname ↓

public xxx xxxValue()

Here, xxx represents any Fundamental datatype

Example:

① Integer io = new Integer(40000);
int i = io.intValue();

② Float fo = new Float(10.75f);
float f = fo.floatValue();

5) Converting string data into objed data -

In order to convert String data into wrapper Class object data we use the following table.

Datatype	wrapperClasiName	Contructor Name
byte	Byte	Byte(String)
short	short	Short(String)
int	Integer	Integer(String)
long	Long	Long(String)
Float	Float	Float(String)
double	Double	Double(String)
char	Character	-

In general, each & every wrapper class contains the following generalized parameterized constructor

Wrapper ClassName ↴

Wrapper ClassName (String)

Example:

① String s = "40000";
Integer io = new Integer(s);

② String s1 = "10.75F";
Float fo = new Float(s1);

6) Converting object data into String data -

To convert wrappe class object data into string data we have the following pre-defined instance method present in each & every wrapper class.

Wrapper ClassName ↴

public String toString()

Example:

① Integer io = new Integer("40000");
String s = io.toString();

② Float fo = new Float("10.75F");
String s1 = fo.toString();

★ Example

```
Float f1 = 10.75F;                    ①
Float fo = new Float(f1);             ②
Float f2 = fo.floatValue();           ③

String s = String.valueOf(f2);        ④
String s1 = fo.toString();            ⑤
```

Date: 5-APR-2011

* Applets *

In Java Programming, a java programmer can develop two types of Applications. They are -

i) Stand Alone Applications.

ii) Distributed Applications.

* Stand AlOne Applications are those which are running in the context of local disk and whose results are not sharable.

Every stand alone application contains main(-) method (for program execution) and System.out.println(-) (for displaying the result on the console)

The main(-) method of stand alone application is also known as Life Cycle Method.

ex. All our previous program comes under Stand alone applications.

* Distributed Applications are those which runs in the context of www/Browser and whose results are sha-rable accross the Globe.

Every distribute application will contain the life cycle methods. In other words if any application is running in the context of browser, that application will contain life cycle methods / loop Back methods + Some pre-defined method to display the result on the browser.

Life Cycle methods are those which are called by the browser automatically when certain operations are performed on the browser.

In the initial days of Sun Microsystems, there was a concept called ~~Ap~~ applets, whose basic aim is to develop distributed applications.

Any program we write with the concept of the applets, those programs are known as applet programs

* Definition of Applet -

An applet is a java programmer which runs in the context of www/ browser and whoose results are sharable accross the globe.

~~For~~

For the concept of applets, sun microsystems has developed a pre-defined class called Applet and it is present in java.applet.*; ~~and th~~

In order to deal with Applet programming we must import a pre-defined package called java.applet.*.

Since applet programs are running in the context of bro--wser and applet concept contains life cycle methods.

* Life Cycle methods of Applet -

Applet contains four life cycle methods, which are present in java.applet.Applet class. They are.

i) public void init()
ii) public void start()
iii) public void stop()
iv) public void destroy()

1) Pubic void init()-

i) This is one of predefined life cycle method present

ii) This method called by the browser only once when the application loaded in the browser or makes a very first request to the applet. ~~In this~~

iii) In this method we write the block of statement which will perform one-time operation. One time operations such as opening a file, obtaining a db connection, initialization of parameters etc

iv) In general in this method we write the block of statements which will obtain the resources (Files databases etc.)

2) public void start() -

i) It is one of predefined life cycle method present in Applet class.

ii) After execution of init() method at the time of making First request, Start() method will be executed. From second request to further subsequent requests, only start() method will be executed. Hence, start() method will be called each & every time.

iii) In this method we write the block of statement which will perform repeated operations such as reading the records from the file, reading the records from the db etc.

iv) In general we always write business logic in start method.

3) public void stop() -

i) This is one of pre-defined life clyce method called by the browser. on demand.

ii) Stop() method executed when the application pro-grammer / user minimizes the window where the applet application is running.

iii) In this method we write the block of statements which will temporarily relinquish the resource (Files, databases). ~~temporarily~~.

4) Public void destroy() -

i) This one of the pre-defined life cycle method present in applet class

ii) This method called by the browser on demand when we terminate or close the window. where the applet application is running.

iii) In general we write the block of statement to relinquish the resources permentely which are ob-tained in init() method.

Hence, all the above life cycle methods defined with null body in Applet Class.

In order to provide our own logic to the life cycle methods, we need to override those life cycle methods into our subclass by extending the java.applet.Applet Class

whichever class is extending java.applet.Applet class, whose modifier must be public other wise we get a ~~con~~ runtime error.

Date: 6-APR-2011

* Displaying the result of the Applet -

To display the output of the applet program, we use the following predefined method.

public void paint (Graphics)

After execution of start() method of applet, result of the applet program is ready. To display this result on the browser we use the above method.

In another words, paint method will be called automatically after execution of start method. Paint() defined in ~~ap~~ Applet Class with null body. To define output of the applet, we must override the paint(-) method in our derived class by inheriting the paint(-) method From Applet Class.

The parameter Graphics is one of the pre-defined class presents in java.awt.* package. When the control comes to paint(-) method, an object of ~~ga~~ Graphics class is implicitely referenced by JVM and it is de-clared by Java Programmer.

Graphics Class contains the following instance method which is used For displaying the result on the browser.

java.awt.Graphics

↓

public void drawString (String, int, int)

the result of applet always in string form.

⇒ First int parameter represents x-coordinate of the display area of the browser.

⇒ Second int parameter represents y-co-ordinate of the display area of the browser.

Therefor the result of the applet must be displayed in the intersection point of x & y co-ordinate of the display area of the browser.

* Write an Applet program which will display a message "Hello Applet" on the browser.

→
```
// MyApplet.java
import java.applet.*;
import java.awt.Graphics;

public class MyApplet extends Applet
{
    public void paint(Graphics g)
    {
        g.drawString("Hello Applet", 100, 100);
    }
} // MyApplet
```

Compile the above program as
javac MyApplet.java
Ensure that MyApplet.class must be generated.

* Number of ways to run the Applet Programs -

Applet programs can be executed in two ways. They are:

i) By using HTML program.

ii) By using appletviewer Tool.

1) By using HTML Program:

We know that an applet is a java program which runs in the context of browsers to make their results as sharable.

The real world browsers never understand the logic High-level, Middle-level and low-level language applications But they understand the logic of Mark up languages like SGML (Standard Generized Markup languages), HTML, ~~XML~~ DHTML, XML etc.

Since applets are written in Java they can not directly run in the browser. So that we need to write an HTML program in which we call applet program.

we know that HTML programmers will write their programs with the help of tags. So that sun microsystems has developed the following tag for HTML programmers to execute the applet programs.

```
<applet code = "<name of the class which extends Apple"
        height = "<int numerical value for height"
        width = "<int numerical value for width" >
</applet>
```

In the above tag <applet --> </applet> is known as Tag. Code, height, width are known as attributes.

The attribute code represents name of the class

Height and width represents integer numerical value in term of pixcels to reserved a selected portion of area with the display area of the browser. The above tag must be always written by the HTML Programmer within the <body> tag of HTML program

Ex:

```
<applet code="MyApplet"
        height="500"
        width="500">
</applet>
```

* write a HTML Program for running the Applet Program.-

→

```
//AppletExample.html
<HTML>
<HEAD>
    <TITLE> Applet Example </TITLE>
</HEAD>
<BODY>
    <applet code="MyApplet" height="500" width="500">
    </applet>
</BODY>
</HTML>
```

Save the above program with an extension either html or htm.

* Open the browser & type address of ApplicationExample.html.

2) By using appletviewer Tool -

Some browsers in the real world may not be supporting the applets. So that sun microsystems has developed a tool called appletviewer to run the applet programs in any circumstance without depending on the browser.

When we are using appletviewer to run the applet, the tag <applet --> </applet> must be written anywhere else in the Java program within the commented statements.

Example:

```
/* <applet code="MyApplet" height="500" width=500">
   </applet> */
```

Syntax:

```
cmd> appletviewer <filename>.java
```

Example:

```
cmd> appletviewer myApplet.java.
```

Ensure before running the applet program by using appletviewer we must compile the program.

* Write a Java program which will illustrate the concept of life cycle of Applet.

-> // LifeApp.java

```
import java.applet.*;
import java.awt.*;
/* <applet code = "lifeApp" height = "500" width = "500"
```

Date -

```
public class LifeApplet
{
    String s = " ";
    public void init()
    {
        s = s + " init ";
    }
    public void start()
    {
        s = s + " start ";
    }
    public void stop()
    {
        s = s + " stop ";
    }
    public void destroy()
    {
        s = s + " Destroy ";
    }

    public void paint ( Graphics g)
    {
        Font F = new Font ("Arial", Font.BOLD, 20);
        g.setFont (F);
        g.drawString (s, 100, 100);
    }
} // LifeApp
```

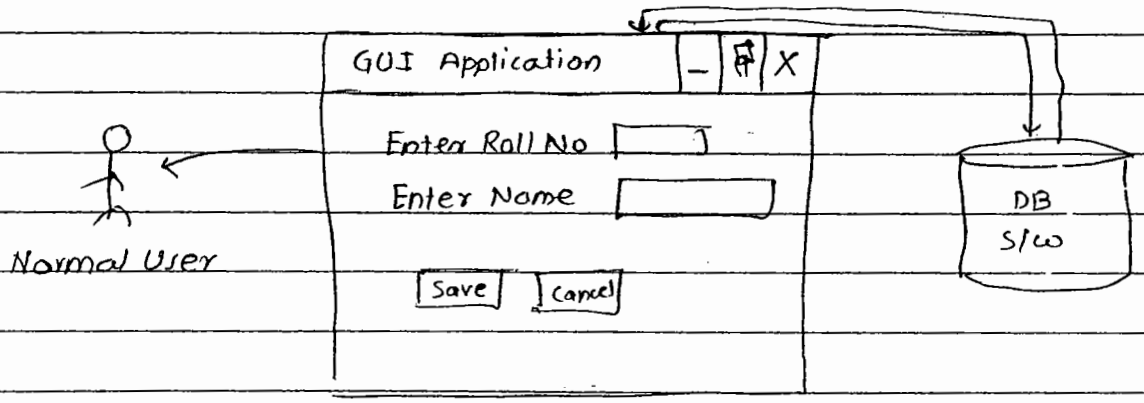
To see that destroy message, on the browser as a part of life cycle methods execution, we need to choose following Applet menu

Date: 7-APR-2011

* Windows Programming or AWT Programming *

AWT is one of the distinct feature of Java programming.

- The basic aim of awt programming is to develop GUI / Windows / Front-end / User Interfaces applications.



The GUI application is one which will always gives "look & feel" approach.

In awt programming to develop any GUI appln we require various GUI Component / Controls / Widgets

Ex: labels, Buttons, TextFields, ScrollBars etc.

In awt all GUI components are given in the Form of predefined classes and they are present in java.awt.* package.

In Java programming creating any GUI component is nothing but creating an object of particular pre-defined particular class

Ex: Create a component Button with caption "Save".

```
Button b = new Button("Save");
```

b ↑ Reference

"Save" ↑ Caption

In order to provide Functionality or life or beha-viour to the GUI components, we need to use various predefined classes & interfaces & they are present in java.awt.event.*;

Here event is a subpackage of awt which contains collection of pre-defined classes & interfaces which will provide functionality GUI components.

To Finally, to develop a complete GUI application, we need to import java.awt.* (for component creation) and java.awt.event.* (For component Functionality)

* Definition of Event -

The change in the state of objed is known as Event.

Ex: Button is clicked
CheckBox is checked
ScrollBar is adjusted vertically or Horizontally etc

* Types of GUI Components -

In awt programming GUI components are classified into two types. They are

i) Auxiliary Components
ii) Logical Components

* Auxiliary Components are those which will give "Touch & Feel" approach. or

Auxiliary Components are the hardware components which are used for interacting with GUI applications.

Ex: Mouse, keyboard.

* Logical Components are those which will give "look &

Logical component are those software components which are required for building GUI applications.

Logical Components are classified into two types

i) Passive / Non-Active.

ii) Active / Interactive

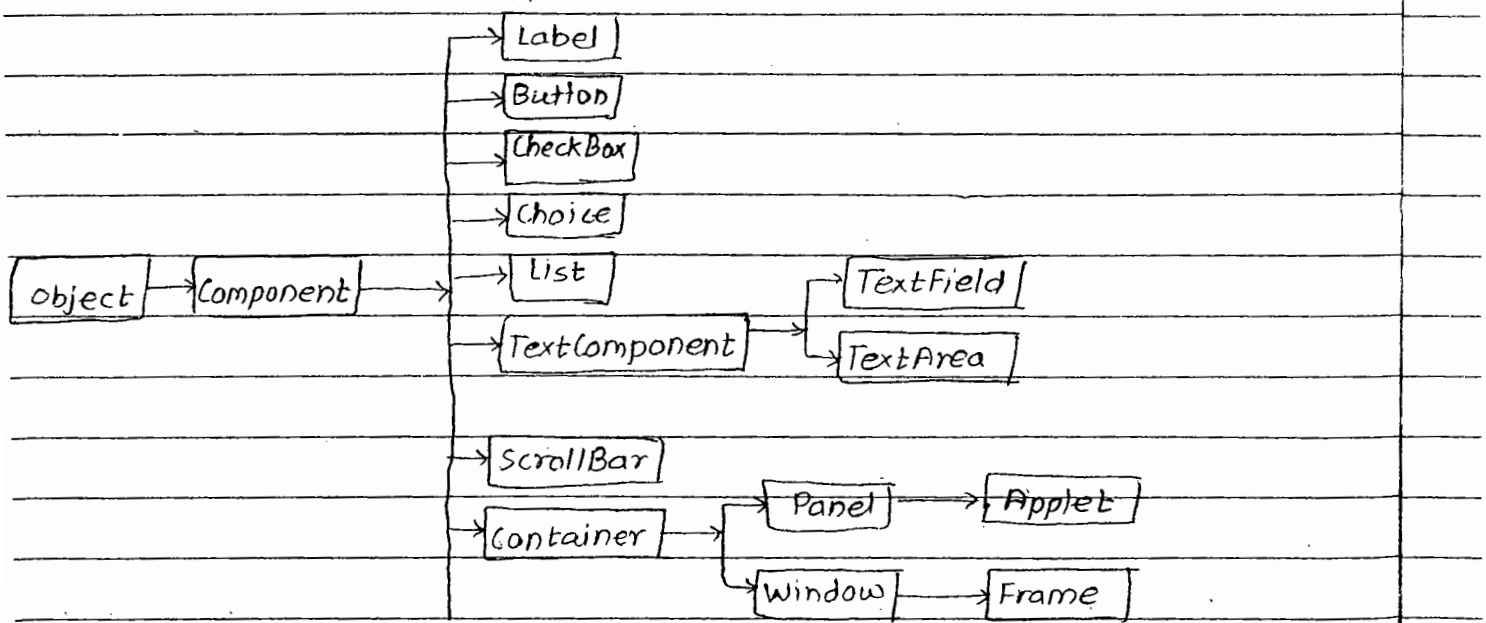
Passive components are those which does not have any interaction with the normal user / mouse.

Ex: label

Active components are those which are always gives interaction with the normal user / mouse.

Ex: Button, CheckBox, ScrollBar (except label) etc.

* AWT Hierarchy Chart *



In the above hierarchy chart -

java.lang.Object is the super class for all the classes

Date: 8-APR-2011

for collecting unused memory space for strengthening the performance of Java applications.

Java.awt.Component is the pre-defined abstract super class for all the specific components in windows programming. This class contains general properties regarding specific components such as changing the height & width of the components, changing the color of the components, changing the font of caption of the component etc.

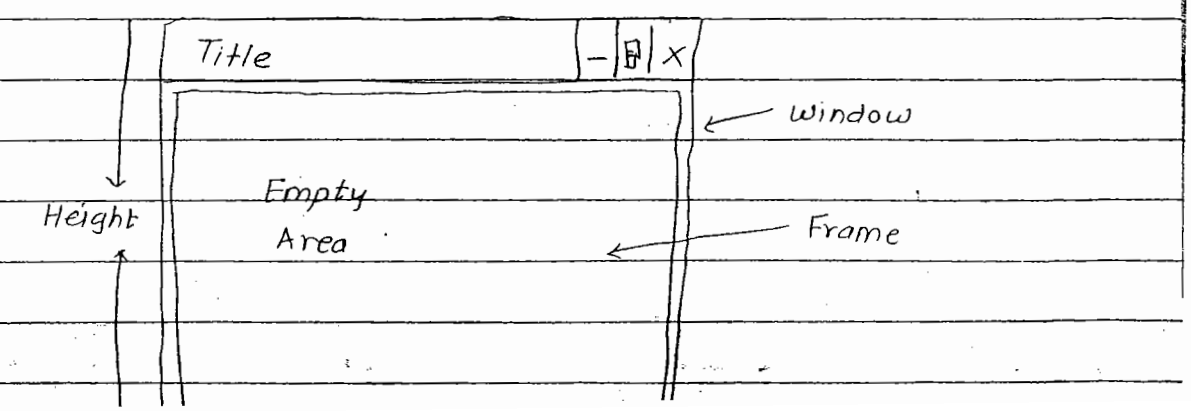
java.awt.container is one of the pre-defined class & whose object is used for combining or integrating or grouping different specific components as a single unit.

java.awt.Window is a pre-defined class & whose object is mandatory for all GUI applications.

java.awt.Frame is one of the bottom most subclass of all GUI components & whose object must be embedded within the window component. Technically, an empty area within the window is known as Frame.

whichever class is extending java.awt.Frame, using that class one can develop standalone GUI Applications.

If we want to develop any distributed GUI applications then we must choose a class & it must extends java.applet.Applet class.



* Steps For developing GUI Application -

Step 1: Create a window by creating an object of window class.

Step 2: Create a Frame by creating an object of Frame class and add it to window.

Step 3: Create a Container by creating an object of Container class & add it to Frame.

Step 4: Create specific components by creating objects of specific predefined classes for building GUI application and add them to Container.

* Write a Java Program which will create a window and a frame

```
// MyFrameDemo.java
import java.awt.frame;

class MyFrame extends Frame
{
    MyFrame()
    {
        setTitle("My Frame window");
        setSize(500, 300);
        setVisible(true);
    }
} // MyFrame BLC

public class MyFrameDemo
{
    public static void main(String args[])
    {
```

In the above program -

1) Public void setTitle (String) -

It is one of the pre-defined method present in window class & it is used for setting title of the window.

2) public void setSize (int, int) -

It is one of the pre-defined method present in window class & it is used for setting height and width of the window in terms of pixels. (A smallest addressable element on the screen is known as Pixel.)

3) public void setVisible (boolean) -

It is one of the pre-defined method present in component class and it is used for making the created components to be visible at the time of running GUI application by passing true for the boolean parameter. By default the created component will not be visible because the default of value of boolean parameter is false.

* Layout Managers *

Layout Managers are those which are organising the components in the container logically. or

Layout Managers are the pre-defined classes in the java which are organising the components in the container logically in certain order.

Based on organising the components in the container layout managers are divided into following four type

i) Border Layout
ii) Flow Layout
iii) Grid layout
iv) GridBag Layout

In order to set the layout manager in the container, as a java programmer we follow the following steps.

Step 1: Create an object of appropriate layout Manager class.

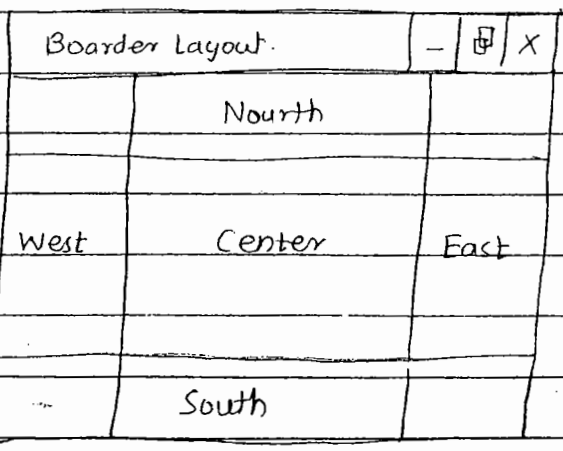
Step 2: Call a pre-defined method called setlayout(-) by passing an object of an appropriate layout manager class.

1) Border Layout -

i) It is one of the pre-defined layout manager.

ii) Border layout organises the components in the container logically in the form of regionwise.

iii) Diagramatic representation of Boarde layout is geiven below.



qualified region name then it is by default treated as Center.

v) Border layout is the default layout Manager in all Standlt alone GUI / Frame applications.

vi) Creating a Boorder layout is nathing but creating an object of Border layout class and calling setlayout(-) method.

```
Borderlayout bl = new Borderlayout();
setlayout(bl);
```

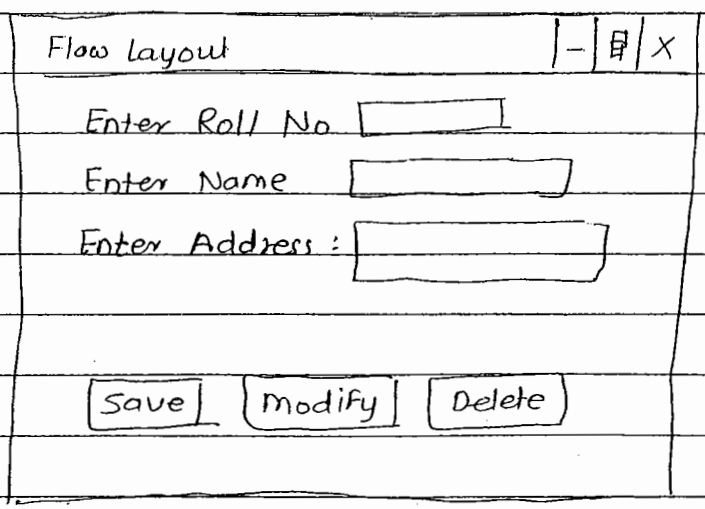
Date: 9-APR-2011

2) FlowLayout -

i) It is one of the pre-defined layout manager. This

ii) This layout organises the components in the container logically in the form of row wise.

iii) The diagramatic representation of Flow layout shown below.



v) Creating a Flow Layout is nothing but creating an object of FlowLayout class & call setLayout(-).

