Write a java program which illustrates the concept of HashMap?

Answer:

```
import java.util.*;
class hmtm
{
	public static void main (String [] args)
	{
		HashMap hm=new HashMap ();
		System.out.println ("CONTENTS OF hm = "+hm);
		System.out.println ("SIZE OF hm = "+hm.size ());
		hm.put (new Integer (10), new Float(129.97f));
		hm.put (new Integer (1), new Float(143.93f));
		hm.put (new Integer (100), new Float(99.8f));
		System.out.println ("CONTENTS OF hm = "+hm);
		System.out.println ("SIZE OF hm = "+hm.size ());
		Set s=hm.entrySet ();
		Iterator itr=s.iterator ();
		while (itr.hasNext ())
		{
			Map.Entry me= (Map.Entry) itr.next ();
			Object kobj=me.getKey ();
			Object vobj=me.getValue ();
			System.out.println (vobj+"-->"+kobj);
		}
	}
};
```

Write a java program which illustrates the concept of TreeMap?

Answer:

```
import java.util.*;
class tmhm
{
	public static void main (String [] args)
	{
		TreeMap tm=new TreeMap ();
		System.out.println ("CONTENTS OF tm = "+tm);
		System.out.println ("SIZE OF tm = "+tm.size ());
		tm.put (new Integer (10), new Float(129.97f));
		tm.put (new Integer (1), new Float(143.93f));
		tm.put (new Integer (100), new Float(99.8f));
		System.out.println ("CONTENTS OF tm = "+tm);
		System.out.println ("SIZE OF tm = "+tm.size ());
		Set s=tm.entrySet ();
		Iterator itr=s.iterator ();
		while (itr.hasNext ())
		{
			Map.Entry me= (Map.Entry) itr.next ();
			Object kobj=me.getKey ();
			Object vobj=me.getValue ();
			System.out.println (vobj+"-->"+kobj);
		}
	}
};
```

Day - 70:

Legacy collection framework:

When SUN Micro Systems has developed java, collection framework was known as data structures. Data structures in java were unable to meet industry requirements at that time. Hence data structures of java was reengineered and they have added 'n' number of classes and interfaces and in later stages the data structures of java is known as new collection framework we have one interface and classes.

Interface:

We have only one interface, namely java.util.Enumeration. This interface is used for extracting the data from legacy collection framework classes.

Classes:

As a part of legacy collection framework we have the following essential classes: V**ector, Stack, Dictionary, Hashtable** and **properties**. Here, Vector and Stack belongs to one dimensional classes whereas Dictionary, Hashtable and Properties belongs to two dimensional classes.

What is the difference between normal collection framework and legacy collection framework?
Answer: All the classes in the normal collection framework are by default belongs to **non-synchronized** classes whereas all classes in legacy collection framework are by default belongs to **synchronized** classes.

Vector:

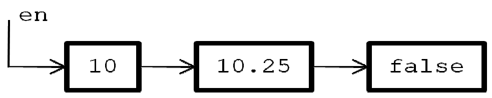
Its functionality is exactly similar to ArrayList but Vector class belongs to synchronized whereas ArrayList belongs to non-synchronized class.

Creating a Vector is nothing but creating an object of java.util.Vector class.

Vector API:

Constructors:
```
Vector (); →1
Vector (int size); →2
```

Instance methods:
```
public int size ();→3
public void addItem (Object obj); [old] →4
public void addElement (Object obj); [new] →5
public void addItem (int pos, Object obj); →6
public void addElement (int pos, Object obj); →7
public Object getItem (int pos); →8
public Object remove (int pos); →9
public void remove (Object obj ); →10
public void removeAll ();→11
public Enumeration elements ();→12
```
The methods 4,5,6 and 7 are used for adding an object to the vector either at end or at specified position. Method-12 is used for extracting the data from vector object.
```
Enumeration en=v.elements ();
```



```
While (en.hasMoreElements ())
{
       Object obj=en.nextElement ();
       System.out.println (obj);
}
```

Write a java program which listed the concept of Vector?

<u>Answer</u>:

```
import java.util.*;
class vector
{
       public static void main (String [] args)
       {
              Vector v=new Vector ();
              v.addElement (new Integer (10));
              v.addElement (new Float (100.37f));
              v.addElement (new Boolean (true));
              v.addElement ("K.V.R");
              System.out.println ("SIZE = "+v.size ());
              System.out.println ("CONTENTS = "+v);
              Enumeration en=v.elements ();
              while (en.hasMoreElements ())
              {
                     Object val=en.nextElement ();
                     System.out.println (val);
              }
       }
};
```

<u>Methods in Enumeration method</u>:

```
public boolean hasMoreElements (); →1
public Object nextElement (); →2
```

Method-1 is used for checking weather we have next elements or not. This method returns false when we have next element otherwise it returns false. Method-2 is used for obtaining next element. Method-2 can be used as long as method-1 returns true .

<u>Stack</u>: Stack is the sub-class of Vector class. The basic working principal of Stack is Last In First Out.

<u>Stack API</u>:

<u>Constructors</u>:

```
Stack ();
Stack (int size);
```

<u>Instance methods</u>:

```
public boolean empty (); →1
public void push (Object); →2
public Object pop (); →3
public Object peek (); →4
public int search (Object); →5
```

Method-1 returns true when the Stack does not contain any elements otherwise false. Method-2 is used for inserting an object into the Stack. Method-3 is used to remove top most elements permanently. Method-4 is used to retrieve top most elements without removing. Method-5 returns relative position of the element, if it found otherwise returns -1.

Write a java program which illustrates the concept of Stack?

Answer:

```
import java.util.*;
class stack
{
	public static void main (String [] args)
	{
		Stack st=new Stack ();
		System.out.println ("IS STACK EMPTY ? "+st.empty ());
		System.out.println (st);
		st.push (new Integer (10));
		st.push (new Integer (20));
		st.push (new Integer (30));
		st.push (new Integer (40));
		System.out.println (st);
		System.out.println ("TOP MOST ELEMENT = "+st.peek ());
		System.out.println (st);
		System.out.println ("DELETED ELEMENT = "+st.pop ());