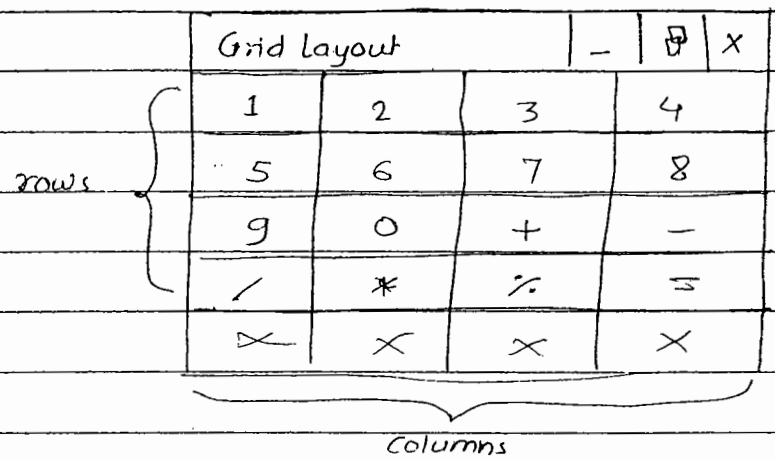
Ex:

```
FlowLayout FL = new FlowLayout();
setLayout(FL);
```

3) Grid Layout -

i) It is one of the predefined layout manager organises the components in the contain. logically in the form of rows & columns (like a matrix).

ii) The diagramatic representation of Grid layout is shown below.



iii) This layout is able to maintain all components same in size.

iv) This layout is not allowing the programmer vertical & horizontal space between Horizontal & vertical compo-nents & not able to maintain different components in different sizes.

an object of Grid layout class by specifying no. of rows & columns & call setLayout() method.

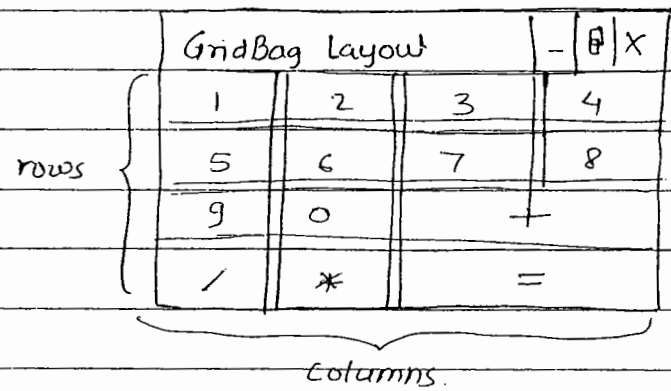
Ex:

```
GridLayout gl = new GridLayout (4,4);
setLayout (gl);
```

4) GridBag Layout -

i) GridBag layout is one of the pre-defined layout manager it organises the components in the container logically in the form of rows & columns.

ii) The diggramatic representation GridBag layout is shown below.



iii) This layout is able to maintain vertical & horizontal space between Horizontal & vertical components and it is also able to maintain different types of components or same components either in same size or in different size or in both.

iv) GridBag layout is of the revised layout of Grid layout.

object of GridBag layout class by specifying no. of rows & columns and call setlayout(-) method.

Ex.

```
GridBagLayout gbl = new GridBagLayout(4,4)
setLayout(gbl);
```

Before adding components to the container, the java programmer must ensure that container must contain an appropriate layout manager for arraging the components logical.

A container can contain any no. of layout managers logically.

* Label Component -

Label component is one of the passive component because the normal user never interacts with label components. label components never participates in Event Delegation model.

Creating a label is nothing but creating an object of label class.

Ex: `Label l1 = new Label("Username :");`

(l1 ↑ reference; "Username :" ↑ Label / Caption / Value)

The Functionality of label component is to describe or to express the activity Functionality of active components.

The nature of the label components is always belongs to string data.

* Label Class Profile -

i) Data Members -

ii) public ~~final~~ static final int CENTER = 1;

iii) public static final int RIGHT = 2;

The above Data members are known as label alignment modifier & they must accessed w.r.t class name. The default alignment modifier is LEFT.

ii) Constructors -

a) Label ()

b) Label (String)

c) Label (String, int)

iii) Instance Methods -

a) public void setText (String)

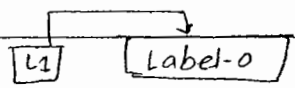
b) public ~~void~~ getTex String getText();

c) public void ~~set~~ alignment (int)

d) public ~~void~~ int getAlignment()

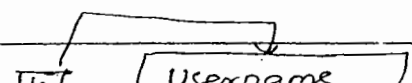
Constructor 1 is used For creating an object class without specifying value For the label. When we create any label w.r.t this constructor by default it contains label-0 as a default name which is of no use.

Ex. label l1 = new Lable ();



Constructor 2 used For creating an object of label class by specifying label name & without specifying any alignment ~~specif~~ modifier.

label l2 = new Label ("Username");



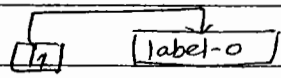
Constructor 3 is used for creating an object of Label class by specifying label value & alignment modifier value.

Ex: Label l2 = new Label ("Username", Label.RIGHT);

("Username") ↑ label value; (Label.RIGHT) ↑ alignment modifier

Methods ④ & ② are used for setting the label to label component & getting the label from the label components.

Ex: Label l1 = new Label();



S.O.P(l1.getText()); // Prints: label-0
l1.setText("Username");

Methods ③ & ④ are used for aligning the label of a label components, either at center or at right or at left.

Ex: Label l1 = new Label("Username");
int av = l1.getAlignment();
S.O.P(av); // Prints: 0 Default Left Alignment.
l1.setAlignment(Label.Right);

Note: If any class is containing set of set() methods then those methods will change the value of the object. Hence such methods are known as mutators / modifiers.
If any class is containing set of getXxx() methods then those methods will obtain the values from the object. Hence, such methods are known as Inspectors.

Date: 11-APR-2011

* Event Delegation Model *

The basic aim of Event Delegation Model is to provide the functionality to active components. In other words, all the active components participates in Event Delegation Model for getting functionality / behaviour / life during the execution time.

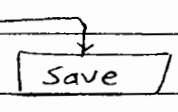
The concept of Event Delegation Model summerised in Four pages. They are -

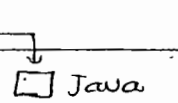
* First Phase -

a) Each & every GUI active component must be qualified with caption & reference.

Ex.

```
Button b1 = new Button ("save");
Checkbox cb = new Checkbox ("java");
```





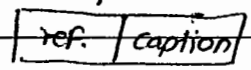
Here, b1, cb are known as references
Save & Java are known as Captions.

b) Whenever we interact with active component, that active component can be processed either based on caption or based on reference. Captions & references are useful at the time of writing code for the programmer to achieve the functionality of the component.

c) In Java execution environment, whenever we interact with any active component, automatically JVM will create an object of one Pre-defined class which will record or store caption & reference. In other words, as & when we interact with any GUI active component, JVM will create an object a generalised predefined

caption & reference of any GUI active components.

XXXEvent obj →



d) The following table gives GUI component name and whose corresponding XXXEvent class name.

Component Name	Event Name
Label	-
Button	ActionEvent
Checkbox	ItemEvent
TextField / TextArea	TextEvent
ScrollBar	AdjustmentEvent
Window	windowEvent

* Second Phase -

a) Whenever we interact with any GUI active component, we need to write a piece of code in terms of block of state-ments. We know that block of statements must be written always in methods. These methods are given by Sun Micro-systems without definition known as predesigned methods. In Java programming, pre-designed methods are known as Abstract Methods and they are present in interfaces. In windows programming interfaces are known as Listners & whose generalised notation is xxxListner.

Each & every GUI active component must contain one xxxListner.

b) The following table gives GUI active component name & its corresponding XXXListner.

Component Name	XXXListner.
Label	—
Button	ActionListner
Checkbox	ItemListner
TextField / TextArea	TextListner
Scrollbar	AdjustmentListner
window	windowListner

All the XXXEvent classes & XxxListners are present in a predefined package called java.awt.event.* ; package.

Third Phase -

a) Identify the type of abstract method present in XXX Listener for writing a code for getting the Functionality of the component.

b) The following table gives XXXListener & whose corresponding abstract method.

XXXListener	Abstract Method
ActionListener	public abstract void actionPerformed(ActionEvent ae)
ItemListener	public abs. void itemStateChanged (ItemEvent ie)
TextListener	public abs. void textValueChanged (TextEvent te)
AdjustmentListener	public abs. void adjustementValueChanged (AdjEvent ae)

* ~~Phase~~ Fourth Phase -

a) In order to provide functionality to any GUI active component, all GUI component must be register and unregister their events with appropriate listeners.

b) The registration must be done by the java programmer & the un-registration must be done by JVM.

c) The Following table gives GUI component name, whose corresponding registration method & unregistration method.

Component Name	Registration Method	Un-registration Method
Button	public void addActionlistener (ActionEvent)	public void removeActionListener(ActionEvent)
CheckBox	public void addItemlistener (ItemEven)	public void removeItemListener (ItemEvent)
TextField/TextArea	public void addTextListener (TextEvent)	public void removeTextListener (TextEvent)
Scrollbar	public void addAdjustment listner(AdjustmentEvent)	public void removeAdjustment Listener(AdjustmentEvent)

Event Delegation Model

Date - 12-APR-2011

* The Following table gives the summary of Event Delegation Model for all GUI Active Components

* Event Delegation Model *

nponent e	XXXEvent	XXXListner	Abstract Method	Registration Method	Unregistration Method
e)	—	—	—	—	—
on	ActionEvent	ActionListener	public void actionPerformed (ActionEvent ae)	public void addAction Listener (ActionListener)	public void removeActi Listener (ActionListene
: Click)	— " —	— " —	— " —	— " —	— " —
Box	ItemEvent	ItemListener	public void itemStateChanged (ItemEvent ie)	public void addItemListener (ItemListener)	public void removeIter Listener (ItemListe
ce	— " —	— " —	— " —	— " —	— " —
t e click)	— " —	— " —	— " —	— " —	— " —
)/TextArea	TextEvent	TextListener	public void textValueChanged (TextEvent te)	public void addTextListener (TextListener)	public void removeText Listener (TextListene
Bar	AdjustmentEvent	AdjustmentListener	public void adjustmentValueChanged (AdjustmentEvent ae)	public void addAdjustment Listener (AdjustmentListene	public void removeAdjustr Listener (Adj. Listener)
low	WindowEvent	WindowListener	WindowAdapter (contains 7 null Body methods)	public void addWindowListener (WindowEvent)	public void removeWindow Listener (WindowEvent)
se					

* Button Component -

1) Button is one of the active component & hence it participates in event delegation model.

2) Creating a button is nothing but creating an object of button class.

ex. Button b1 = new Button("OK");
(b1 ← reference, "OK" ← caption)

3) The purpose of the button component is to perform an automic unit of operations such as saving the details, deleting the details, updating the details of database.

* Button Class Profile -

* Constructors -

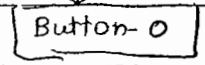
1) Button()
2) Button(String)

* Instance methods -

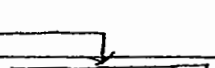
3) public void setLabel(String)
4) public String getLabel()
5) public void setActionCommand(String)
6) public String getActionCommand()
7) public void addActionListener(ActionListener)
8) public void removeActionListener(ActionListener)

Constructor 1 is used for creating an object of Button class without specifying the caption.

Ex. Button b1 = new Button();



ex. Button b2 = new Button("ok");



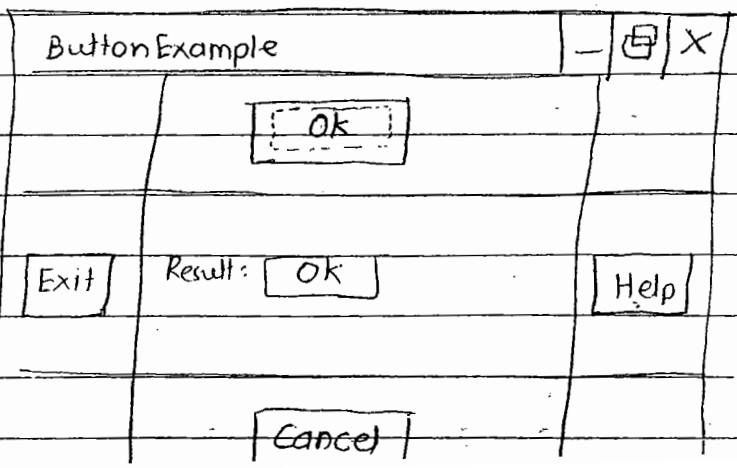
Methods ③ & ⑤ are used for setting the caption to the Button component.

Method ④ & ⑥ are used for geting the caption from the Button component.

```
Ex. Button b1 = new Button ();
    String cap = b1.getlabel ();
    System.out.println (cap) : // Button-0
    b1.setLabel ("ok");
        or
    b1.setActionCommand ("ok");

    Cap= b1.getActionCommand ();
    System.out.println( cap); // ok
```

Methods ⑦ & ⑧ are used for registering / unregistering the events of Button with ActionListener.

* Write a java program to develop the following screen.



The above application can be developed in two ways.

i) By using Frames
ii) By using Applets

i) By using Frames –

```
// ButtonDemo.java
Step:1  import java.awt.*;   (For component Creation)
        import java.awt.event.*;   (For component Functionality)

Step:2 ↓   Step:3   Step:3
Class ButtonDemo extends Frame implements ActionListener
{
Step:4  Button b1, b2, b3, b4;
        Label l1;
        ButtonDemo()
        {
Step:5      setTitle("Button Example");
            setSize(500, 300);
            /* BorderLayout bl = new BorderLayout();
               setLayout(bl); */

Step-6      // Component Creation
            b1 = new Button("OK");
            b2 = new Button("Cancel");
            b3 = new Button("Help");
            b4 = new Button("Exit");
            l1 = new Label();
            l1.setAlignment(Label.CENTER);

Step:7      // Add components to the container
            add(b1, "North");
            add(b2, "South");
```

```
add(l1);

// Registering Components
b1.addActionListener(this);
b2.addActionListener(this);
b3.addActionListener(this);
b4.addActionListener(this);
// Here this represents an object of current class which
// implements ActionListener (I) i.e 'this' is an object
// of ActionListener
setVisible(true);
} // Constructor of ButtonDemo

public void actionPerformed(ActionEvent ae)
{
    String cap = " ";
    if(ae.getSource() == b1)
        cap = ae.getAction b1.getLabel();
    elseif(ae.getSource() == b2)
        cap = b2.getLabel();
    elseif(ae.getSource() == b3)
        cap = b3.getLabel();
    elseif(ae.getSource() == b4)
        cap = b4.getLabel(); // or System.exit(0) or
                             //                dispose();
    l1.setText(cap);
}

Class ButtonMain
{
    public static void main(String args[])
```

Step: 8

Step: 9

Step-10

Date: 14-APR-2011

In the above program -

1) add (Component, String), add (component c) are two pre-defined methods present in Java.awt.Container and these methods are used for adding the specific components to the container.

2) getSource() is one of the pre-defined method present in every pre-defined event classes and it is used for processing any GUI component based on its reference.

* 3) Steps or Guidelines for Developing Stand-alone or Frames Applications -

Step 1: Import java.awt.* (For component creation), java.awt.event.* (For component functionality) and other packages if required

Step 2: Choose user-defined class.

Step 3: The class which is selected in Step-2 must extends Frame and implements appropriate listener if (if required.)

Step 4: Identify the type of GUI components required for building any GUI application

Step 5: Set title, size and layout if required.

Step 6: Create the components which are identified

Step 7: Add the created component to the container.

Step 8: Register the events of GUI active component with appropriate listener.

Step 9: Make the created component to be visible.

Step 10: Define / override an appropriate abstract method which inherited from appropriate listener.

2) By using Applets -

* Steps or Guidelines For developing Distributed GUI / Applet Applications -

Step 1: Import java.awt.*, java.awt.event.*, and java.applet.* (For Life Cycle methods) and other packages if required.

Step 2: Choose user-defined class name.

Step 3: The class which is selected in Step no-2 must extends java.applet.Applet class and implements appropriate listener if required.

Step 4: Identify the type of GUI applications which are required For building GUI applications

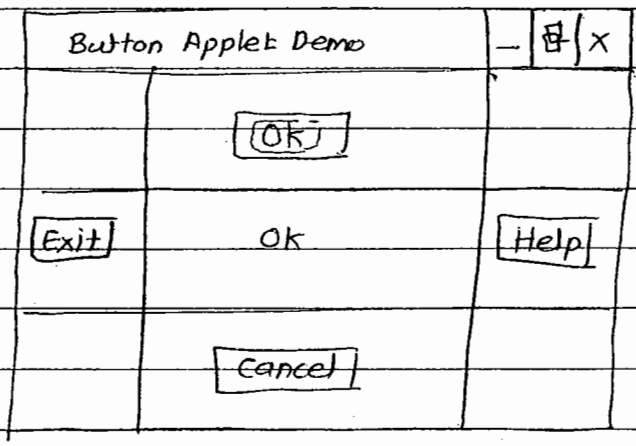
Step 5: Override the public void init method with the Follow
5.1) Set the layout Manager if required
5.2) Create the component which are identified

Step 6: Override the public void start() method with
6.1) Register the events of all GUI active compo-
-nents with appropriate listener.

Step 7: make the created component to be visible (op-
-tional Applets)

Step 8: Define or override the appropriate abstract
method which is inheriting from appropriate
listener

* write a java program which will implement the following screen with Applets



Step-1

```
// ButtonApplet.java
import java.awt.*;
import java.awt.event.*;
import java.applet.Applet;

/*<applet code="ButtonApplet" height=300 width=300>
</applet> */
```

Step-3

Step-4 { Button b1, b2, b3, b4;
label l;

Step-5 → public void init()
{
Borderlayout bl = new Borderlayout();
setlayout(bl);
b1 = new Button("OK");
b2 = new Button("cancel");
b3 = new Button("Help");
b4 = new Button("Exit");
l = new label();
l.setAlignment(label.CENTER);

// Add the components to the container
add(b1, "North");
add(b2, "south");
add(b3, "Fast");
add(b3, "west");
add(l);
} // init()

Step-6 → public void start()
{
// Register components.
Satya so = new Satya();
b1.addActionListener(so);
— " —
— " —
— " —

} // Start

Step-2 → class Satya implements ActionListener // Inner/Nested Class

{

public void actionPerformed (ActionEvent ae)

{

// Same as previous application

}

} // Satya

} // ButtonApplet.

Q] Define Inner or Nested Class & Containership.

→ IF an x class is defined within y class then x class definition is known as inner / nested class.

The process of defining one class definition in another class definition is known as Containership.

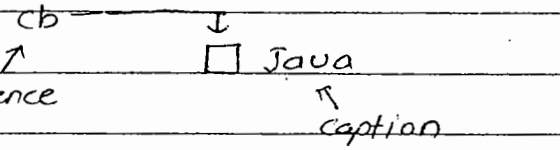
* Checkbox ~~Class~~ ~~Profile~~ Component -

1) Checkbox is one of the active component and hence it participate in Event Delegation model.

2) Creating a checkbox component is nothing but creating an object Checkbox class

ex:

Checkbox cb = new Checkbox ("Java");



The graphical definition of checkbox is collection

The component checkbox will have two states. They are.

i) Unchecked State (False)

ii) Checked state (true)

The default state of the checkbox is unchecked.

3) The realtime implementation of checkbox is that to implement checklist related application

* Checkbox Class Profile -

* Constructors -

1) Checkbox()
2) Checkbox(string)
3) Checkbox(string, boolean)

* Instance methods

4) public void setlabel(String);
5) public ~~void~~ String getlabel();
6) public void setstate(boolean);
7) public boolean getState();
8) public void addItemlistener(ItemEvent);
9) public void removeItemlistener(ItemEvent);

Constructor ① is used to create an object of check box without specifying label & state

Checkbox cb = new Checkbox(); cb →

Constructor ② user for creating an object of checkbox by specifying the label & without specifying state.

Constructor ③ is used for creating an object of checkbox by specifying the label and state.

```
Checkbox cb = new checkbox ("Java", true);
```

cb → ☑ Java

Methods ④ & ⑤ are used for ~~obtai~~ setting & getting the label of the checkbox.

```
Checkbox cb = new Checkbox();
cb.setLabel ("Java");          ☐ Java.
System.out.println ( cb.getLabel()); // Print "java"
```

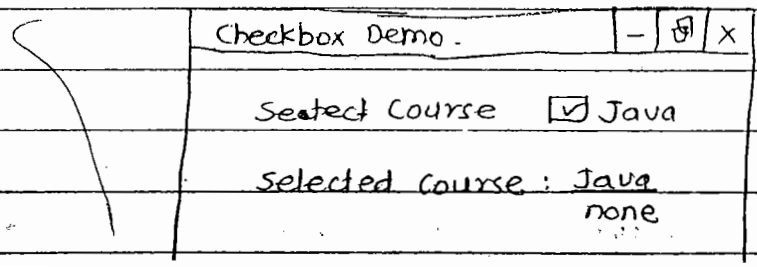
Methods ⑥ & ⑦ are used for setting and getting state of the checkbox.

```
Checkbox cb = new Checkbox ("Java");
boolean state = cb.getState();
  S.O.P ( state); // Print false
  cb.setState ( true);
  S.O.P ( ~~sta~~ cb.getState()); // Prints True.
```

Methods ⑧ & ⑨ are used for registering / un-registering the events of checkbox with ItemListener.

Date: 15-APR-2011

* Write a Java program to implement the following screen.



```
// CheckBoxDemo.java
import java.awt.*;
import java.awt.event.*;

public class CheckBoxDemo extends Frame
{
    Label l1, l2;
    Checkbox cb;

    CheckboxDemo()
    {
        setTitle("Checkbox Demo");
        setSize(200,200);
        setLayout(new FlowLayout());

        // Comp. Creation
        l1 = new Label("Select the course");
        l2 = new Label();
        cb = new Checkbox("Java");

        // add the component to container
        add(l1); add(cb); add(l2);

        //Registration
        cb.addItemListener(new Hyd());
        setVisible(true);

        class Hyd implements ItemListener
        {
            public void itemStateChanged(ItemEvent ie)
            {
```

```
            if( cb.getstate())
            {
                l2.setText("selected Course:" + cb.getlabe());
            }
            else
            {
                l2.setText("selected Course:" NONE");
            }
        }
    }
}

Class checkBoxTest
{
    new CheckboxDemo(); // Nameless Object Approach
}
```

Note: An object of xxxListener can be specified either by current class object or by inner class object

In the above program as & we when the state of the checkbox is changing, automatically itemStateChanged() method of Itemlistener will be called. This work can be done by the concept of registration.

* Radio Buttons ~~Component~~ -

In java.awt.* package there is no component called RadioButton. so there is no pre-defined class for creating ~~an object of~~ a component Radio Button

programmatically by converting the checkbox component

* Converting or obtaining the radio button component from the checkbox component is of three steps. They are

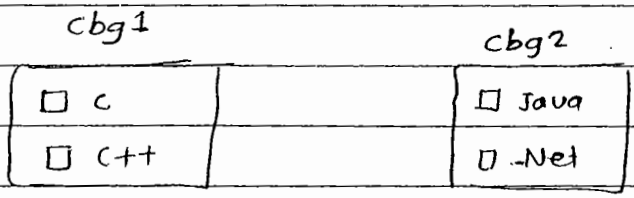
1) Create no. of Checkbox class objects. & identify their groups

Ex. Checkbox cb1 = new checkbox ("C"); } Group 1
Checkbox cb2 = new checkbox ("C++"); }
Checkbox cb3 = new Checkbox ("Java"); } Group 2
Checkbox cb4 = new Checkbox (".Net"); }

2) To select one from the group or two select one checkbox from group of multiple checkboxes we need to use a pre-defined class called java.awt.CheckBoxGroup class create no. of checkbox group class objects depends on no. of groups identified in Step no. 1.

Ex:

CheckboxGroup cbg1 = new CheckboxGroup();
CheckboxGroup cbg2 = new CheckboxGroup();



3) Add the corresponding checkboxes ~~two~~ objects to the CheckboxGroup class object. To do this we use the following method.

java.awt.checkbox

public void setCheckboxGroup (checkboxGroup);

Ex. cb1.setCheckboxGroup (cbg1);
cb2.setCheckboxGroup (cbg2);

Cbg1	Cbg2
O C	O Java
O C++	O .Net

The event delegation model for the radio button is similar to Checkbox.

* // Java Program to implement the above concept

```
import java.awt.*;
import java.awt.event.*;
import java.applet.Applet;

/*< applet code = "rdapp" width=500 height=300 >
</applet> */

public class rdapp extends Applet
{
    Label l1, l2;
    Checkbox cb1, cb2, cb3, c4;
    CheckboxGroup cbg1, cb2;

    pubit ~~At~~ void init()
    {
        l2 = new Label();

        cb1 = new cheekbox ("C");
        cb2 = new Checkbox ("C++");
        Cb3 = new checkbox ("Java");
```

```
// Step2
cb1.setCheckboxGroup(cbg1);
cb2.setCheckboxGroup(cbg1);
cb3.setCheckboxGroup(cbg2);
cb4.setCheckboxGroup(cbg2);

// Add comp to the container
add(cb1); add(cb2); add(cb3); add(cb4);
add(l2);

public void start()
{
    // Registration
    cb1.addItemListener(new Hyd());
    cb2.addItemListener(new Hyd());
    cb3.addItemListener(new Hyd());
    cb4.addItemListener(new Hyd());
}

class Hyd implements ItemListener
{
    public void itemStateChanged(ItemEvent ie)
    {
        Object Checkbox cb = ie.getItemselectible();
        if(cb.getState())
        {
            l2.setText("Select Courses:" + cb.getLabel());
        }
    } // Hyd
} // main
} // xdapp
```

Date: 16-APR-2011

* Define object Type Casting ? Explain its necessity ?

→ The process of obtaining an address of base class and converting into subclass object. is known as Object Type Casting.

* Necessity -

i) The result of the Java program is available in the object of java.lang.Object class

ii) To process the result of object of java.lang.Object, we are not having any special methods present in java.lang.Object.

iii) To process the result of object of java.lang.Object, we are having the special methods present in subclass java.lang.object.

iv) we know that an object of java.lang.Object does not access the special methods of its subclass.

v) To process the result of object of java.lang.Object. with respect to the special methods of derived class of java.lang.Object, we require the concept of Object Type Casting.

* Syntax: -

```
<subclass name> subobj = (<subclassname>) SuperObj ;
```

* Example -

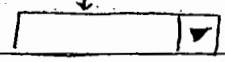
```
Object obj = ["satya"];
    // obj.length(); Invalid.
    |
    v
String str = (String) obj;
S.o.p(str.length()); // Prints : 5
```

* Choice Component -

i) Choice is one of the active component. So that it participates in Event Delegation Model.

ii)

iii) creating a choice component is nothing but creating an object of choice class.

ex. Choice ch = new Choice(); ch → 

The component choice contains only reference but it is not containing any caption.

iii) The functionality of choice component is to select only one Item from multiple items.

iv) The component choice allows us to add the data only in the form of string.

* Choice Class Profile -

* Constructor -

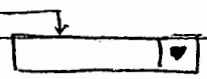
1) Choice()

* Instance Methods -

2) public void add(String);
3) Public void addItem(String);
4) public void add(String, int);
5) public void addItem(String, int);
6) public ~~void~~ String remove(int);
7) public ~~void~~ String remove(String);
8) public void removeAll();
9) public String getSelectedItem();
10) public ~~void~~ int getSelectedIndex();
11) public void addItemListener(ItemEvent)

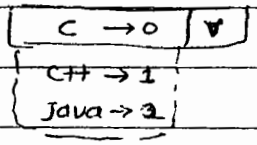
Constructor ① is used for creating an object of Choice class.

Ex. Choice ch = new Choice();



Methods ② & ③ are used for adding the elements to the choice component only at the end.

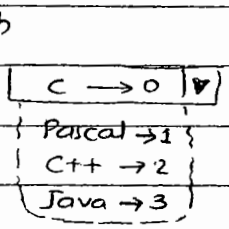
Ex. ch.add("C");
ch.addItem("C++");
ch.add("Java");
~~ch.add~~



Internally, all the elements of choice are represented or processed by the concept of 0-based indexing like Arrays

Methods ④ & ⑤ are used for adding an element to the choice at a specific position

Ex. ch.add("Pascal", 1) or
ch.addItem("Pascal", 1)



Method ⑥ & ⑦ are used for removing the elements of choice either on the bases of position or on the content when we are removing any element of choice on the basis of content, that element will be removed in the case of first occurrence and meaning & case must be same.

Ex: ~~ch.remove~~

String s = ch.remove(1); // Based on Position
System.out.println(s); //

or

ch.remove("Java"); // Based on content

Method ⑧ is used for removing all the entries of Choice component

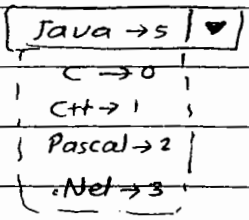
Ex:

```
ch.removeAll();
```

Method ⑨ is used for obtaining or selecting an Item at runtime.

Ex.

```
String str = ch.getSelectedItem();
System.out.println(str); // Print: Java
```



Method ⑩ is used for obtaining an index or position of that item which is selected at run time

Ex. `int index = ch.getSelectedIndex();`

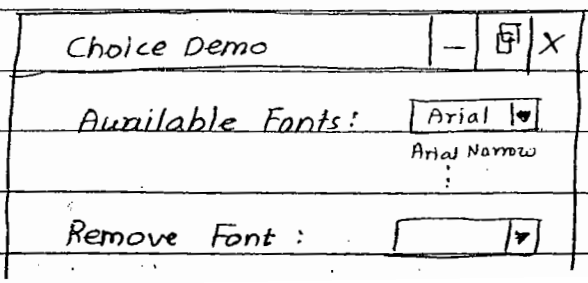
`System.out.println(index); // Prints : 5`

* Methods ⑪ & ⑫ are used for registering / unregistering the events of Choice component with ItemListener.

In other words, as and when an item of the choice component is selected automatically itemStateChanged() method of ItemListener will be called. This can be done by the concept of registration.

* Write a Java Program to implement the following Screen-

→



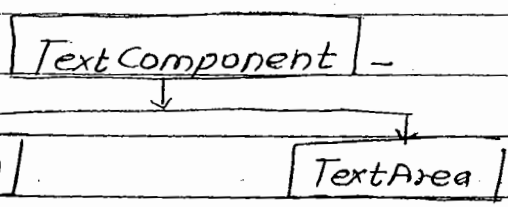
```
// ChoiceDemo
import java.awt.*;
import java.awt.event.*;

public class ChoiceDemo extends Frame
{
    label l1, l2;
    Choice ch1, ch2;
    public ChoiceDemo()
    {
        super("ChoiceDemo");
        setsize(500,300);
        setLayout(new FlowLayout());

        // Create the components
        l1 = new label("Available Fonts:");
        ch12 = new label("Removed Fonts:");
        ch1 = new choice();
        ch2 = new choice();

        // Add the components to the container.
        add(l1); add(ch1); add(l2); add(ch2);

        // Registering Event Listener.
        ch1.addItemListener(new Hyd());
        setVisible(true);

        GraphicsEnvironment ge = GraphicsEnvironmet.getLocal
                                 GraphicsEnvironment();
        String f[] = ge.getAvailableFontFamilyNames();
        for(int i=0; i< f.length; i++)
        {
            ch.add(f[i]); // Adding Fonts names to Choice
```

```
class Hyd implements ItemListener
{
    public void itemStateChanged(ItemEvent ie)
    {
        if( ie.getSource() == ch1)
        {
            String s = ch1.getSelectedItem();
            ch2.add(s);
            ch1.remove(s);
        }//if
    }//itemStateChanged
}//Hyd

public class ChTest
{
    public static void main(String args[])
    {
        new ChoiceDemo();
    }//main
}//ChTest.
```

Date - 18-APR-2011

* Text Component -

Text Component is one of the predefined super abstract class for both TextField & TextArea.



The component is called TextComponent allows the programmer to enter the input in GUI component either in single line of text or multiple lines of text.

* Methods -

1) public void setText (String) ;
2) public string getText ( ) ;

Method ② is used for obtaining the data from TextField or TextArea.

method ① is used for setting the string data in TextField or TextArea.

The above two methods are inherited into both TextField & TextArea.

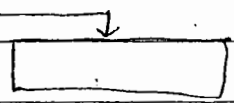
* TextField Component -

i) TextField is one of the active component and hence it participates in Event Delegation Model.

ii) Creating a TextField is nothing but a creating object on TextField Class

Ex:

```
TextField tf = new TextField ( ) ;
```



Here,

TextField Contains only reference but not caption.

iii) whatever data we enter into TextField which is by

iii) The functionality of TextField component to enter only one line of Text as input by the normal user.

iv) TextFeield component classified into ~~type~~ two types They are -

i) Non-echo TextField.

ii) Echo TextField.

A non-echo TextField is one which allows us to understand the content of the TextField.

Ex. Username : [Sachin]

An echo TextField is one whose content is not able to be understandable by other application users

Ex. Password [******]

Here, '*' is called as Echo-Character

All the echo TextFields are nothing but Password Fields.

vi) Whenever we create a TextField which is by default non-echo and one can programatically converts a non-echo TextField into Echo TextField.

* TextField Class Profile -

* Constructors -

① TextField ()

2) TextField (int)

3) TextField (String)

* Instance methods -

6) public void setText (string);
7) public string getText();
8) public void setEchoChar(char);
9) public char getEchoChar();
10) public boolean getsetEchoCharIsSet();
11) Pubic void addTextListener(~~Item~~ TextEvent);
12) pubic void removeTextListener(TextEven);

Constructor ① is used for creating an object of the TextField without specifying ~~it size~~ its size.
ex: Texfield tf = new TextField();
The default size of the TextField is 10 pixels.

Constructor ② is used for creating an object of TextField by specifying its size in terms of pixels.
Ex: TextField tf = new TextField(50);

Constructor ③ is used For creating an object of TextField by specifying the text data. This is useful to provide the fixed data like Company names, their addresses, system date etc.
ex: TextField tf1 = new TextField("Satya Technology")

tf → "Satya Technology"

Methods ④ & ⑤ are used for setting & getting the size of the TextField.
Ex. TextField tf1 = new TextField();
int size = tf1.getsize();
S.O.P (size); // 10
tf1.setSize (size+10);

Methods ⑧ & ⑨ are used for setting & getting the echo character to the TextField. Specially method ⑧ is used for converting non-echo TextField into echo TextField.

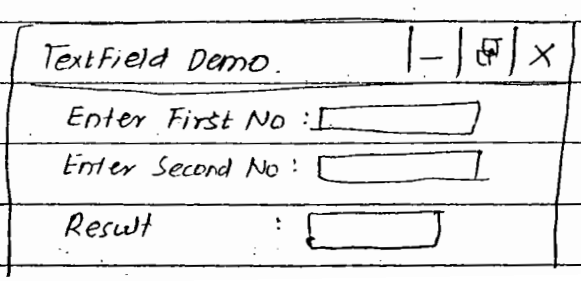
Method ⑩ returns true provided TextField belongs to Echo otherwise it returns false.

Ex.
```
TextField tf1 = new TextField();
tf1.setEchoChar('*');

TextField tf2 = new TextField();

S.O.P("TextField Is Echo : " + tf1.setEchoCharIsSet());
// Prints Is Echo : True

S.O.P("Is Echo :" + tf2.setEchoCharIsSet());
// Prints Is Echo : False.
```

methods ⑪ ⑫ & ⑫ are used for registering the event of ~~TextEvent~~ TextField / TextArea with TextListener

In other words, as & when the value of the TextField is changing, automatically textValueChanged(-) method of TextListener must be called. To due this we must go for registration process.

* write a Java Program to implement the following screen



```
import java.awt.*;
import java.awt.event.*;
class TextFieldDemo extends Frame
{
    Label l1, l2, l3;
    TextField tf1, tf2, tf3;
    Button b1, b2, b3, b4;

    TextFieldDemo()
    {
        setTitle("TextField Demo");
        setSize(500, 300);
        setLayout(new FlowLayout());

        l1 = new Label("Enter First No.");
        l2 = new Label("Enter Second No.");
        l3 = new Label("Enter Result :");

        tf1 = new TextField();
        tf2 = new TextField();
        tf3 = new TextField();

        b1 = new Button("Add");
        b2 = new Button("Sub");
        b3 = new Button("Multi");
        b4 = new Button("Exit");

        add(l1); add(tf1); add
        add(l2); add(tf2);
        add(l3); add(tf3);
        add(b1); add(b2); add(b3); add(b4);

        b1.addActionListener(new AL());
```

```
        b3.addActionListener(new AL());
        b4.addActionListener(new AL());
        setVisible(true);
    } // constructor

    class AL implements ActionListener
    {
        public void actionPerformed(ActionEvent ae)
        {
            int a = Integer.parseInt(tf1.getText());
            int b = Integer.parseInt(tf2.getText());
            int z = 0;
            if(ae.getSource() == b1)
                z = a+b;
            else if(ae.getsource() == b2)
                z = a-b;
            else if(age.getSource() == b3)
                z = a*b;
            else if(ae.getSource() == b4)
                System.exit(0); // dispose();
            tf3.setText(~~z~~; String.valueOf(z));
        }
    } // AL

} // TextFieldDemo

public class TextFieldTest
{
    public static void main(String args[])
    {
        new TextFieldDemo();
```

In the above program.

System.exit(0) is used for terminating or closing the window at runtime.

dispose() ; it is one of the pre-defined method present in window class and it is used for closing the window at runtime.

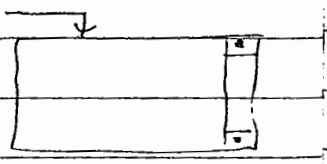
Date: 19-APR-2011

* TextArea -Component -

i) TextArea is one of the active component & hence it participates in Event Delegation model.

ii) Creating TextArea is nothing but creating an object of TextArea class.

Ex: TextArea ta = new TextArea();



iii) TextArea component is having only reference but there is no caption

iv) The Functionality of TextArea Component is to accept multiple lines of Text.

v) Graphically, TextArea is the collection of Rows & columns.

vi) When the no. of rows are exceeding than the specified then vertical scrollbar will be enable & it is used for to view hidden no. of rows

Similarly, when the no. of columns are exeeeding than the specified, automatically horizontal scroll

Whenever we create a TextArea component, by default vertical scrollbar will be appended & it is in disabled state.

Whatever data we entering to TextArea, that data is by default treated as String data.

* TextArea Class Profile -

* Data members -

public static final int SCROLLBARS_NONE;
public static final int SCROLLBARS_BOTH;
public static final int SCROLLBARS_HORIZONTAL_ONLY;
public static final int SCROLLBARS_VERTICAL_ONLY;

The above data members are known as ScrollBar visibility modifiers and they must be accessed with respect to the class name.

The default scrollbar visibility modifier is SCROLLBARS_VERTICAL_ONLY.

* Constructors -

① TextArea()
② TextArea(int, int)
③ TextArea(int, String)
④ TextArea(int, int, int)

* Instance methods -

⑤ public void getRows(int)
⑥ public void setCols(int)
⑦ public int getRows()
⑧ public int getCols()
⑨ public void setScrollBarVisibility(int)

(12) public void addTextListener(TextEvent)

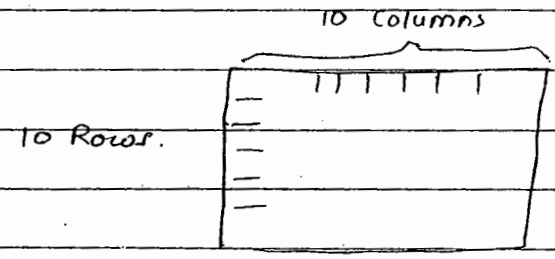
(13) public remove removeTextListener(TextEvent)

Constructor (1) is used for creating an object of TextArea without specifying no. of rows & columns.

Ex TextArea ta = new TextArea();

Constructor (2) is used for creating an object of TextArea by specifying no. of rows & columns.

Ex. TextArea ta = new TextArea(10,10);



Constructor (3) is used for creating an object of TextArea by specifying the fixed content like company addresses, landmarks of the companies etc.

Ex TextArea ta = new TextArea("Satya Technology");

Constructor (4) is used for creating an object of TextArea by specifying the no. of rows, columns and Scroll Bar visibility modifier value.

Ex. TextArea ta = new TextArea(10,10, TextArea.
SCROLLBARS_BOTH);

methods (5) & (6) are used for setting the number of Rows & columns.

Ex. TextArea ta = new TextArea();

Methods ⑦ & ⑧ are used for obtaining no. of rows & columns.

Ex: ~~TextArea~~

```
int rs = ta.getRows();
int cs = ta.getCols();
S.OP (rs); // Prints 10
SOP (cs); // Prints 10
```

Method ⑨ is used for setting the scrollbar visibility modifier to the TextArea component.

Ex. `TextArea ta = TextArea();`
`ta.setScrollBarVisibility (TextArea.SCROLLBARS_BOTH);`

Method ⑩ returns true, provided the programmer has set ScrollBarVisibility modifier value. It returns false provided the programmer is not changing the existing SCRO-LLBAR visibity ~~bar~~ value.

Ex. `TextArea ta = new TextArea();`
`boolean b = ta.setScrollBarVisibilityIsSet();`
`SOP (b); // false`
`ta.setScrollBarVisibility (TextArea.SCROLLBARS_NONE);`
`b = ta.setScrollBarVisibilityIsSet ();`
`SOP(b);`

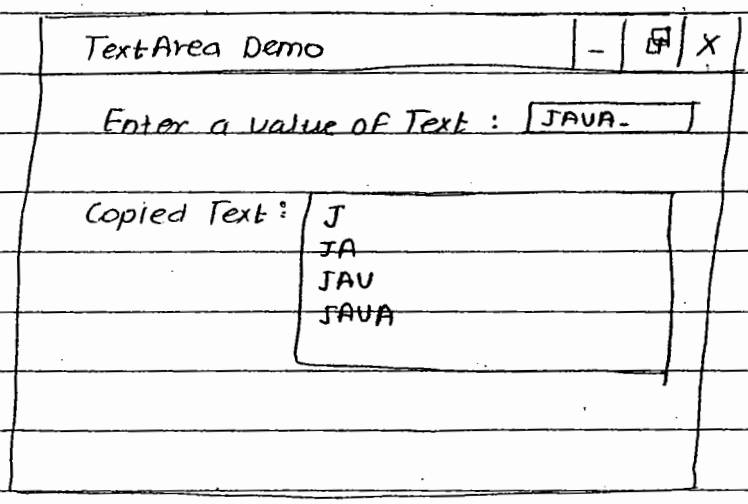
Method ⑪ is used for adding some new data at the end of existing data of TextArea component. Whereas setText (-) method is replacing the data of TextArea with new data.

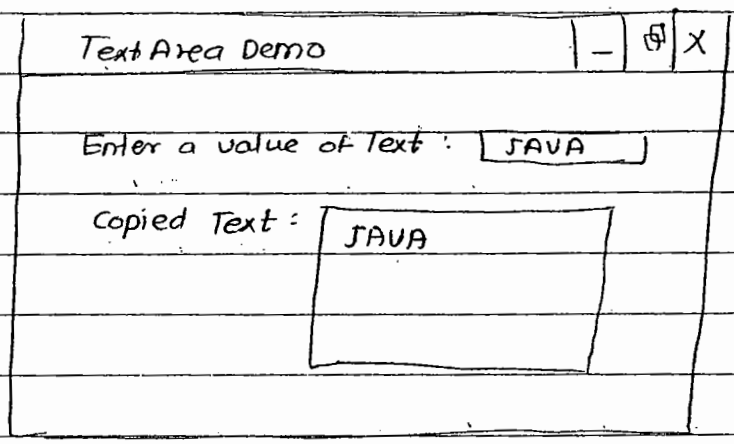
Ex. `TextArea ta = new TextArea ("Sathya");`
`SOP (ta.getText()); //Prints : Sathya`
`ta.append (" Technology");`
`SOP (ta.getText()); // Prints: Sathya Technology`

Methods (12) & (13) are used for registering / unregi-stering the events of TextArea / TextField with TextListener

In other words, as & when the value of TextArea is changing, automatically textValueChanged Method of TextListener will be called.

* write a Java Program to implement the following Screens






```
import java.awt.*;
import java.awt.event.*;

class TextAreaDemo extends Frame
{
    Label l1, l2;
```

```
gi-
:ner
        TextAreaDemo()
        {
            setTitle("TextArea Demo");
            setSize(500, 300);
            setLayout(new FlowLayout());

            Label l1 = new Label("Enter a value of Text:");
is          l2 = new Label("Copied Text:");
            ta = new TextArea();
            t1 = new TextField();
            add(l1); add(t1);
            add(l2); add(ta);

            t1.addTextListener(new TL());
            setVisible(true);
        } // Constructor

        class TL implements TextListener
        {
            public void textValueChange(TextEvent te)
            {
                ta.append(t1.getText() + "\n");
            }   // ta.setText(t1.getText());
        } // TL
    } // TextAreaDemo

    public class TextAreaTest
    {
        public static void main(String args[])
        {
            new TextAreaDemo();
        } // main
    } // Test
```

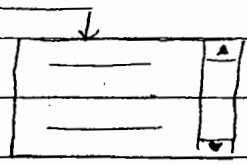
Date- 20-APR-2011

* List Component -

i) List is one of the active component so that it participates in event delegation model.

ii) Creating list is nothing but creating an objects of list class.

Ex. List l = new List();



iii) When we create list component by default 3 items will be visible out of the multiple items. But we can change it programatically the no. of items to be visible.

iv) The functionality of List component is to select multiple items from multiple items.

v) Bydefault the component list allows us to select only one item from multiple items. (known as single item selection). Programmatically one can convert single item selection to multiple item selection

vi) Whatever data we add to list component those items must belongs to string type.

vii) All the elements of the list can be processed based on zero based indexing concept (whose positions start from 0 to n-1)

Q] What is difference between choice & list components?

→ The component choice allows us to select only one item from multiple items.

The component list allows us to select either one or multiple from multiple items.

* List Class Profile -

* Constructors -

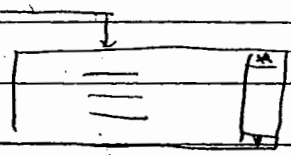
1) List ()
2) List (int vsize)
3) List (int vsize, boolean)

* Instance methods -

4) public void add (String)
5) public void addItem (string)
6) public void add (~~in~~ String, int)
7) public void addItem (string, int)
8) pubic string remove (int)
9) public void remove (string)
10) Public void removeAll ()
11) public string getSelectedItem ():
12) public string[] getSelectedItems ()
13) public int getSelectedIndex()
14) public int[] getSelectedIndexes ()
15) public void setMultipleMode (boolean)
16) public boolean ~~get~~setMultipleMode ~~Is~~set ().
17) public void addItemlistener (ItemEventlistener)
18) public void removeItemListener (ItemListener)
19) public void addActionlistener (Action Listener)
20) public void removeActionlistener (ActionListener)

Constructor ① is used for creating an object of list without specifying the no. of items to be visible

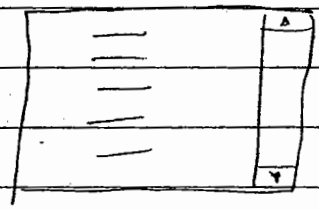
Ex. List l = new List ();



Constructor ② is used for creating an object of list by specifying no of items to be visible.

In general, if we specifies the value for vsize then at runtime we get vsize-1 element will be visible and other multiple items will be hidden and they can be viewed by vertical scrollbar.

Ex: List l = new List(6);



Here, 5 elements (6-1) will be visible out of 6 multiple values

Constructors ① & ② allows us to select single item from multiple items.

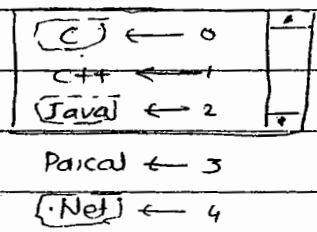
Constructor ③ is used for allows us to select multiple items from the list component by passing true for boolean parameter.

Ex. List l = new List(6, true);

In other words, constructor is used for converting single item selection to multiple item selection.

Method (12) is used for selecting multiple items from list component provided List must support multiple item selection.

Ex. List l = new List(4, true);
String s[] = l.getselectedItems();




```
for(int i=0; i<s.length; i++)
{   sop(s[i]); SOP(s[i]);
}
```

This method is quite advisable to the java programmer to select the items from list component whether it is belong to single item selection or multiple item selection.

Method (14) is used for selecting the indexes of those

items which are selected at run-time from list component.

Ex: int arr[] = l.getSelectedIndexes();
for (int i=0; i<arr.length; i++)
{
SOP(arr[i]);
}

Method (15) is used for converting single item selection to multiple item selection by passing true for the boolean parameter.

Method (16) is used for checking whether the list component is single mode selection or multiple mode selection. This methods returns true provided list component belongs to multiple item selection. Returns false provided list components belongs to single item selection.

Ex. List l = new List();
l.setMultipleMode(true);
List l2 = new List();

boolean b = l.setMultipleModeIsSet();
SOP(b); // Prints: true
b = l2.setMultipleModeIsSet();
SOP(b); // Prints: false

Methods (17) & (18) are used for registering / unregistering the events of List component with ItemListener provided the items of the list selected with single click. In other words as and when items of the list selected with the single click, automatically itemstate changed() ItemListener will be called.

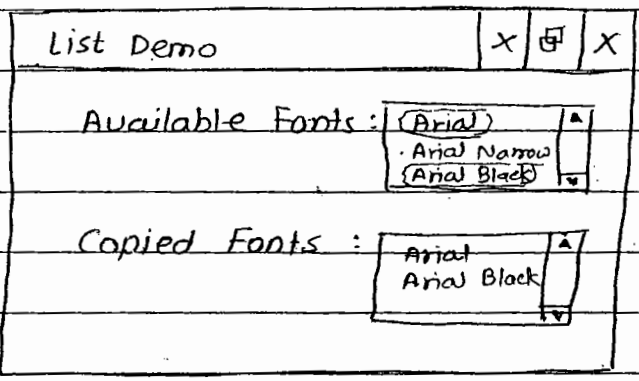
Method (19) & (20) are used for registering / un-registering the events of list component with Action-Listener provided items of the list will be selected with double click. In other words, as & when the items of the list selected with double click, automatically actionPerFormed() of ActionListener will be called

The following diagram gives the type se of selection and associated click. for seleeting items from list.

Type of selection / Type of click	Single Item Selection	Multiple Item selection
Single Click	boolean = false Listener = IL	boolean = true Listener = IL
Double Click	boolean = ~~True~~ False Listener = AL	boolean : true Listener : AL

Date: 21-APR-2011

* Write a Java Program to implement the following Screen




```
// ListDemo.java
import ~~awt~~.java.awt.*;
import java.awt.event.*;
```

```
class ListDemo extends Frame
{
    Label l1, l2;
    List li;
    TextArea ta1;

    ListDemo()
    {
        super("List Example");
        setSize(500,300);
        setLayout(new FlowLayout());

        // Component Creation
        l1 = new Label("Available Fonts");
        l2 = new Label("Copied Fonts");
        li = new List(6); // 5 Items will be visible (6-1)
        ta1 = new TextArea();
        li.setMultipleMode(true);

        // Get the System Fonts and add to li object
        GraphicsEnvironment ge = GraphicsEnvironment.get
                                 LocalGraphicsEnvironment();
        String F[] = ge.getAvailableFontFamilyNames();
        For(int i=0; i< f.length; i++)
        {
            li.add(F[i]);
        }

        // Add components to Container
        add(l1); add(li);
        add(l2); add(ta1);

        // Register Event Listener.
```

```
        setVisible(true);
    }// constructor
}// ListDemo
    class itl implements ItemListener
    {
        public void itemStateChanged(ItemEvent ie)
        {
            if(ie.getSource() == li)
            {
                String s[] = li.getSelectedItems();
                for(int i=0; i<s.length; i++)
                {
                    ta1.append(s[i] + "\n");
                }
            }
        }
    }// itl
}// ListDemo


// ListDemoTest.java
public class ListDemoTest
{
    public static void main(String args[])
    {
        new ListDemo();
    }
}
```

* Adapter Classes *

In the ~~exti~~ earlier versions of Java, if an interface is containing n- number of Abstract methods and if the derived class is implementing that interface then the derived class programmer must define all n-number of abstract methods eventhough the derived class programmer is interested in only one abstract method. And moreover defining uninterested methods leads to waste s/w development time. This is one of the drawback of interfaces.

To eliminate this drawback, in the current version of Java we have a concept called Adapter Classes

The Basic Advantage Adapter Class is that to define only the interested method by leaving uninterested methods.

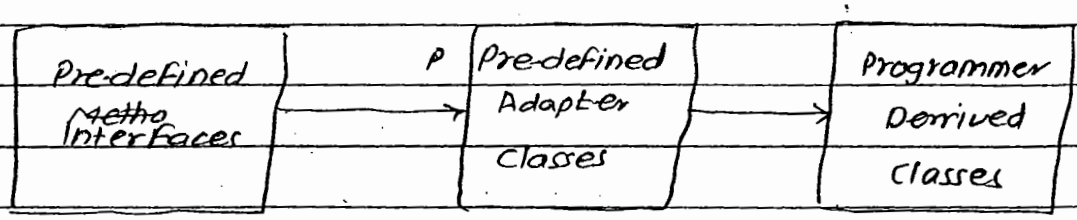
This results in minimization of s/w development time.

* Definition -

An Adapter Class is one which contains null body of methods definition for those abstract methods which are inheriting from predefined interfaces.

Or

Adapter classes acts as a middle man role between predefined interfaces and programmer derived classes



Since, all pre-defined adapter classes are containing null body definition For the Abstract methods of Pre-defined Interfaces, all the pre-defined adapter classes must be abstract.

All the pre-defined Adapter classes in java are reusable.

Whichever pre-defined interface if contains more than one abstract methods, For such interfaces there exist pre-defined adapter classes

In windows Programming iterfaces are known as Listeners and whose generalised notation is xxxListener. whichever xxxListener contains more than one Abstract method then there exist a pre-defined @Adapter Class and whose generalised notation is XxxAdapter.

In window programming all the pre-defined Adapter classes are present in the predefined package called java.awt.event.* package. The following table gives XXXListener and corresponding XxxAdapter.

XXXListener	XXXAdapter
ActionListener	-
WindowListener	WindowAdapter
MouseListener	MouseAdapter
MouseMotionListener	MouseMotionAdapter.

Note: Window Event Delegation Mode.

⇒ Component - Window

⇒ XXXEvent - WindowEvent

⇒ xxx Listener – windowListener
windowAdapter

⇒ Registration Method – public void addWindowListener (windowEvent)

⇒ Unregistration Method – public void removeWindow Listener (windowEvent)

* Write a program which will close the window of the Frame application

→ // WindowDemo.java

```
import java.awt.*;
import java.awt.event.*;

public class MyWindow extends Frame implements
{
    MyWindow()
    {
        setTitle("My Window Demo");
        setSize(500, 300);
        setVisible(true);

        addWindowListener(new MyAdapter());
    }
    class MyAdapter extends WindowAdapter
    {
        public void windowClosing(WindowEvent we)
        {
            System.exit(0);
        }
    } // Inner class
```

```
public class MyWindowTest
{
    public static void main(String arg[])
    {
        new MyWindow();
    } //main
} // MyWindowTest
```

Date: 22-APR-2011

* Mouse Handling Operations *

In windows programming of Java we have two types of components. They are

i) Logical Components
ii) Auxiliary Components

Logical Component are thoses which are having "look & feel" approach. In other words, all the s/w oriented components are comes under logical components.

Ex. Label, Button, CheckBox etc.

Auxiliary components are those which are having the "Touch & Feel" Aproach. In other words, all the hardware components which are used for interacting with GUI application is known as @Auxiliary Components.

Ex: Mouse, keyboard, etc.

Auxiliary Components does not contain any Predefined classes whereas logical components contains predefined classes.

Whatever the operations we do with mouse component, those operations are known as Mouse Handling Operations.

Mouse Handling operations are classified into two types.

i) Basic operations of Mouse
ii) Advanced operations of Mouse.

* Basic operations of Mouse -

Basic operations of mouse are classified into five types. They are

i) MouseClicked
ii) Mouse Released
iii) Mouse Preseed
iv) Mouse Entered
v) Mouse Existed

To perform the above operations of mouse, we have a pre-defined listener called java.awt.event.MouseListener.(I)

* methods in MouseListener -

i) public void mouseClicked (MouseEvent)
ii) public void mouseReleased (MouseEvent)
iii) public void mousePressed (MouseEvent)
iv) Public void mouseEntered (MouseEvent)
v) public void mouseExited (MouseEvent)

Since MouseListener is containing more than one abstract method there exist a pre-predefined Adapter class called java.awt.event.MouseAdapter. This class contains null body methods definitions for the abstract methods of mouse Listener.

Advanced Operations of Mouse -

Advance operations of mouse are divided into two types. They are -

i) Mouse Dragged

ii) mouse moved

To perform the advanced operations of mouse, we have pre-defined interface called java.awt.event.MouseMotionListener. (I)

* Methods in MouseMotionListener -

i) public void mouseDragged (MouseEvent)

ii) ~~Pub~~lic void mouseMoved (MouseEvent)

Since mouseMotionListener is containing more than one abstract method, there exist a pre-defined Adapter class called java.awt.event.MouseMotionAdapter. This class contains null body methods for the abstract methods of MouseMotionListener.

* Event Delegation Model For MouseComponent -

Component Name : Mouse (Auxiliary Component)

XxxEvent : MouseEvent

XxxListener : MouseListener
MouseMotionListener

xxxAdapter class : MouseAdapter
MouseMotionAdapter

Abstract methods : 5 Abs. methods (Basic Operations)
2 Abs. methods (Adv. Operations)

Registration Methods : public void addMouseListener (MouseEvent) (ML)
public void addMouseMotionListener (MouseEvent) (MML)

Unregistration methods: public void removeMouseListener (MouseListener)
: public void removeMouseMotion Listener (MouseMotionListener)

The above registration & unregistration method are present in java.awt.Component class.

* Example Application—

```
// MouseOp.java
import java.awt.*;
import java.awt.event.*;
import java.apple.Applet;
/* <applet code="MouseOp" width=500 height=300 /> */
public class MouseOp extends Applet
{
    pub int x, y;
    String op;
    public void init()
    {
        x=100;
        y=100;
        op=null;
        setBackground(Color.cyan);
        setForeground(Color.red);
    }
    public void start()
    {
        addMouseListener(new ML());
        addMouseMotionListener(new MML());
    }
```

```
public void paint(Graphics g)
{
    Font f = new Font("Arial", Font.BOLD, 50);
    setFont(f);
    g.drawString(op+"("+x+","+y+")", x, y);
}

class m1 extends MouseAdapter
{
    public void mouseEntered(MouseEvent me)
    {
        showStatus("Mouse Entered in the window");
    }
    public void mouseExited(MouseEvent me)
    {
        showStatus("Mouse Exited from window");
    }
}// ~~mt~~
    public void mouseClicked(MouseEvent me)
    {
        ~~showstatus~~(
        op = "mouse clicked";
        x = me.getX();
        y = me.getY();
        repaint();
    }
public void mousePressed(MouseEvent me)
{
    op = "Mouse Pressed");
    x = me.getX();
    y = me.getY();
    repaint();
}
```

```
public void mouseReleased(MouseEvent me)
{
    op="Mouse Released";
    x = me.getX();
    y = me.getY();
    repaint();
}
} // ml

class mml extends MouseMotionAdapter
{
    public void mouseDragged(MouseMotionEvent me)
    {
        op="Mouse Dragged";
        x = me.getX();
        y = me.getY();
        repaint();
    }
    public void mouseMoved(MouseEvent me)
    {
        op="mouse moved";
        x = me.getX();
        y = me.getY();
        repaint();
    }
} mml
} // MouseOp
```

In the above program -

1) public void setBackground(int) }
   public void setForeground(int) }
   are the two pre-defined methods present java.awt.

colors. The paramenter int represents the color number. In java programming all the colors are available in java.awt.Color class as a static data members.

2) Pubic void showStatus(String) is one of the pre-defined method present in Applet class and it is used for changing the status bar message of the Applet. Here string parameter represent name of the message to be appearing on statuse bar of the applet.

3) public void repaint() is one of the pre-defined method present in Applet class and it is used for calling the paint method again & again.

4) public int getX() & public int getY() are the two predefined methods present in a pre-defined class called MouseEvent and these methods are used for obtaining x co-ordinating & y co-ordinates where the mouse pointer is waiting.

* Disadvantages of Applets -

As on today, if we develop any distributed appln with the concepts of Applets, we may get following drawbacks

i) When we run / access the server-side application, in the client-side which was developed in applets, there is a possibity of bringing virus from server over the nlw so that client appln are unable to run.

ii) Applets are not fully secured from an unauthorised

Date: 23-APR-2011

* Collection Framework *

(java.util.*)

Collection Framework is one of the distinct facility for strengthening the performance of java and J2EE applications.

In another words, Collection Framework is one of the additional service or add-on service to the library of java for enhancing the performance of java, J2EE applications. In order to deal with Collection Framework programming we must import the pre-defined package called java.util.*.

* Definition -

Collection Framework is the standardised mechanism of java which allows us to group or integrate multiple values which are belongs to either same type or different type or both with dynamic size in a single variable. This single variable is known as Collection Framework variable.

Q] What are the difference between arrays and collection Framework ?

Arrays	Collection Framework
i) An array is collective name given to group of consecutive memory locations which are all refered by similar type of values	i) A collection framework variable is one which allows us to group both similar type of values and different type of values.
ii) Concept arrays always allows to place Homogeneous values.	ii) Collection Framework concepts allows us to place both homogeneous

iii) Concept of arrays contains fixed size in nature.	iii) The concept of Collection framework contains dynamic size in nature.

* Goals / Aim / Properties / Charactestics of Collection Frame works -

When we use the concept of collection Framework, in our Application Development, we may achieve the following goals. -

1) Performance of java applications are enhanced.

2) Adaptability is acheived (the process of adding the content of one Collection Framework variable to another Collection Framework variable either at the end or at the specific position).

3) Collection Framework provides extendability.

4) Collection Framework is one of the Algorithmic oriented.
   a) Collection Framework contains in-built sorting techniques.
   b) Collection Framework contains in-built searching techniques.
   c) Collection Framework contains preliminary data structures, like stack, Linked list etc.

* Types of Collection Framework -

Collection Frameworks are classified into two types. They are

i) New Collection Framework.

ii) Legacy Collection Framework.

1) New Collection Framework -

In earlier days of Sun microsystems the concept of Collection Framework was known as Data structures. The concept of Data structures was fullfi-lling some of the requirements of the industry and it was unable to be fulfilled some other require-ments of the industry. This is one of the limitation of data structures concept of the java. Hence the data structures concept of Java re-engineered by sun microsystems and released to the industry on the name of New Collection Framework.

In other words revised form of Data structures is known as New Collection Framework.

Later on, the existing data structures concept is renamed as Legacy Collection Framework.

Date: 24-APR-2011

* Types of New Collection Framework.

New Collection Framework is classified into two types They are -

1) 1-D / Single Collection Framework -

2) 2-D / Double Collection Framework / Maps -

1-D Collection Framework is one which is organising the data in a single variable either in the form of row or in

the form of column but not in the form of (key, value) pair.

Ex. ⇒ Grouping set of names of a class.

⇒ Grouping set of salaries paid by the company to emp.

etc.

A 2-D collection Framework is one in which the data is organising in a single variable in the form of (key, value) pair.

In (key, value) pair, the value of 'key' must be unique or distinct and value of 'value' may or may not be unique. (Duplicates)

Ex. ⇒ Organising student no., name in the attendance register

⇒ organising account no., customer name in the deposit register of a Bank

⇒ Organing phone no, name in Phonebook of the mobile etc.

Studunt Attendance Register, Deposit registe, phone-book are the examples of 2-D collection Framework variable.

In order to deal with new Collection Framework we must learn the Collection of Classes and the interfaces which are present is java.util.* package

* 1-D / Single Collection Framework -

we know that 1-D collection Framework is the one which is organising the data either in the form of row or in the form of column. But not in the form of (key, value).

In order to deal with single collection Frame

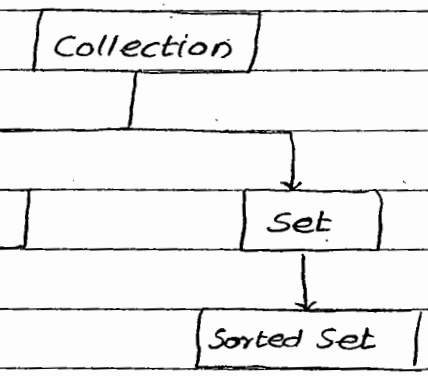
work, we need to deal with 1-D Collection Framework interfaces and classes.

* 1-D Collection Framework Interfaces -

1-D Collection Framework interfaces are classified into four types. They are -

i) java.util.Collection (I)
ii) java.util.List
iii) java.util.Set
iv) java.util.SortedSet

* Hierarchy Chart *



Collection	List	Set	Sorted Set
i) The interface Collection is available on the top of Hierarchy of all collection framework interfaces	i) List interface is a sub interface collection.	i) Set is a sub interface of Collection.	i) Sorted Set is a sub interface of Set
ii) An object of collection interface allows us to add duplicates	ii) An object of list interface allows us to add duplicates	iii) Set interface object does not allow duplicates	iv) Sorted Set interface object does not allow duplicates.
iii) Collection interface obj displays the data in the	iii) List interface object displays	iii) An object of Set	iii) Sorted Set interface

Collection	List	Set	Sorted Set
iv) Collection Interface object allows us to orga-nise the data only at end.	iv) An object of list interface allows us to organise the data either at end or at specified position	iv) An object of set interface allows us to organise data only at end	iv) An object of sorted set inter-face allows us to organise data only at end
v) An object of Collection interface allows us to retrieve data only in Forward direction. but not in backward direction.	v) An object of list interface allows us to retrieve the data both in forward & Backward directions (bidirectionally)	v) An object of set interface allows us to retrieve data only in forward direction.	v) An object of Sorted set Inter-face allows us to retrieve data only in forward direction.

Note: 1) The variables of Collection, List, Set & Sorted Set are known as Collection Framework variables

2) The objects of Collection, set & Sorted set are known as unit directional objects whereas an object of list is known as bi-directional objects.

* 1-0 Collection Framework Process *

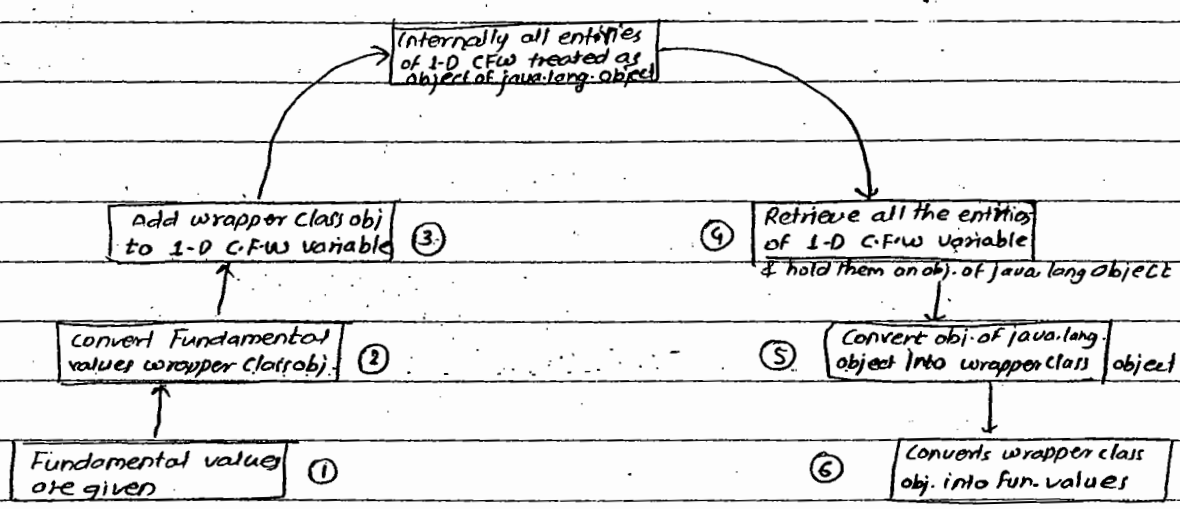
Collection Framework process contains two phases.

They are

1) Assembling / Grouping
2) Deassembling / Ungrouping

The following diagram gives the sequence of steps for assembling and deassembling.

→



The steps 1, 2, 3 are performing assembling or grouping phase. The steps 4, 5, 6 are performing deassembling or ungrouping phase.

Date - 25-APR-2011

* Methods in java.util.Collection -

Collection is one of the pre-defined interface available on the top of Hierarchy of all collection framework interface.

1) public int size() -

This method is used for obtaining the number of elements presents in Collection Framework variable.

2) public boolean isEmpty() -

This method returns true provided no elements presents in Collection Framework variable. It returns false provided collection Framework variable contains elements.

3) public boolean add(object) -

This method is used for adding the data to 1-D

Collection Framework variable. This method returns boolean as a return type which represents either true or false.

When we call this method w.r.t. Collection & List Interface objects then it returns true always Because Collection & List Interface objects allows duplicates.

- When we call this method w.r.t. set and Sorted set interface objects and if we tried to add duplicate elements then this method returns false Because set and Sorted toSet interface objects doesnot allows duplicates.

4) Public boolean addALL (Collection) -

This method is used for adding the content of one Collection Framework variable to another Collection Framework variable only at the end.

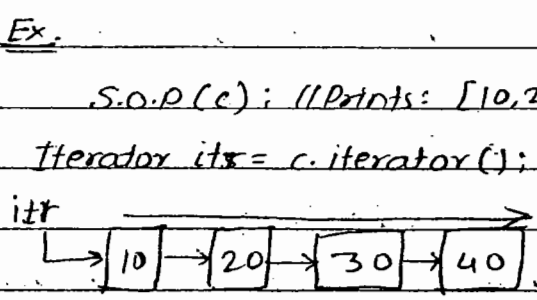
Here, also boolean return type represents either true or false (similar to add method.)

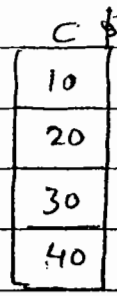
5) public Iterator iterator()

This method is used for retrieving the data from any Collection Framework variable in forword direction.

Ex.

```
S.O.P(c); //Prints: [10,20,30,40]
Iterator itr = c.iterator();
```






```
int s=0;
while (itr.hasNext())
{
```

```
        Integer io = (Integer) itr.next();
        int n = io.intValue();
        s = s+n;
    }
    s.o.p("sum of = " + s);  // Print : 100
```

5) public object[] toArray() -

This method is used for converting all the elements of Collection Framework variable into Array of objects of java.lang.Object. and they will be processed based on zero based indexing concept.

In other words, this method is used for retrieving the data from any collection Framework variable.

Ex:

```
s.o.p(c); // Prints : [10,20,30,40]
int s=0;
object obj[] = c.toArray();
for(int i=0; i< obj.length; i++)
{
    Integer io = (Integer) obj[i];
    int n = io.intValue();
    s = s+n;
}
s.o.p("sum is " + s);
```

c

10
20
30
40

* java.util.Iterator Interface -

Iterator is one of the pre-defined interface whose object is used for retrieving data from any collection Framework variable in forward direction only.

when we create an object of iterator indirectly, it by default pointing just before the first element.

* Method -

i) public boolean hasNext() -

ii) public object Next() -

Method ① returns true provided an object of Iterator interface is having next element in the collection Framework variable. Otherwise it returns false.

Method ② is used for obtaining next element of any collection Framework variable with respect to Iterator interface object provided method ① returns true.

Once method ① return false, we cannot apply method ②.

Once an object of Iterator interface returns false with respect to hasNext() method as Object of Iterator interface is pointing after last.

Date - 26-4-2011

* Methods in List Interface -

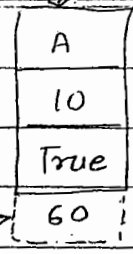
The interface list is a sub interface of Collection. So that all the methods of collection are inherited into List interface.

1) public void add(int, Object) -

This method is used for adding an element to any collection Framework variable either at the specific position or at the end.

Q) How do you add an element 60 at the end of the list by using add(int, Object) -?

l.add (3, new Integer(60));



2) public object get(int) -

This method is used for obtaining the value from the list by passing valid position.

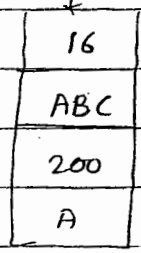
Ex. Object obj = l.get(2)

[100]

object obj = l.get(5)

[NULL]

Object obj = l.get(-3); // Invalid



3) public Object remove(int) -

4) public void remove(Object) -

These methods are used for removing the element of list either based on position or based on content.

5) public void removeAll() -

This method is used for removing all the elements of the List.

Ex: 3) Object obj = l.remove(0);

4) Object obj = l.remove(new Integer(10));

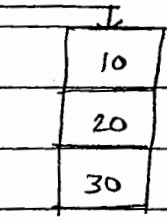
6) public ListIterator ListIterator -

This method is used for retrieving the data from any collection Framework variable either in forward direction or in Backward direction or in both.

Ex. int s=0;

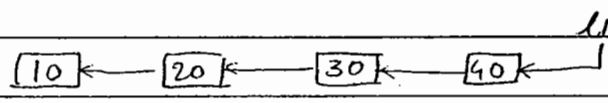
ListIterator lit = l.ListIterator();

lit



```
while ( lit.hasNext())
{
    Object obj = lit.next();
    Integer io = (Integer) obj;
    int n = io.intValue();
    s = s+n;
}
System.out.println("Sum = " + s);
```

Alternate way




```
while ( lit.hasPrevious())
{
    Object obj = lit.previous();
    Integer io = (Integer) obj;
    int n = io.intValue();
    s = s+n;
}
System.out.println("Sum = " + s);
```

③

* Methods of ListIterator -

④

i) public Object First();
ii) public Object Last();
iii) public List headList (Object obj); [$n_i <= obj$]
iv) public List tailList (Object obj); [$ni > obj$]

⑤

v) public List subList (Object obj1, Object obj2)
[$obj1 <= ni < obj2$]

Methods ① and ② are used for obtaining ~~list~~ First & last elements of any Collection Framework variable.

Method ③ is used for obtaining those elements (x_i) which are less than or equal to the target objec

Mathematically = $[x_i <= obj]$

Method ④ is used for obtaining those values (x_i) which are greater than target object.

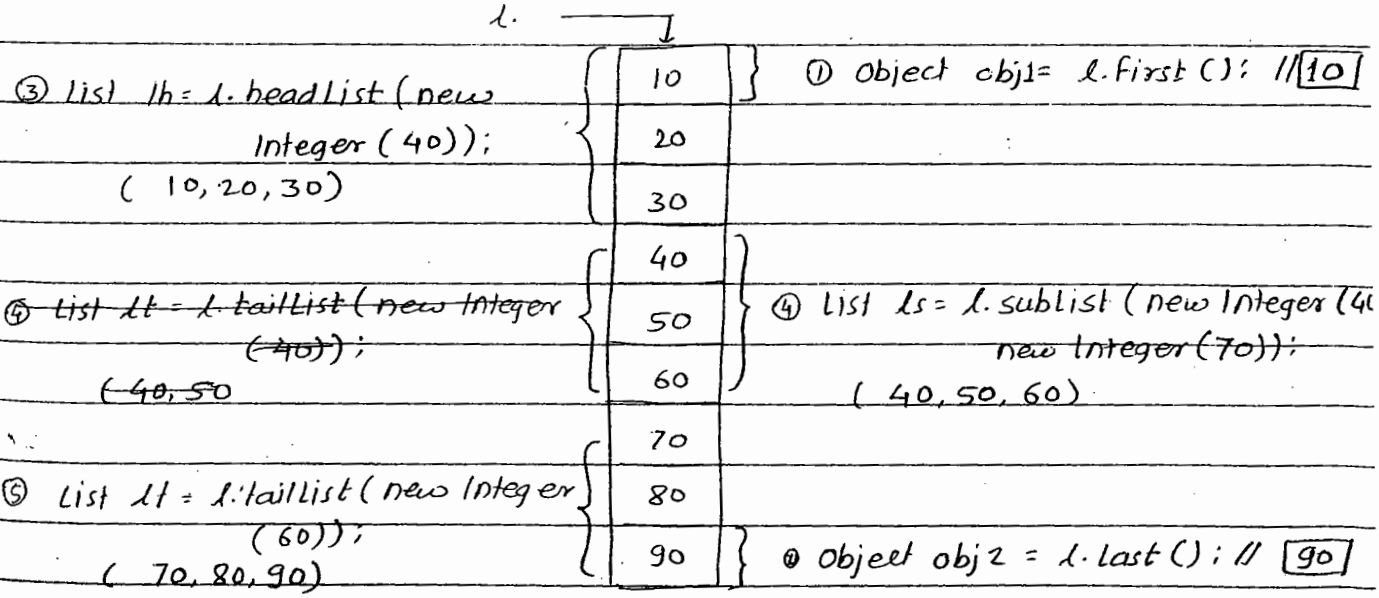
Mathematically = $[x_i >= obj]$

In other words, method ③ & ④ are used for obta-ning few top most and few bottom most value of the List.

Method ⑤ is used for obtaining those values (x_i which are greater than or equal to the target object obj1 and less than target object obj2.

Mathematically = $[obj1 <= x_i < obj2]$

Ex.



* ListIterator Interface -

ListIterator interface is one of the predefined sub-interface. Iterator interface is present in java.util.* package.

An object of List ~~interface~~ Iterator interface is always used for ~~fxe~~ extracting or retrieving the data from any Collection Framework variable either in forward direction or in backward direction or in both.

When we create an object of listIterator, which is by default ~~positioning~~ pointing just before the first element of any Collection Framework variable.

* Methods -

1) public boolean hasNext();
2) public object next();

} Inherited from Iterator (I)

3) public boolean hasPrevious();
4) public object previous();

Method ③ returns true provided ListIterator (I) object is having previous element in the Collection Framework variable. otherwise it returns false.

Method ④ is used for obtaining the previous element of any Collection Framework variable with respect to ListIterator interface object provided method ③ must return value.

Method ① & ② looks in forward direction & method ③ & ④ looks in the backward direction.

Q] What are the differences between the Iterator and ListIterator ?

	Iterator	ListIterator
	i) Iterator is one of the best interface for the listIterator.	i) ListIterator is sub interface of Iterative.
	ii) An object of Interface Iterator always allows the programmer to extract or retrieve one data only in the forward direction of any Collection Framework variable.	ii) An object of ListIterator allows one programmer to retrieve/extract data of any collection Framework variable either in forward or in backward direction or in Both.
	iii) public boolean hasNext(); public object next();	iii) public boolean hasPrevious(); public object previous();

I)

* Methods in java.util.Set -

1) we know that Set is a sub-interface of Collection, so that all the methods of collection Interface are inherited into set Interface.

And Interface Set does not contains any special methods except the methods of Collection.

Even though methods of Set & Collection are same, methods of Collection are defined by some predefined class in such a way that duplicate elements are allowed and displaying in random order. The methods of set are defined by some other predefined class in such a way that only unique elements are allowed, and they must be displayed in random order.

In other words, even though, these methods are same in both collection and Set interface, their implementation or

* Methods in java.util.SortedSet -

We know that SortedSort is sub-interface of Set so that all the methods of set are inherited into Sorted Set which are nothing but methods of Collection.

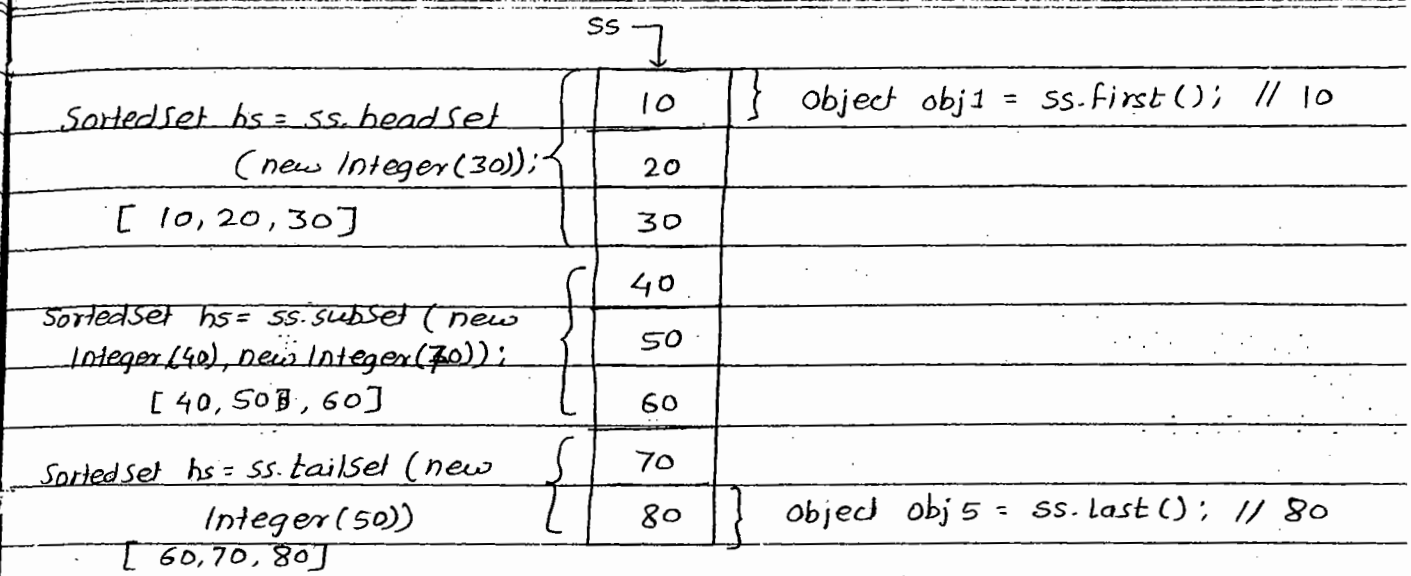
Even though, the methods Set and SortedSet are same, the methods of Set are defined is some pre-defined class in such a way that unique elements are allowed and they must be displayed in Random order. whereas methods of SortedSet are defined in some other predefined class in such a way that unique elements are allowed and they are displaying in Sorted order.

* Methods in java.util.SortedSet - Date- 27-APR-2017

We know that SortedSet is a sub interface of Set, so that all the methods of set are inherited into SortedSet which nothing but methods of Collection.

Even though the methods Set and SortedSet are same, these methods of Set are defined in some other pre-defined class in such a way that unique elements are allowed and they must be displayed in random order. Whereas methods of SortedSet are defined in some other pre-defined class in such a way that unique elements are allowed and they are displaying in Sorted order.

1) public Object first();
2) public Object Last();
3) public SortedSet headSet(Object obj); [xi <= obj]
4) public SortedSet tailSet(Object obj); [xi > obj]
5) public SortedSet subSet(Object obj1, Object obj2);
[obj1 <= xi < obj2]



* 1-D Collection Framework Classes -

1-D Collection Framework classes are those which are containing the definition for more abstract methods which are coming from 1-D Collection Framework interfacee

All the predefined classes of 1-D collection Frame-work are present in java.util.* package.

The following table gives 1-D collection framework interface name, whose corresponding 1-D collection framework class name & their hierarchy.

1-D CFW Interface	1-D CFW class Name	Hierarchy
Collection	① AbstractCollection	← implements Collection
List	② AbstractList	← extends AbstractCollection
Set	③ AbstractSet	← extends AbstractCollection ← implements (Set)
SortedSet	④ AbstractSequentialList	← extends. AbstractList

1-D CFW Interface	1-D CFW Class Name	Hierarchy
—	⑤ LinkedList ← extends.	AbstractSequentialList
—	⑥ ArrayList ← extends	AbstractSequentialList
—	⑦ HashSet ← extends	AbstractSet
—	⑧ TreeSet ← extends	AbstractSet
	← Implements	SortedSet

In the above hierarchy chart, the class ①, ②, ③, ④ are predefined abstract classes and whose object can not be created directly. Hence they will not be used directly in real time application But all these classes are playing the very important background role in defining abstract methods which are inherited from predefined 1-D Collection Framework Interfaces.

However these classes are treated as playing a middle man role between predifined 1-D CFW interface and predefined bottommost concreate sub classes of 1-D collection Framework.

The classes ⑤, ⑥, ⑦, ⑧ are pre-defined concrete subclasses of all abstract classes of 1-D collection Framework. Since these are concrete, whose objects can be created directly. So that we can use these objects in real time applications.

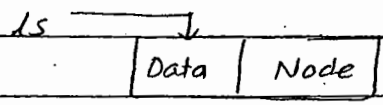
* LinkedList -

1) LinkedList is one of the concrete subclass of all abstract classes of 1-D Collection Framework.

ii) Creating a LinkedList is ~~one of the concrete subclass of~~ nothing but creating an object of LinkedList class

ex:

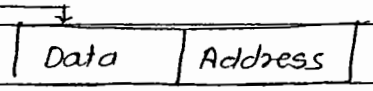
```
LinkedList ls = new LinkedList();
```



Here, ls is an object of Linked List which is nothing but Collection Framework variable which allows us to group different types of values or similar type of value or both.

In LinkedList the data is ~~on~~ organised internally in the form of nodes.

iii) Node contains two parts and whose structure is

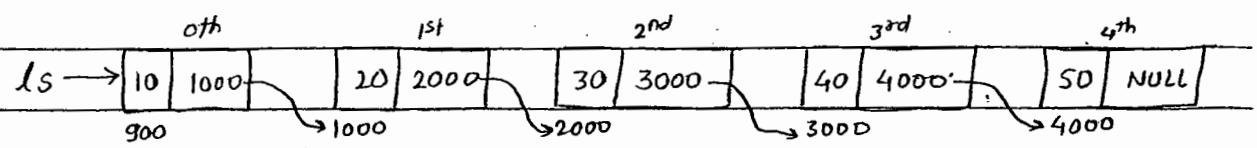


Every object of LinkedList is nothing but the name of the node.

iv) The data part represents the type of data being represented in LinkedList class object

Address part represents address of next node and for the last node, address part must be null which includes end of the LinkedList.

Ex: Represent 10, 20, 30, 40, 50 in a LinkedList



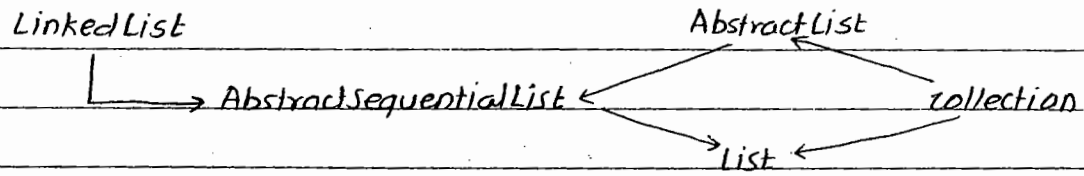
The data organised in the LinkedList in such a way that address of (ith) node stored in the address part of $(i-1)^{th}$ node contains address of $(i+1)^{th}$ node and for the

Date- 30-APR-2011

last node address part must be null which indicates end of the LinkedList.

Note: Unlike implementation of Data Structure of C, C++, the concept of linkedlist in java programming the node values can be processed based on the position.

* Hierarchy of the Linked List -



* LinkedList Class Profile -

* Constructor-

1) LinkedList() -
2) LinkedList(int size)

* Instance methods -

3) public void addFirst(object)
4) public void addLast(object)
5) public Object removeFirst()
6) public Object removeLast()
7) public Object getFirst()
8) public Object getLast()

Constructor ① is used for creating an object of Linked list.

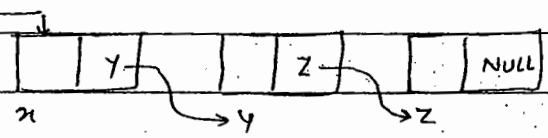
Ex: LinkedList ll = new LinkedList();

ll. → null

Constructor ② is used for creating an object of Linked List by specifying no. of nodes to be created.

Ex:

```
LinkedList ll = new LinkedList(3);
```



It is highly recommended to use default constructor in collection framework rather than using parameterized con-structor.

Method ③ & ④ are used for adding an element to the LinkedList either at 1st position (0th position) or at the last position (n-1th position).

Ex:

```
LinkedList ll = new LinkedList();
ll.addFirst(new Integer(10));
```

or

```
ll.add(0, new Integer(10));

ll.addLast(new Integer(30));
```

or.

```
ll.add(ll.size(), new Integer(30));
```

Methods ⑤ & ⑥ are used for the removing 1st & last element of the linkedlist.

Ex: `object obj1 = ll.removeFirst();` or

`object obj1 = ll.remove(0);`

`object obj2 = ll.removeLast();` or

`object obj2 = ll.remove(ll.size()-1);`

Methods ⑦ & ⑧ are used for obtaining first and last element of a LinkedList.

Ex. Object Fobj = ll.getFirst();
Object Lobj = ll.getlast();
or
Object Fobj = ll.get(0);
Object Lobj = ll.get(ll.size()-1);

Q] write a Java program which illustrate the concept of LinkedList.

→ Whenever we write collection Framework related applications, it is mandatory for java programmer to follow always "Uses-a" relationship (A method ~~in which~~ one class is using an object of some other class)

```
// LinkedListDemo.java
import java.util.*;

Class LinkedListDemo
{
    public static void main(String args[])
    {
        LinkedList ll = new LinkedList(); // Uses-a
        S.O.P("Size = " + ll.size());
        S.O.P("Content = " + ll);

        // Add element or data to LinkedList.
        ll.add(new Integer(20));
        ll.add(new Integer(30));
        ll.add(new Integer(40));

        S.O.P("Size = " + ll.size()); // 3
```

```
// Adding data to linked list either at first or at last
ll.addFirst(new Integer(10));
ll.addLast(new Integer(50));
S.O.P("Size = " + ll.size()); // 5
S.O.P("Content = " + ll); // [10,20,30,40,50]

// Extracting data from linked list Iteration.
S.O.P("Extracting data using Iterator()");
int s=0;
Iterator itr = ll.iterator();
while(itr.hasNext())
{
    Object obj = itr.next();
    Integer io = (Integer) obj;
    int x = io.intValue();
    S.O.P(x);
    s = s+x;
}
S.O.P("Sum → Iterator() = " + s);

// Extracting data using ListIterator()
S.O.P("Extracting data using ListIterator");
ListIterator ltr = ll.listIterator();
int s1=0;
while(ltr.hasNext())
{
    Object obj = (Integer) ltr.next();
    Integer io = (Integer) obj;
    int x1 = io.intValue();
    s = s+x1;
}
```

```
// At this stage list pointing after last element of the
// Collection Framework Variable
S.O.P("Sum - ListIterator = " + s);
S.O.P("Extracting Data in Backward Direction");
int s2 = 0;
while (itr.hasPrevious())
{
    Object obj2 = itr.previous();
    Integer io = (Integer) obj2;
    int n = io.intValue();
    s2 = s2 + n;
}
S.O.P("Sume ListIterator() " + s2);
S.O.P("Extracting Data From ll.toArray()");

Object obj[] = ll.toArray();
int s3 = 0;
for(int i=0; i < obj.length; i++)
{
    Integer io = (Integer) obj[i];
    int n3 = io.intValue();
    s3 = s3 + n3;
}

S.O.P("Sum: toArray() = " + s3);
S.O.P("Extracting Random Element");
S.O.P("Second Element : " + ll.get(1));
S.O.P("Last Element : " + ll.get(ll.size()-1));
} //Main
} // LinkedListDemo
```

* Disadvantages of LinkedList -

1) LinkedList will take more amount of memory space because both Data Part & address part takes the considerable memory space on Heap memory.
2) Retrieving the data from LinkedList takes more amount of time because, retrieving the address of next node from heap memory takes time. Hence LinkedList is not recommended to be used in realtime application

To eliminate the above problems, we use the concept of ArrayList.

* Array List *

1) ArrayList is one of the bottom most concrete subclass of all I-D Collection framework Abstract classes.
2) Creating ArrayList is nothing but creating an object of ArrayList class.
   Ex. ArrayList al = new ArrayList ();
3) In the ArrayList data is organised in the form of cells

* Advantages of ArrayList over Linked List -

1) ArrayList takes less memory space & performance is improve because user defined values are stored in Heap Memory & whole address are stored in Associative Memory. (Cell values are stored in Heap Memory and cell Address are stored in Associative memory.)

2) Retrieving the data from Arraylist takes negligible amount of time i.e. retrieval time is very fast because anything we retrieve from associative memory will take negligible amount of time.

Q] Write a java program which illustrate the concept of ArrayList.

→ Hint: Substitute the following code in LinkedListDemo.java

a) Substitute ArrayList in place of LinkedList.

b) Substitute add(0, new Integer(10)) in place of addFirst(-);

c) Substitute add(l.size(), new Integer(10)) in the place of addLast(new Integer(10));

* Constructor of ArrayList -

i) ArrayList() // late memory Allocation.

ii) ArrayList(int size); // Advance memory Allocation

iii) ArrayList(int size, int incrementRatio); // Pre-// memory allocation.

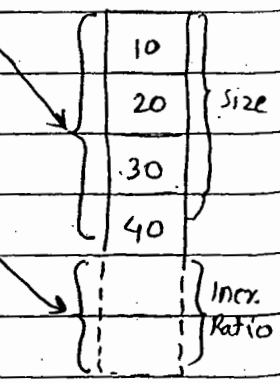
Constructor ① is used for creating an object of ArrayList class with the default size. The default size of the ArrayList is 10 cells.

Constructor ② is used for creating an object of ArrayList class by specifying the no. of cells to be created.

Constructor ③ is used for creating an object of ArrayList by specifying no. of class to be created & increment Ratio.

Ex. ArrayList al = new Arraylist(4,2)

The no. of values in al object are 4. Arraylist object uses pre-memory allocation formula when list becomes full. For this it uses incrRatio value.



* Formula -

size = size + incrRatio;

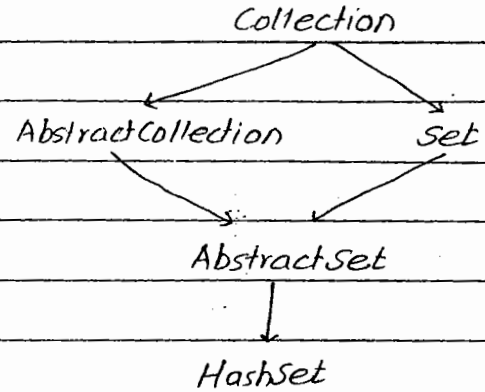
Date- 1-May-2011

* HashSet and TreeSet *

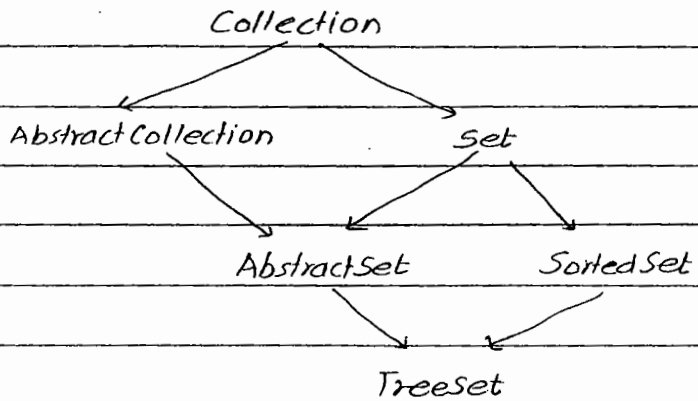
HashSet and TreeSet are the bottommost concrete subclasses of all abstract classes of 1-D Collection Framework.

HashSet	TreeSet
i) HashSet Class object organises data in the form Hashtable by following Hashing Mechanism.	i) TreeSet class object organises the data in the main memory by following Binary Trees co-ncepts (AVL Trees). (Anderson Velkerion Lendis)
ii) Whatever data we add to the HashSet, we can not dete-rmine in which order. HashSet Class objed displays the data. Because Sun microsystems didnot diclose ~~in~~ which ~~order~~ the hashing mechanism followed.	ii) TreeSet class object always display the data in sorted or-der.
iii) The operations like insertion, deletion & modification takes considerable amount ~~to~~ of time i.e. very expensive.	iii) The operations like insertion, deletion & modification takes neglible amount of time i.e. less expensive.
iv) Retreiving the data from HashSet will take considerable amount of time i.e more time.	iv) Retriving the data from TreeSet will take negligible amount of time i.e. fasting retrieval.
v) Creating HashSet is nothing but creating an object of HashSet class Ex: HashSet hs = new HashSet();	v) Creating TreeSet is nothing but creating an object TreeSet class ex: TreeSet ts = new TreeSet();

* Hierarchy Structure of HashSet -



* Hierarchy Structure of TreeSet -



* write a java program which illustrate the concept of HashSet and TreeSet.

```
// HSTs.java
import java.util.*;
class HSTs
{
    public static void main (String args[])
    {
        // HashSet hs = new HashSet();
        TreeSet ts = new TreeSet();
        S.O.P("size = " + ts.size()); //0
```

```
// Adding data
    ts.add (new Integer(100));
    ts.add (new Integer(1));
    ts.add (new Integer(10));
    ts.add (new Integer(90));
(False) ts.add(new Integer(90)); // Duplicates are not
                                 // allowed.

    S.O.P("size = " + ts.size()); // 4
    S.O.P("Content = " + ts); // [....]

// Retrieving the data
    Iterator itr = ts.iterator();
    int s=0;
    while (itr.hasNext())
    {
        Object obj = itr.next();
        System.out.println(obj);
        Integer io = (Integer) obj;
        int x = s + x;
    }
    S.O.P("Sum = " + s);
} // Main
} // HSTS
```

It is highly recommended to use TreeSet class object for organising the data in the main memory under set family. and to get more internal performance.

* 2-D / Double Collection Framework or Maps *

A 2-D Collection Framework is one in which the data is organised in the form of (key, value) pair.

In the (key, value) pair, the value key must be unique / distinct and the value of value may or may not be unique.

The values of (key, value) pair must be treated internally as objects of java.lang.Object.

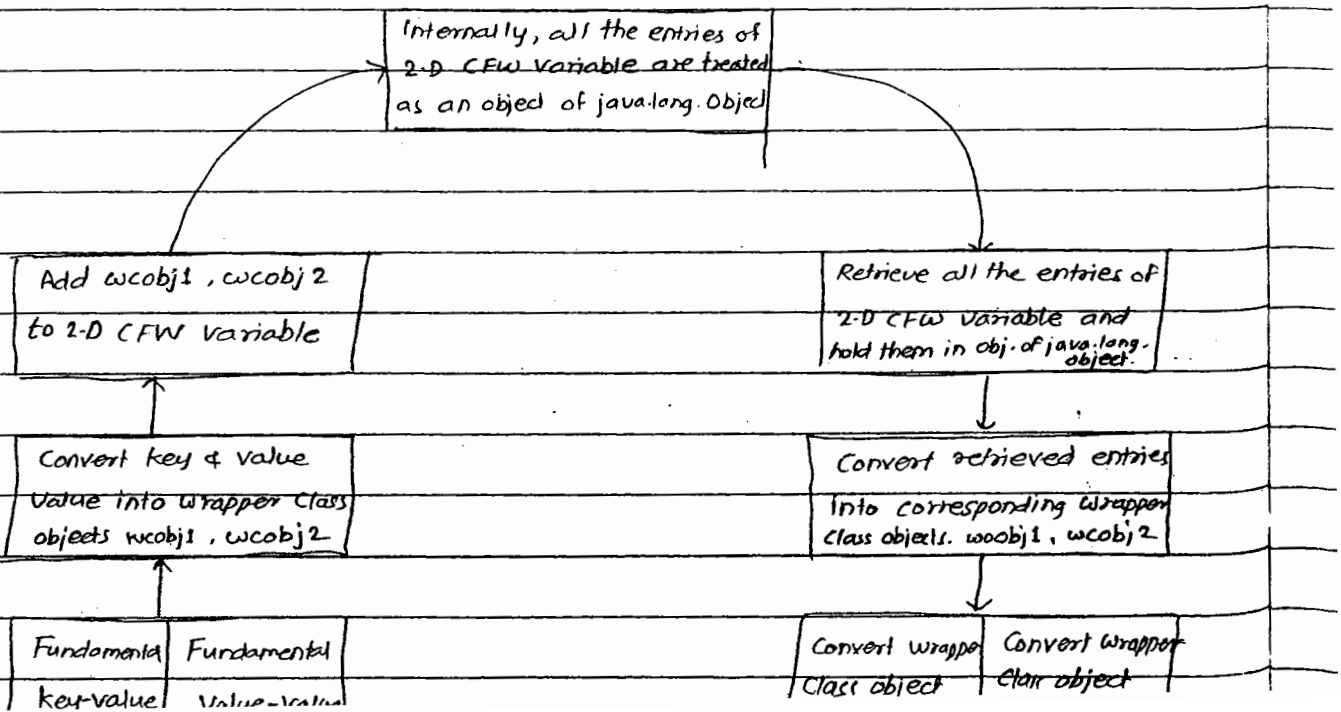
* 2-D Collection Framework Process -

2-D Collection Framework Process is nothing but how to organise the data and how to deorganise the data. In other words, this process contains two phases. They are -

i) Assembling or Grouping phase

ii) Deassembling or Ungrouping Phase

The following diagram gives sequence of steps for Assembling and Deassembling Phases.



The steps ① ② ③ are used for performing assembling phase and steps ⑤, ⑥ ⑦ are used for performing the deassembling phase.

In order to deal with 2-D collection framework, we need to learn 2-D collection Framework Interfaces and classes

Date- 2-May-2011

* 2-D collection Framework Interfaces-

2-D collection Framework Interfaces are divided into three types

i) java.util.Map
ii) java.util.map.Entry
iii) java.util.SortedMap.

1) java.util.Map (I) -

i) It is one of the topmost predefined interface in 2-D collection framework interfaces.

ii) An object of Map Interface can not created directly but it can be instantiated indirectly.

Ex:

```
Map m = new XYZ();
```

Here, m is an indirect object of map interface and XYZ is a subclass of Map Interface.

iii) An object of map allows us to organise the data in the form (key, value) pair. In (key, value) pair, the value of key must be distinct and value of value may or may not be distinct.

iv) The values of (key, value) pair must be objects.

v) whatever the data we add to the map interface variable

in whichever order we add the data to map Interface variable, in the same order the data will be displayed.

* Methods in Map Interface -

1) public int size() -

This method is used for finding no. of elements in 2-D Collection Framework variable or determining the size of 2-D Collection Framework variable.

2) public boolean isEmpty() -

This method returns true provided 2-D collection framework variable doesn't contain any element (empty). This method returns False provided 2-D Collection Framework variable contains some elements (non-empty)

3) public void put (Object kobj, Object vobj) -

This method is used for adding the data to the 2-D Collection Framework variable in the form of (key, value) pair.

4) public Object get (Object kobj) -

This method is used for obtaining value of 'value' by passing key value. If the key value is not found in the 2-D Collection Framework variable then the value of value is null. otherwise it is not null.

Ex. ⇒ Object vobj = m.get(new Integer(20));
S.O.P(vobj); // 1.7
⇒ Object vobj2 = m.get(new Integer(50));

m

10	1.5
20	1.7
30	1.9

5) public object void remove (int);

6) public void remove (Object) -

The above methods are used for removing the entries of 2-D Collection Framework variable either on basis of position or on the basic of content i.e. key object.

7) public void removeAll() -

This method removes all the entries of 2-D Collection Framework variable.

8) public Set entrySet() -

This method is used for extracting or retrieving the content of 2-D collection Framework variable.

Ex.1

```
Set s = me.entrySet();
// s { (10=1.5), (20=2.5), (30=3.5) }
Iterator itr = s.iterator();
```

m →

10	1.5
20	2.5
30	3.5

itr → [10=1.5] → [20=2.5] → [30=3.5]

```
while ( itr.hasNext())
{
    Object mobj = itr.next();
    MapEntry me = (MapEntry) mobj;
    Integer io = (Integer) me.getKey();
    Float fo = (Float) me.getValue();
    int accno = io.intValue();
    Float bal = fo.floatValue();
    S.O.P (" Balance of A/c No = " + accno + " is " + bal);
} // Iterator.
```

Ex 2

m →

10	1.5
20	2.5
30	3.5

```
Object Set s = m.entrySet();
Object mobj[] = s.toArray();

for(int i=0; i<mobj.length; i++)
{
    MapEntry me = (MapEntry) mobj[i];
    Object kobj = me.getKey();
    Object vobj = me.getValue();
    Integer io = (Integer) kobj;
    Float Fo = (Float) fObj;
    int accno = io.intValue();
    Float bal = Fo.FloatValue();

    System.out.println("Balance of AccNo = " + accno +
        " is " + bal);
} // while
```

9) public Set keySet() -

This method is used for obtaining set of keys from 2.0 Collection Framework variable and pass these sets of key values to get method and obtain set of values.

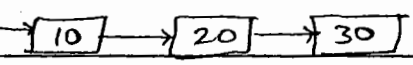
Ex. 1

m →

10	1.5
20	2.5
30	3.5

```
Set s = m.getKeySet();
// s = {10, 20, 30}
Iterator its = s.iterator();
```

itr.




```
while(itr.hasNext())
{
```

```
Object kobj = its.next();
Object vobj = m.get(kobj);

Integer io = (Integer) kobj;
Float fo = (Float) vobj;
int accno = io.intValue();
float bal = fo.floatValue();

S.O.P("Balance of A/c No = " + accno + " is " +bal);
} // while.
```

Ex: 2

```
Set s = m.keySet();
// s = {10,20,30}
Object obj[] = s.toArray();

for(int i=0; i < obj.length ; i++)
{
    Object kobj = obj[i];
    object vobj = m.get(kobj);

    Integer io = (Integer) kobj;
    Float fo = (Float) vobj;
    int accno = io.intValue();
    Float bal = fo.floatValue();

    S.O.P("Balance of Acc.No = " + accno + " is " + bal)
}
```

Date: 3-May-2011

Q] What are the approaches used for extracting the data from 2-D Collection Framework.?

→ We have two approches to extract the data from 2-D collection Fromework variable.

1) public set entrySet()
2) public set keyset()

2) java.util.Map.Entry -

Map is an interface and Entry is one of the predefined class in Map Interface.

Map.Entry is used for seperating "key" value & value of "value" from 2-D collection Framework variable.

* Methods -

1) public object getkey():
2) public object getvalue():

The above two methods are used for obtaining key value and value of value from any 2-D collection variable by type casting mobj of java.lang.Object into java.util.Map.Entry.

3) java.util.SortedMap -

SortedMap is one of the sub interface of Map. So that all the methods of Map are inherited into Sorted map.

An object of SortedMap also organises the data in the form of (key, value) pair.

An object of SortedMap always displays the data is sorted order and sorting can be done based on key value.

The realtime implementation of Sorted map in wireless application development is phonebook of the mobiles.

* Methods -

1) public object first();
2) public object last();
3) public SortedMap headMap(object obj1)

xi <= obj1

4) public SortedMap tailMap (object obj2)

xi > obj2

5) public SortedMap subMap (object obj1, object obj2)

obj1 <= xi < obj2

Object obj = sm.first();

10	1.5

10	1.5
20	2.5
30	1.5

SortedMap sm2 = sm.headMap(new Integer(30));

// Sm2 = {(10=1.5), (20=2.5), (30=1.5)}

40	4.5
50	3.5

SortedMap sm3 = sm.subMap(new Integer(40), new Integer(60));

// sm3 = {(40=4.5), (50.35)}

60	6.7
70	7.5

SortedMap sm4 = sm.tailMap(new Integer(50));

Object obj = sm.last();

70	7.5

* 2-D or Double Collection Framework Classes or Map Classes -

2-D C.F.W Interfaces	2-D CFW classes	Hierarchy
Map	① AbstractMap	← implements Map
Map.Entry	———	———
SortedMap	② AbstractsortedMap	← extends AbstractMap
		← implements SortedMap
	③ HashMap	← extends AbstractMap
	④ TreeMap	← extends AbstratSortedMap

2-D Collection Framework classes contains defi-nition for those abstract methods which are inherited from 2-D collection Framework interfaces. All the 2D collection Framework classes are present in a predefined packages called java.util.* package.

The classes ① & ② are abstract classes so that whose objects can not be created directly and we never use them directly in realtime applications. And moreover these classes are playing very impor-rtant bagckground role in defining abstract methods of 2D collection Framework interfaces.

In other words, these classes are playing the middleman role between 2D collection framework Interface names and 2D C.F.W concrete classes.

The classes three and for are the bottommost

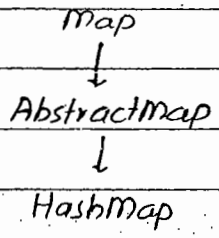
concrete subclasses of all 2D abstract classes. Hence, the classes of ③ and ④ objects can be created directly and we use them in realtime applications.

* HashMap and TreeMap -

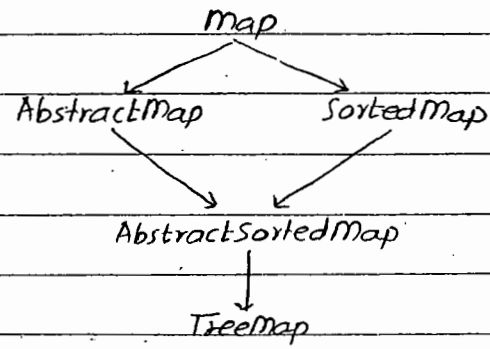
The objects of HashMap & TreeMap never allows to place duplicate values for the keys.

HashMap	TreeMap
i) An object of HashMap organises the data in the form of (key,value) pair by following Hashing mechanism	i) An object of TreeMap organises the data in the form of (key,value) pair by following the concept of Binary Trees.
ii) We can not determine in which order HashMap object displays the data. Because sun microsystems did not disclosed which hashing mechanism is followed.	ii) An object of TreeMap displays the data always in sorted order.
iii) The operations like insertion, deletion, modification takes considerable amount of time i.e quite expensive	iii) The operations like insertion, deletion, modification takes negligible amount of time i.e. less expensive.
iv) Retrieving data from HashMap take considerable amount of time i.e more time.	iv) Retrieving the data from TreeMap takes negligible amount of time i.e. less time.
v) Creating HashMap is nothing but creating an object of HashMap class	v) Creating TreeMap is nothing but

Hierarchy For HashMap



Hierarchy For TreeMap



* write a java program which illustrate the concept of HashMap & TreeMap -

→ // HMTM.java

```
import java.util.*;

public class HMTM
{
    public static void main(String args[])
    {
        // HashMap hm = new HashMap();
        TreeMap tm = new TreeMap();
        S.O.P("size =" + tm.size());
        S.O.P("Content =" + tm);

        // Adding data TreeMap
        tm.put(new Integer(100), new Float(1.5F));
        tm.put(new Integer(1), new Float(2.5F));
        tm.put(new Integer(40), new Float(1.5F));
        tm.put(new Integer(90), new Float(9.5F));
        tm.put(new Integer(100), new Float(9.5F));

        S.O.P("Size =" + tm.size());
        S.O.P("Content =" + tm);
```

```
S.O.P ("Extracting the data from 2D CFW variable
        using entrySet()");
Set s = tm.entrySet();
Iterator itr = s.iterator();

while (itr.hasNext())
{
    MapEntry me = (Map.Entry) itr.next();
    Object kobj = me.getKey();
    Object vobj = me.getValue();
    Integer io = (Integer) kobj;
    Float fo = (Float) vobj;
    int accno = io.intValue();
    Float bal = fo.floatValue();

    S.O.P ("Bal. of A/c No = " + accno + " is " + bal);
}

S.O.P ("Extracting data from 2D CFW variable using
        toArray()");
s = tm.keySet();
Object obj[] = s.toArray();
for (int i=0; i < obj.length; i++)
{   object kobj = obj[i];
    Integer io = (Integer) tm.get (kobj);
    Float fo = (Float) kbb
    Integer io = (Integer) kobj;
    Float Fo = (Float) m.get (kobj);
    int accno = io.intValue();
    Float bal = fo.floatValue();
    S.O.P ("Bal. of Accno = " + accno + " is " + bal);
}
} //main
```

Date - 4-May-2011

* Legacy Collection Framework *

In the old versions of Java, the concept of Collection Framework was known as ~~Legacy~~ to Data Structures. The concept of Data Structure was meeting some of the requirements of the industry and unable to fulfill some other requirements of the industry. Hence, Sun microsystems has revised or re-engineered the concept of data structures and released to the industry on the name of New Collection Framework. And the existing Data Structures concept was renamed as legacy Collection Framework.

The basic difference between New Collection Framework and Legacy Collection Framework is that all the classes & interfaces of new collection Framework are belongs to non-synchronised and they provides concurrent access. All the classes & interfaces of the legacy Collection Framework are belongs to synchronised and they provides sequential Access.

Legac Collections framework also contains two categories. They are

i) 1-D Legacy Collection Framework
ii) 2D Legacy Collection Framework.

Dealing 1-D & 2-D of any Legacy CFW is nothing but learning about their interfaces and classes

1) java.util.Enumeration } Interface
2) java.util.Vector } 1D legacy CFW Classes
3) java.util.Stack }
4) java.util.Dictionary } 2D Legacy CFW Classes.
5) java.util.Hashtable }
6) java.util.Properties. }

1) java.util.Enumeration -

i) It is one of the predefined interface whose object is always is used for extracting or retrieving the data from legacy Collection Framework classes in forward direction only.

ii) An object of Enumeration is by default pointing just before the first element of any legacy collection Framework variable.

iii) The functionality of enumeration is exactly similar to iterator. But enumeration retrieves the data sequentially and iterator retrieves the data concurrently.

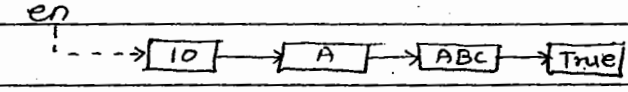
* Methods --

1) public boolean hasMoreElements();
2) public Object nextElement();

Method ① returns true, provided enemeration interface object is having more elements in legacy collection framework variable. Otherwise it returns false.

Method ② is used for obtaining an element of legacy collection Framework variable until method ① returns false.

Ex-




```
while ( en.hasMoreElements())
{
    object obj = en.nextElement();
    s.o.p( obj);
}
```

2) java.util.Vector -

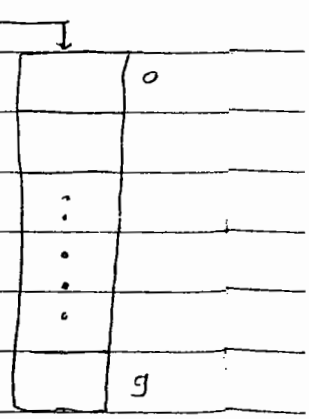
i) Vector is one of the concrete class presents in JD legacy collection Framework.

ii) The functionality of vector is exactly similar to ArrayList. But vector data can be accessed in sequential and ArrayList data can be accessed in concurrent.

iii) In Vector the data is organised in the form of Cells Cell values are stored in Heap Memory and cell addresses are stored in Associative memory.

iv) Creating a vector is nothing but creating an object of Vector class.

Ex. Vector v = new Vector();



v) The default capacity of the vector object is 10 cells.

* Constructors -

1) Vector();
2) Vector(int size);

* Methods -

3) public int size();
4) public int capacity();
5) public void addElement(object);
6) public void addElement(int, object);
7) public ~~void~~ object removeElementAt(int);
8) public void removeElement(object);

9) public void setElementAt(int, object);

10) public Enumeration ~~enumeration()~~ elements();

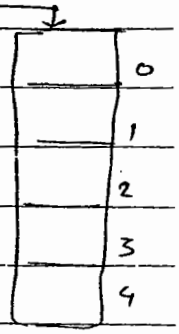
Constructor ① is used for creating an object of the Vector class without specifying the explicit size.

Ex: Vector v = new Vector();

S.O.P(~~v.size()~~); v.capacity()); // 10

Constructor ② is used for creating an object of vector by specifying the explicit size.

Ex: Vector v = new Vector(5);



Method ③ is used for finding no. of elements presents in the cells of vector.

Method ④ will give the total no. of cells available in the vector.

Ex: Vector v = new Vector();

S.O.P(v.size()); // 0

S.O.P(v.capacity()); // 10

Method ⑤ is used for adding an element to the vector only at the end.

Method ⑥ is used for adding an element to the vector at the specific position.

Ex. Vector v = new Vector();

v.addElement(new Integer(10));

v.addElement(new Integer(1000));

S.O.P(v);

// v = [10, 1000]

```
v.addElement(1, new Float(10.75f));
sop(v); // [10, 10.75, 1000]
```

All the elements of vector can be processed based on 0-Based indexing concept.

Methods ⑦ & ⑧ are used for removing elements from the vector either based on the position or based on the content.

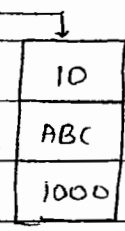
Ex.
```
Object obj = v.removeElementAt(1);
s.o.p(obj); // 10.75f
```
or
```
s.o.p("Removed Element : " + v.removeElement(new Float(10.75f)));
```

Method ⑨ is used for replacing an element into the vector by specifying its position.

Ex.
```
v.setElementAt(1, new Float(100.75f));
s.o.p(v); // [10, 100.75, 1000]
```

Method ⑩ is used for extracting the data from 1-D legacy collection Framework variable in forward direction.

Ex.
```
s.o.p(v); // [10, ABC, 1000]
Enumeration en = v.elements();
while(en.hasMoreElements())
{
    Object obj = en.nextElement();
    s.o.p(obj);
}
```



* Write a java program which illustrate the concept of Vector Class.

→ // Vector~~-java~~Demo.java

```
import java.util.*;

public class VectorDemo
{
    public static void main (String args[])
    {
        Vector v = new Vector();
        System.out.println("Capacity = " + v.capacity());
        System.out.println("Size = " + v.size());
        System.out.println("Content = " + v);
        // Adding data to Vector
        v.addElement (new Integer (10));
        v.addElement (new Float (10.75F));
        v.addElement (new Character('A'));
        v.addElement ("Sathya");

        // Extracting the data.
        Enumeration en = v.elements();
        while( en.hasMoreElements())
        {
            Object obj = en.nextElement();
            S.o.P (obj);
        }
    } // main
} // VectorDemo.
```

Date- 5-May-2011.

3) java.util.Stack -

Stack is one of the 10 legacy Collection Framework class.

ii) The class Stack is the subclass of vector. So that all the methods of vector are inherited into stack.

iii) The basic working principle of stack is LIFO. That is the element which is inserted at last is that element will be taken as of first.

iv) Creating a stack is nothing but creating an object of Stack class

Ex: Stack s = new Stack();

v) The elements of the stack are processed and stored & displayed with right to left associativity.

vi) The diagramatic representation of stack class object in the Heap memory is given below.

Relative Stack Array Position	Absolute Stack Array Position	Stack	Absolute Array Position	Relative Array Position
4	3	10	0	1
3	2	20	1	2
2	1	30	2	3
1	0	40	3	4

L ← R

System.out.println(s); // [10, 20, 30, 40]

* Stack Class Profile -

* Constructor -

① Stack();

* Instance Methods -

1) public ~~void~~ boolean empty()

2) public void push(Object)

3) public Object pop()

4) public Object peek()

5) public int search(Object)

Constructor ① is used for creating an object of Stack.

Ex: Stack s = new Stack();

Method ② returns true provided stack is not containing any elements. It returns false provided stack contains some elements.

method ③ is used for inserting an element into the stack.

Method ④ is used for removing top most element of the stack permenently. (Recent Absolute stack Array Position element).

Method ⑤ is used for retrieving or obtaining the top most element of the stack.

Method ⑥ is used for searching an element into the stack. This method returns relative stack array position, provided the search is successful. It returns -1, provided search is unsuccessful.

* write a java program which illustrates the concept of Stack

→ // StackDemo.java

```
import java.util.Stack;

public class StackDemo
{
    public static void main( String args[])
    {
        Stack s = new Stack();
        System.out.println("Size = " + s.size()); //0
        System.out.println("content = " + s.); // []
        System.out.println("Is Empty = " + s.empty()); // True
        s.push( new Integer(10));
        s.push (new Integer(20));
        s.push (new Integer(30));
        s.push (new Integer(40));

        System.out.println("Content = "+s); // [10 20 30 40]
        System.out.println("Size = " + s.size()); //4
        System.out.println("Is Empty = "+s.empty()); // False

        System.out.println("Popped Item = "+s.pop()); //40
        System.out.println("Content after Deletion = "+s); // [10 20 30]
        System.out.println("Top Element = " + s.peek()); //30

        System.out.println("Content after Peek="+s); // [10 20 30]
        System.out.println("is 10 found ="+ s.search (new Integer(10));
                                                    //3  RASP RSAP
        System.out.println("is 100 found = "+ s.search(new Integer(100));
                                                    //-1
    } // main
} // StackDemo
```

4) java.util.Dictionary -

i) It is one of 2-D legacy Collection Framework Abstract class.

ii) Since it is an abstract, whose object cannot be created directly. So that the user of dictionary is very less in realtime application and moreover this class provides the very important background roll in providing the common methods in for both Hashtable and Properties classes.

iii) When we create an object of Disctionary indirectly it organises the data in the form of k(key,value) pair. And we know that the value of key must be unique and the value of value may or may not be unique. The values of both (key, value) pair must be an objects.

* Methods -

1) public int size()
2) public boolean isEmpty()
3) public void put(object kobj, object vobj)
4) public Object get(Object kobj)
5) public void remove(Object obj)
6) public void removeAll()
7) public Enumeration keySet();

Method (7) is used for retrieving the data from 2-D legacy Collection Framework variable. In other words, this method obtains set of keys from 2D legacy Collection Framework variable and we pass those key values to get(-) method.

* Example -

```
System.out.println(d);
// {(AP=Hyd), (KAR=Bang), (MH=Mum)}
```

d

AP	Hyd
KAR	Bang
MH	Mum

```
Enumeration en = d.keys();

while ( en.hasMoreElements())
{
    Object kobj = ~~d.get~~ en.nextElement();
    Object vobj = d.get(kobj);
    System.out.println( kobj + "  " + vobj);
}
```

5) java.util.Hashtable -

i) Hashtable is one of the pre-defined subclass of dictionary, so that all the methods of dictionary are inherited into Hashtable.

ii) Since, Hashtable is a concrete class, whose object can organise the data in the form of (key, value) pair by following Hashing mechanism.

iii) we can not determine in which order Hashtable object displays the data, Because the java programmer is not aware off which type of Hashing mechinism followed by Sun microsystems in the implementation of Hashtable class.

iv) The functionality of Hashtable is exactly similar to Hashmap. But Hashtable is one of the synchronised class and it ~~never~~ never allows null value

for (key, value) pair. whereas HashMap is one of the non-synchronised class and it allows us to place null values for (key, value) pair.

* Methods -

* Constructors -

1) Hashtable()
2) Hashtable (int)
3) Hashtable (int size, Float Fillratio)
loadratio.

Constructor (1) is used for creating an object of Hash table without specifying size and loadRatio.

Ex. Hashtable ht = new Hashtable();

Constructor (2) is used for creating an object of Hash-table by specifying the size and without specifying the loadratio.

Ex. Hashtable ht = new Hashtable (4);

ht

—	—	0
—	—	1
—	—	2
—	—	3

Constructor (3) is used for creating an object of Hashtable by specifying size & loadRatio.

The no. of entries into the Hashtable is equal to size of Hashtable, then automatically, jvm will call the pre memory allocation formula for restructuring the size of the array.

Ex. Hashtable ht = new Hashtable (4, 0.25)
size, loadratio

ht

10	1.5
20	2.5
30	1.5
40	3.5

The formula for pre-memory allocation is

size = size + ⌈ size * Fillratio ⌉

In this example after inserting four entries

into the Hashtable, the size of the Hashtable will be restructured by using pre-memory allocation formula.

* Before inserting fifth entry by using pre-memory allocation formula, one entry will be added to the Hashtable.

size = 4 + ⌈4 * 0.25⌉

= 5

* Similarly, after inserting fifth entry prememory allocation formula will be calculate as follows

Size = 5 + ⌈5 * 0.25⌉

= 5 + ⌈1.25⌉

= 7

ht

10	1.5	0
20	2.5	1
30	1.5	2
40	3.5	3
		4

ht

10	1.5
20	2.5
30	1.5
40	3.5
50	4.5

* Write a java program to implement the following.

State	Capital
A.P	Hyd
KAR	Bang
MH	mum

* Validations -

⇒ Accept the state and capital display its capital.

Ex. Banglore is the capital of karnataka.

⇒ If state is not found then display "No Idea".

⇒ If state is not passed then display "Plz enter the State

Hint: The above problem must be implemented by the concept of Hashtable.

```
// HtDemo.java
import java.util.*;

public class HtDemo
{
    public static void main(String args[])
    {
        Hashtable ht = new Hashtable();
        // Add states and capitals to Hashtable
        ht.put("A.P.", "Hyd");
        ht.put("KAR", "Bang");
        ht.put("MH", "Mum");

        // Print States and Caps
        Enumeration en = ht.keys();
        while(en.hasMoreElements())
        {
            Object kobj = ht.get en.nextElement();
            Object vobj = ht.get(kobj);
            System.out.println(kobj + " " + vobj);
        }

        // Accept the state from command prompt.
        if(args.length == 0)
        {
            System.out.println("Plz enter the State");
        }
```

```
        else
        {
            String state = args[0];
            if( state.toUpperCase() =
            Object capital = ht.get(state);
            if (capital != null)
            {
                System.out.println(capital + " is the capital
                of " + state);
            }
            else
            {
                System.out.println("NO IDEA !");
            }
        }
    }// main
}// HtDemo
```

* Limitations of Hashtable -

With the Hashtable we are unable to do the following

1) Unable to read the data from properties file / resource bundle file.

2) Unable to the data from Environmental variables.

3) Unable to develop Flexible java applications

The advantage of Hashtable is maintaining the user-defined data in the form (key, value) pair in the Heap Memory.

To eliminate the problems of Hashtable one is reco-

5) java.util.Properties -

Properties class is one of the subclass of Hashtable. with respect to properties class one can get the follo- -wing advantages.

i) It is able to read the data from Properties / Reso- -urce Bundle file.

ii) It is able to read the data from Environmental variables of the System.

iii) It allows to develop flexible Java applications.

iv) Like Hashtable, Properties class object also able to maintain user-defined data in the form of (key,value) pair in Heap memory.

v) Creating properties is nothing but a creating object of Properties class.

Ex. Properties p = new Properties();

* Definition -

A properties file is one which is organising the data in a file in the form of (key, value) pair. Properties file concept is also known as Resource Bundle file.

* Example - Student.prop / Student.rbf

stno=10
stname=satya
marks = 99.99

All properties files are residing in the secondary memory.

Technically, Properties files are the text file to be created in the Notepad or WordPad and it must be saved with some Filename with extension either .prop or ~~.rbf~~ with .rbf.

* Properties Class Profile -

* Constructor -

1) Properties()

2)

* Instance Methods -

2) public void load (FileInputStream)

3) public String getProperty (String)

Constructor ① is used for creating an object of the Properties class.

Ex. Properties p = new Properties();

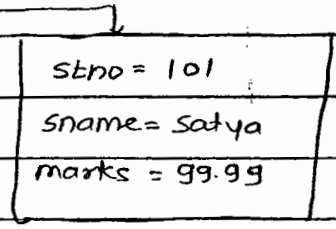
Here, object P resides in Heap Memory.

Method ② is used for transfering the content of properties file into the Properties class object. In other words, load() method transfers the contents of Properties files of secondary memory into properties class object of prim- -ary memory.

Ex. FileInputStream fis = new FileInputStream("student.prop");

Properties p = new Properties();

p.load(fis);



Here, (stno, sname, marks) are called key or the properties names.

(100, Satya, 99.99) are called values or the properties values.

method ③ is used for obtaining property values by passing the property names. or obtaining the values of value by passing key values.

Ex.

```
String sno = p.getProperty("stno");
String sname = p.getProperty("sname");
String marks = p.getProperty("marks");

S.O.P.("Student No : " + sno);
S.O.P.("Student Name : " + sname);
S.O.P.("Student marks : " + marks);
```

In the above methods, FileInputStream class is used for opening the file in read mode. If the specified file is not found then we get an exception called java.io.FileNotFoundException. Specified file is existing but the data of the file is corrupted then (such files are known as Corrupted Files) for this we get an exception called IOException.

* Steps for developing Flexible Java Application -

Step 1: Create the Properties file either in notepad or in wordpad and save it on some file name with as extension either with .prop or with .rbf.

Ex: emp.prop or emp.brf

eno = 1
ename = Satya

Step 2: Create an object of Properties class for holding the data of properties file.

Ex.

```
Properties p = new Properties();
```

Here, object P resides in the Primary memory.

Step 3 - Open the Properties file in the read mode with the help of with pre-defined class called FileInputStream class.

FileInputStream class contains the following constructor.

java.io.FileInputStream

public FileInputStream(String) throws FileNotFound Exception, IOExpeception.

Ex.

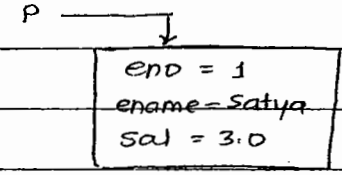
```
FileInputStream fis = new FileInputStream("emp.rbf");
```

Here, fis contains the address of "emp.rbf" file.

Step 4: Load the content of Properties file into properties class object.

Ex.

```
p.load(fis);
```



Step 5: Obtain the Property values or value of values by passing key value or property name.

Ex.

```
S.O.P("EmpNo = " + p.getProperty("eno"));
S.O.P("Emp Name = " + p.getProperty("ename"));
S.O.P("Emp Salary = " + p.getProperty("sal"));
```

Step 6: close the File which was opened in the read mode.

Ex.

```
Fis.close();
```

* Write a Java Program which illustrate the concept of Flexible Application development with Properties class and Properties File.

→ * Properties File -

```
// Emp.prop
emp = 1
eno = 1
ename = Sathya
sal = 3.0
```

* Main Application -

```
// FlexApp.java
import java.util.*;
import java.io.*;

public class FlexApp
{
    public static void main(String args[])
    {
        try
        {
            // Step-1 Creation of Properties File.
            // Step-2
            Properties p = new Properties();
            // Step-3
            FileInputStream Fis = new FileInputStream(args[0]);
```

```
// Step - 4
p.load(fis);
// Step - 5
S.O.P("Emp No = " + p.getProperty("eno"));
S.O.P("Emp Name = " + p.getProperty("ename"));
S.O.P("Emp Salary =" + p.getProperty("sal"));
// Step - 6
fis.close();
}
catch(ArrayIndexOutOfBoundException aoe)
{
    S.O.P("Please, enter the file name");
}
catch(FileNotFoundException fnf)
{
    S.O.P("Fide does not found");
}
catch(IOException ioe)
{
    S.O.P("Unable to read content of the file");
}
catch(Exception e)
{
    e.printStackTrace();
}
} // main
} // FlexApp.
```

* Write a java program which will display environmental variable system.

→ // EnvProp.java.

```
import java.util.*;

public class EnvProp
{
    public static void main(String args[])
    {
        Properties p = System.getProperties();
        Enumeration en = p.keys();
        while (en.hasMoreElements())
        {
            Object env = en.nextElement();
            Object evv = p.get(env);

            System.out.println(env + " ==> " + evv);
        }
    } // main
} // EnvProp
```

In the above program,
To list the environmental properties of the system, we use the following static method which is present in the System class

java.lang.System

```
public static Properties getProperties();
```

* java.util.Scanner -

• Scanner is one of the pre-defined class which is used for reading the data dynamically from the keyboard.

* Constructor -

```
Scanner(InputStream)
```

This constructor creates an object of Scanner class by taking an object of InputStream class. An object of InputStream class called in is created as a static data member in the System class.

Ex.

```
Scanner sc = new Scanner( System.in);
```

Here, the object in use the control of keyboard.

* Instance methods -

1) public byte nextByte ()
2) public short nextShort ()
3) public int nextInt ()
4) public long nextLong ()
5) public float nextFloat ()
6) public double nextDouble ()
7) public char nextChar ()
8) public boolean nextBoolean()
9) public string nextLine ()

methods ① to ⑧ are used to reading fundamental values from the keyboard.

method ⑨ is used for reading any kind of data in the form of String. data.

* Write a java program which will accept two values dynamically from the keyboard & compute sum.

→

```
// ScannerDemo.java
import java.util.Scanner

public class ScannerDemo
{
    public static void main( String args[])
```

```
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter First No = ");
        int num1 = sc.nextInt();

        System.out.println("Enter Second No. = ");
        int num2 = sc.nextInt();

        System.out.println("Sum of is = " +(num1 +num2));
    } // main
} // ScannerDemo.
```

* Multi - Threading *

Date- 08-May-2011

i) Multithreading is one of the distinct facility available in java programming.

ii) The basic aim of multithreading is to achieve the concurrent Execution.

iii) A Flow of control is known as Thread.

iv) The basic uses of the Thread is to execute user/programmer defined methods.

v) If any java program is containing multiple Flow of Controls then that java program is known as multiThread

vi) The languages like C, C++, Pascal, COBOL etc are treated as Single Threaded Modelling Languages.

vii) All the single Threaded modelling language execution environment contains single Flow of Control and these

languages does not contain any pre-defined library for developing multiThreaded Application.

viii) with single Threaded modelling language we are able to achieve sequential execution but not the concurrent execution.

ix) The languages / technologies like Java and .Net are treated as multithreaded modelling languages.

x) All the multithreaded modelling languages execution environmet contains multiple flow of controls & provides the facility to create multiple flow of controls.

xi) with multithreaded modelling languages one can achieve Sequential & concurrent executions & these languages contains an effective library for devloping multithreading application

In Java programming we use the following API for development of multithreading Applications

a) java.lang.Thread (c)
b) java.lang.Runnable (I)

In Java programming, creating a flow of control is nothing but creating an object Thread either java.lang.Thread class or java.lang.Runnable (I) (indirectly)

The realtime implementation of multithreading concept is that to develop real world servers. The following table gives Server s/w name & server vendor

Server S/w Name	Server Vendor Name
Tomcat	Apache Jakarta S/w Foundation
Weblogic	BEA
Websphere	IBM
Oracle 10g	Oracle Corporation
GlassFish	Sun microSystems
JRun	Macro Media
Pramathi	Pramathi S/w Pvt Ltd. Hyd.

All the above real world servers developed by the Server vendors in Java language ~~with~~ with the concept of multithreading.

Whenever we write the java program, by default its contains two types of Threads. They are.-

i) Foreground / Child Thread
ii) Background / Parent Thread.

Foreground Threads are those which are created by the java programmer for executing the user / programmer defined method or logic of the user defined methods.

Background Thread is one which is always monitoring the execution status of Foreground Threads.

By default, there exist single foreground thread and single background thread. Programmatically, one can permitted to create multiple foreground Threads and recommended to have single background per Java Program.

Multithreading is one of the specialized forms of multitasking of Operating System.

Therefore, if we write any java program without th

multithreading concept., the user/programmer defined methods executing sequential. If the same program is rewritten with the concept of multithreading, those methods will be executed concurrently. So in java programming one can achieve sequential as well as concurrent execution.

Q] How do you justify, "each & every java program is multithreaded" ?

→ Whenever we execute any java program, the logic of the Java program executed by one of the thread known as <u>foreground Thread</u>.

To monitor the execution status of foreground Thread, internally there exist one more thread known as Background Thread.

Hence, every java program is containing multiple Threads Therefore, every java program is multithreaded

Q] what are the differences betn Program and Process?

Program	Process
i) Set of optimited instructions is known as Program.	i) A program is under execution is known as Process.
ii) Programs always resides in Secondary Memory (HDD).	ii) Processes always resides in Primary Memory (RAM).
iii) Programs resides permanantly is secondary memory until we delete	iii) Process exists in ~~Ram~~ main Memory for limited span of time.
iv) In programming point of view,	iv) In programming point of view,

Date- 09-May-2011

* Process based Applns and Thread Based Applns -

As a software engineer in the industry, we develop two types of Applications. They are -

i) Process based Application
ii) Thread based Application.

	Process based Appln	Thread based Appln.
i)	PBA are those which are containing single flow of control.	i) TBA are those which contains multiple flow of Controls.
ii)	All C, C++, Pascal, COBOL etc applications are treated as PBA.	ii) All Java and .Net Applns are treated as TBA.
iii)	In PBA, context switch is more.	iii) In TBA applications, context switch is very less.
iv)	In PBA, for each & every sub-program, there exist a seperate address space.	iv) In TBA, for each & every subprogram, there exist a single Address Space that is irrespective of no. of subprograms there exist only one address space.
v)	All PBA, are treated as a Heavy weight components.	v) All TBA are treated as a Light weight components.
vi)	PBA provides only sequential execution but not concurrent execution.	vi) TBA provides both sequential and concurrent executions.

* Address space / workspace -

The amount temporary memory space created by o.s. on stack memory for the execution of the method is known as Address space / workspace.

* Context Switch -

The mechanism of changing the control of CPU from one address space to another address space in the main memory of a computer is known as Context Switch.

For the best programs, context switch must be less.

* Heavy weight & Light weight Components -

Heavy weight components are those which will take more amount of processing time whereas light weight components are those which takes less processing time.

* CPU Burst Time -

The amount of time taken by the Thread from the CPU to execute the method is known as CPU Burst Time.

Q] How do you justify, "every TBA is the PBA" ?

→ Whenever we execute TBA, TBA will be entered into main memory and one main process will be created by the JVM. The main process in turns creates multiple sub-processes. In Java programming point of view, the main process is called as Background Thread / Thread Group and sub-processes are known as ForeGround Thread.

Every ~~process~~ PBA cannot be a Thread Based Appln.

Date - 10- May - 2011

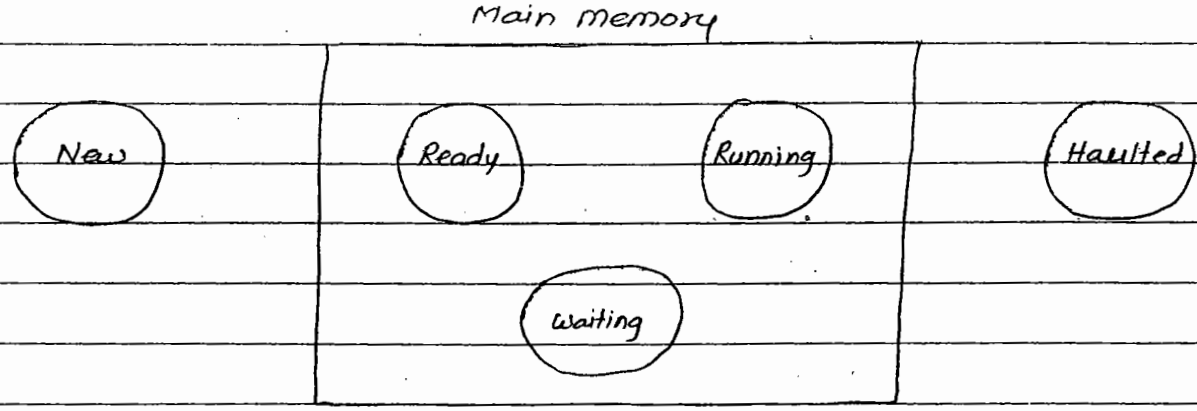
* States of A Thread or Life Cycle of a Thread *

Whenever we write a multithreading program, there exist many no. of foreground Threads. During the execution of foreground Threads, they may undergo various states of the Thread.

States of the Thread are classified into five types. They are -

1) New State
2) Ready State
3) Running State
4) waiting State
5) Haulted State

* State Chart Diagram -



1) New State -

A new state is one in which thread is created and it is about to enter into main memory.

2) Ready State -

A Ready state is one in which thread is ente-red into main memory, address space is created and first time waiting for the CPU.

3) Running State -

A Running state is one in which Thread is under the control of CPU.

4) Waiting State -

A Thread is said to be in waiting state, if and only if it satisfies any one of the following factors.

a) For the remaining CPU Burst Time.
b) Suspending the currently executing Thread.
c) Making the currently executing Thread to go to sleep by specifying amount of sleep time in terms of milliseconds (1 second = 1000 milisec)
d) Making the currently executing thread to go to wait by specifying amount of waiting time in terms of milliseconds.
e) Making the currently executing thread to go to wait without specifying amount of waiting time.
f) Making the threads to join after there com-plete execution.

5) Haulted State -

Haulted State is one in which a Thread has completed its total execution.

Therefore, as long as Thread is present in Ready, Running & waiting states, whose execution status is true. And these states are known as In-Memory States.

As long as Thread is present in New & Haulted States, whose execution status is falce. And these state are known as Out-Memory States.

Q] How do you Justify, "All the Threads of multithreading Application executes concurrently"?

→ If the Threads are executing Hour by Hour then it is known as Sequential.

If the Threads are executing minute by minute and second by second then it is also treated as a Sequential. Because Human being calculations can identify the difference of Hours, Minutes & seconds.

If the Threads are executing by milliseconds by milliseconds then we can say this execution as a Concurrent Execution. Because identifying the differences of millisecond is beyond the scope of Human Being calculations.

Hence, all the Threads in multithreading applications executing concurrentely with the differences of the milliseconds by following Round Robin Algorithm.

* Number of ways to Create a Thread *

In order to create a Flow of control (Thread) in java we have two ways. They are

A) By using java.lang.Thread Class

B) By using java.lang.Runnable Interface

A) By using java.lang.Thread class -

To create a Flow of control in Java by using the Thread Class, we have three ways.

1) Directly with new operation.

Ex. Thread t1 = new Thread();

2) Using factory method -

Ex. Thread t2 = Thread.currentThread();

3) An object of subclass of Thread is nothing but an object of Thread class.

Ex.

```
public class Th extends Thread
{
    =
}

Th t3 = new Th();
```

Here, t3 is an object Th class and Th is the subclass of Thread class. Therefore, t3 is indirectly an object of Thread class.

* java.lang.Thread Class Profile -

* Data Members -

1) public static final int MAX_PRIORITY = 10;
2) public static final int MIN_PRIORITY = 1;
3) public static final int NORM_PRIORITY = 5;

The above data members are known as Thread Priority modifiers. The default Thread Priority modifier value is NORM_PRIORITY.

Q] How do you display the Thread Priority Modifier values?

→

```
Class ThPriority
{
    public static void main (String arg[])
    {
        s.o.p ("Value of MAX_PRIORITY =" + Thread.MAX_PRIORITY);
        s.o.p ("value of NORM_PRIORITY =" + Thread.NORM_PRIORITY);
        S.o.p ("Value of MIN_PRIORITY =" + Thread.MIN_PRIORITY);
} //main
```

* Constructors -

1) Thread () -

This constructor is used for creating an object of Thread class without specifying user defined name of the Thread.

If we create n no. of flow of controls w.r.t. default constructor then whose default name starts with Thread-0, Thread-1 ---- Thread-N. It is highly recommended for the Java Programmer to give the name to the Thread for effective processing of user-defined methods.

Ex. Thread t1 = new Thread();

2) Thread (String) -

This constructor is used for creating an object of Thread class by specifying the name of the Thread.

Ex: Thread t1 = new Thread ("JavaTh");

t1 ↓ Reference to Thread. "JavaTh" ↑ Name of the Thread.

3) Thread (Runnable) -

This constructor is used for converting Runnable Interface object into Thread class object for making use of all the methods of Thread class. (specially start() method) to enter into the run method of Runnable Interface. with this constructor, we unable to given name of the user-defined Thread.

4) Thread (Runnable, String) -

This constructor functionality is exactly similar to the above constructor, But with this constructor we are able to give name for the User defined Thread.

Ex: Th t = new Th(); or

Runnable t = new Th();

Thread t2 = new Thread (t1, "JavaTh");

* Instance methods -

5) public final void setName(String) -

6) public final String getName() -

The above methods are used for setting the name to thread and getting the name of Thread.

Ex. Thread t1 = new Thread();
String tname = t1.getName();
S.O.P(tname); // Thread-0

t1.setName("JavaTh");
~~S.O.P~~(tname = t1.getName();
S.O.P(tname); // JavaTh

7) public final void setPriority(int) -

8) public final int getPriority() -

The above methods are used for setting and getting the Priority of the Thread.

Ex. Thread t = new Thread();
int pri = t.getPriority();
S.O.P(pri); // 5 → NORM_PRIORITY

t.setPriority(Thread.MAX_PRIORITY);
pri = t.getPriority();
S.O.P(pri); // 10 → MAX_PRIORITY

9) public final void suspend() -

This method is used for suspending the currently executing Thread. When the Thread is suspended

the thread will be enter into the waiting state from running state by saving temporary result of thread into the PCB.

10) public final void resume() -

This method is used for transfering the thread from waiting state to ~~running~~ ready state. When the thread is resumed it starts continuing its execution where it suspended previously by retrieving temporary result from PCB.

11) public final void start() -

This method is used for transferring the thread from new state to ready state. Once we call the start method upon the thread object, it is not only entering into ready state but also automatically calling run() method by providing internal services of multithreading. Some of the internal services of the multithreading are providing states, concurrent execution, synchronisation and inter thread communication etc.

12) public void run() -

This method is used for defining the logic of the thread. In other words, the business logic of the Java program can be written in the form of user-defined methods and those methods can be called as a part of run() method. It is highly recommended for the java programmer not to call the run method direct

public void run() method is originally defined in Thread class with null body. To provide the logic of the Thread we need to choose the class extends Thread class and override run() method by writing our own logic.

Ex.

```
Class ThreadDemo extends Thread
{
    public void run()
    {
        - - - -
        // Logic of Thread
        - - - -
    }
} // ThreadDemo

ThreadDemo th = new ThreadDemo();
th.start();
```

13) public boolean isAlive()

This method is used for checking the execution status of a Thread. This method returns true provided the Thread is in ready, running and waiting state. It returns false provided the Thread is in new and hallted states.

Ex.

```
ThreadDemo th = new ThreadDemo(); //New State
boolean estate = th.isAlive();
S.O.P(estate); // False

th.start(); // Ready state
estate = th.isAlive(); // True
S.O.P(estate); // True
```

14) public final void join() - throws InterruptedException

This method is used for joining the threads which are completed there execution as a single unit

for hand overing them to Garbage collector. This approach improves internal performance of multi-threading applications. This method throws a pre-defined exception called java.lang.InterruptedException

This exception will not be occuring in single machine But there is a posibility of occuring this exception in Busy environment like Client-Server environment.

* When InterruptedException occurs -

Assume a multithreaded program starts exe--cuting with n- no. of threads. N-1 no. of threads completed their execution and joined and still nth thread is executing. During this period of time, due to mic- memory management of the Server, n-1 comple-ted threads are taken by Thread Group and trying to hand over them to Garbage Collector at this situation JVM generates a pre-defined exception called java.lang.InterruptedException.

Ex.

```
                ≡
try
{
    t1.join();
    t2.join();
    t3.join();
}
catch( InterruptedException ie)
{
    System.err.println("Problem in Thread Execution")
}
```

15) public final void stop() -

This method is used for stopping the thread which is currently executing. When the Thread is stopped

its execution, the Thread will be entered into Haulted state. When stopped Thread is restarted its execution, it starts its execution from the beginning.

(6) public

* static methods -

1) public static final Thread currentThread () -

This method is used for obtaining the Threads which are by default running in Java environment.

Ex. Thread t1 = new Thread().currentThread();

S.o.p(t1); // Thread [main, 5, main]

FGT Priority TGP

t1.setName("JavaTh");

S.o.P(t1); // Thread [JavaTh, 5, main]

2) public static final void sleep (Long ms) throws InterruptedException

This method is used for making the Thread to sleep by the for a period of time in terms of milliseconds. Once the sleep time is completed, automatically the thread will be entered into ready state from waiting state.

This method throws a pre-defined exception called java.lang.InterruptedException.

* When InterruptedException occurs -

Assume that some of threads of current java program is sleeping in the address space of the main memory of server. The threads related to new program are trying to sleep in the same address space where current threads are sleeping because of the mis-memory

management of the server, w.r.t. the current thread JVM raises an exception called java.lang.InterruptedException.

Ex.

=

```
try
{
    Thread.sleep(10000);
}
catch (InterruptedException ie)
{
    System.err.println("Problem in Thread Execution");
}
```

Date:- 13-May-2011

* write a java program which will print prelimiinary information about the Thread. such as Default name of Thread which are running, execution status of Default Thread, Default Name of U-Defined Thread, execution status UD Thread, printing Thread Priority Modifier values - etc.

→

```
// ThreadDemo.java
public class ThreadDemo
{
    public ~~void~~ static void main(String args[])
    {
        Thread t1 = Thread.currentThread();
        S.O.P(~~t1~~"Default Thread Name:="+t1); // [main,5,main]
        t1.setName("JavaTh");
        S.O.P("Default Threads names after modification="+t1);
        // Thread [JavaTh,5,main]
```

```
System.out.println("Execution status = " + t1.isAlive());
    // True
Thread t2 = new Thread(); // New State
System.out.println("Default Name of t2 = " + t2.getName());
S.O.P("Execution status of t2 = " + t2.isAlive()); // False
S.o.P("Val. of MAX_PRI = " + Thread.MAX_PRIORITY);
S.o.P("Val. of NORM_PRI = " + Thread.NORM_PRIORITY);
S.oP("Val. of MIN_PRI = " + Thread.MIN_PRIORITY);
} // main
} // ThreadDemo
```

* Internal Flow of Threads - (Thread Model)

Whenever we write a multithreading application there is the possibility that many no. of foreground Threads are created. When the java program start executing the Thread Group & Foreground Threads will undergo the following sequence of steps or The following sequence of steps gives how Threads are internally executing.

1) Java program starts executing.

2) JVM creates a Thread Group (TGP) which resides in main method).

3) TGP creates Foreground Threads (FGT which always resides in run() method).

4) TGP dispatches the FGPT to their respective run() methods.

5) Foreground Threads executes run() method & gives the result to Thread Group either at once or one by one.

6) Thread Group receives the result from Foreground Threads either at once or one by one.

7) Thread Group gives the result back to programmer / normal user.

8) Thread Group collects the foreground Threads and Hand over it to Garbage Collector.

9) JVM collects Thread Group & Hand over to Garbage Collector.

10) Java Program stops executing.

* Write a java program which will print one to Ten numbers after each & every second with the concept of Threading.

→

```
// MyThread.java
public class MyThread extends Thread
{
    public void run()
    { try
      {
          for(int i=1; i<=10; i++)
          {
              S.O.P("value of i=" +i);
              Thread.sleep(1000);
          }
      }
```

```
        catch ( InterruptedException ie)
        {
            System.err.println ("Problem in Thread Execution");
        }
    } // MyThread

    public class ThreadDemo
    {
        public static void main (string args[])
        {
            MyThread t = new MyThread(); // New State
            S.O.P ("Status before start = " + t.isAlive());
            t.start();
            S.O.P ("Status after start = " + t.isAlive());

            try
            {
                Thread.sleep (10002);
            }
            catch (InterrupteException ie)
            {
                System.err.println ("Problem in Thread Exec.");
            }
            S.O.P ("Status of Thread t = " + t.isAlive());
        } //main
    } // ThreadDemo
```

B) By using java.lang.Runnable Interface -

Runnable is the one of the predefined interface which is used for developing multithreading application. This interface contains the following abstract method.

```
public abstract void run();
```

The above method is used for defining the logic of the Thread.

To provide the logic of the Thread b the run method of Runnable Interface, we must override the run method of Runnable Interface within derived class.

Ex.

```
Class Th implements Runnable
{
    public void run()
    {
        // logic of Thread.
    }
}
```

```
Th t = new Th(); or // Invalid.
Runnable t1 = new Th();
t1.start(); // Invalid.
```

Because start method is niether defined in Th ornor it is inherited from Runnable interface hence it is mandatory to java programmer to convert an object of Runnable Interface to Thread class object for making use of all the methods of Thread class (Specially start() method) to enter into run() method of Runnable Interface.

Ex. `Thread t1 = new Thread(t); // valid.`

* Write a Java Program which will print one to Ten numbers by using Runnable Interface by every second.

→

```
class ThreadDemo implements Runnable
{
    public void run()
    {
        for(int i=1 ; i<=10; i++)
            System.out.println("value of i =" +i);
    }
} // ThreadDemo

public class MyThread
{
    public static void main(String args[])
    {
        ThreadDemo t = new ThreadDemo();
        Thread t1 = new Thread(t, "JavaTh");
        S.O.P("Name of t1 =" + t1); // Thread[JavaTh, 5, main]
        S.O.P("Status of t1 =" + t1.isAlive()); // False
        t1.start();
        S.O.P("Status of t1 =" + t1.isAlive()); // True
    } //Main
} // MyThread
```

* Write a Java Program which will create multiple Threads in which one Thread will do sum and another Thread will do substraction of two number.

→

```
class ThreadSum extends Thread
{
    int a,b;
```

```
    void set (int a, int b)
    {
        this.a = a;
        this.b = b;
    }
    public void run()
    {
        int c = a+b;
        S.O.P("sum is " + c);
    }
} // Threadsum.

class ThreadSub implements Runnable
{
int a, b;
void set (int a, int b)
{
    this.a = a;
    this.b = b;
}
public void run()
{
    int c = a-b;
    S.O.P("substraction is " + c);
}
} // ThreadSub.

public class ThreadMain
{
    public static void main (String args[])
    {
        ThreadSum ts = new ThreadSum();
```

```
Thread ts1 = new
ThreadSub ts1 = new ThreadSub();
Thread ts2 = new Thread(ts1);

int x = Integer.parseInt(args[0]);
int y = Integer.parseInt(args[1]);
ts.set(x,y); ts.start();
ts1.set(x,y); ts1.start();

// Join completed Threads as single unit.
try
{
    ts.join();
    ts1.join();
}
catch(InterruptedException ie)
{
    ie.printStackTrace();
}
} // main
} // ThreadMain
```

Note: In slw development we may came acerass N no. of isolated or independent task. To solve these task concurrently, it is mandatory for Java programmer to write 'N' no. of subclasses by overriding 'N' no of run() method by inheri-tting from either java.lang.Thread Class or from java.lang.Runnable Interface. Each run() method is containing a seperate logic for solving a specific requirement.

* Write a java programming which will display a scrolling message or Implement Banner Animation.

```
import java.awt.*;
import java.applet.Applet;

/*<applet code="AnimDemo" height="300" width="200">
</applet> */

public class AnimeDemo implements Runnable   extends Applet
{
    String msg = "WELCOME TO JAVA";
    public void init()
    {
        setBackground(Color.cyan);
        setforeground(Color.red);
    }
    public void start()
    {
        Thread t = new Thread(this); // this refers to current
        // class obj which implements Runnable Interface &
        // converts it into Thread class obj.
        t.start();
    }
    public void paint(Graphics g)
    {
        Font F = new Font("Arial", Font.BOLD, 60);
        g.setfont(F);
        g.drawString(msg, 100, 100);
    }
    public void run()
    {
        try
        {
```

```
try
{
    while(true)
    {
        char ch = msg.charAt(0);
        msg = msg.substring(1, msg.length());
        msg = msg + ch;
        repaint();
        Thread.sleep();
    }
}
catch(InterruptedException ie)
{
    System.err.println("Problem in Thread");
}
} // run
} // AnimDemo
```

* Write a java program which will increment the number the automatically by using Threads after each & every second.

→
```
// AutoInc.java
import java.awt.*;
import java.applet.*;
/* <applet code="AutoInc" width=300 height=500>
</applet> */
public class AutoInc extends Applet implements Runnable
{
    int i=0;
    public void init()
    {
        // same as before
    }
```

```
public void start ()
{
    Thread t = new Thread (this);
    t.start();
}

public void paint ( Graphics g)
{
    Font f = new Font ("Arial", Font.BOLD, 60);
    g.setFont (f);
    g.drawString ( String.valueOf (i), 100, 100);
}

public void run()
{
    try
    {
        while (true)
        {
            ++i;
            repaint();
            Thread.sleep (1000);
        }
    }
    catch ( InterruptedException ie)
    {
        ie.printStackTrace();
    }
}
} // AutoInc
```

* Write a java program which will implement Digital Clock functionality. in the format HH:mm:ss.
(Homework).

Date - 16-May-2011

* Thread Synchronisation *

Thread Synchronisation is one of the distinct facility used in multithreading to eliminate inconsistent results.

In Operating System's point of view, this concept was known as Mutual Exclusion Principle.

* Definition - The process of allowing only one Thread at a time among multiple Threads into the area which is sharable to perform read and write operations. This is known as Synchronisation.

In other words, if anything is sharable by multiple threads then there is possibility of getting inconsistent result. To eliminate this inconsistency, it is mandatory to the Java Programmer to make sharable thinks as Synchronised by using Synchronised keyword.

* Necessity of Synchronisation -

Let us assume, there exist a sharable variable bal. which contains an initial value 0. Let us assume, there exist two threads t1 & t2, want to update sharable variable value by 10 & 20 respectively. Start both threads for execution and after completion of t1 & t2

threads execution, the result of sharable variable value is either 10 or 20. But not 30 which is an inconsistent result.

To eliminate this inconsistent result, we apply the concept of Synchronisation.

* Synchronisation Concept Applied -

Let us assume, synchronisation concept applied on the above problem. Both the threads are started (t1 started first and t2 started later with difference of milliseconds). When t1 started first, the value of the sharable variable given to the thread t1 by the jvm and balance variable value will be locked. Meantime, if thread t2 is trying to access the balance variable value then the thread t2 will be made wait until the thread t1 completes its execution. The thread t1 completed it execution, the bal variable will be unlocked by the JVM, given the bal variable value to thread t2 and once again variable bal will be locked. The thread t2 completed its execution and finally the bal variable will be unlocked. The value of the bal variable is 30, which is a consistent result.

Hence, during the synchronisation concept the process locking & unlocking will be continued until the threads are completed their execution. Because of these locking & unlocking process, consistency of the sharable variables will be achieved.

* Thread Synchronisation Techniques *

To achieve the Synchronisation concept, in the multithreading we have two thread synchronisation techniq

1) Synchronised methods.
2) Synchronised Blocks.

1) Synchronised methods -

If any method accessible by multiple threads then that method may generate inconsistent results. To eliminate this inconsistent results, the ordinary method definition must be preceded by a keyword called synchronised.

Synchronised methods are divided into two types. They are

a) Synchronised Instance methods
b) Synchronised Static methods.

a) Synchronised Instance methods -

If an ordinary instance method is accessible by multiple threads., then there is a possibility that an ordinary instance method may generate inconsitent result w.r.t. the threads.

To eliminiate this inconsistent result, an ordinary instance methods definition must be made as synchronised by using synchronised keyword.

Syntax -

```
synchronised <ret_type> method_name (<list_of_params>)
{
    // Block of statements
}
```

Once an ordinary instance method is synchronised, then corresponding class object will be locked

Ex: class Account
{

```
int balance = 0;

Synchronised void deposit (int amount)
{
    balance = balance + amount;
    S.O.P("Current Balance = " + balance);
}
```

Once the thread is entered into deposit method, an object of account will be locked by JVM.

b) Synchronised Static Methods -

If an ordinary static method is sharable or accessible by multiple threads then there is possibility that an ordinary static method may generate inconsistent results w.r.t threads.

To eliminate this inconsistent result, a definition an ordinary static method must be made as synchronised by using synchronised keyword.

Syntax:

```
Synchronised static <ret.type) method.name (<list_of_params>)
{
    // Block of statements
}
```

Once an ordinary static method is synchronised then corresponding class will be locked.

Ex.

```
class Account
{
    static int balance = 0;
```

```
synchronised static void deposit(int amount)
{
    balance = balance + amount;
    S.O.P("current Balance = " + balance);
}
}
```

In any thread is entered into above deposit method, automatically JVM locks Account class.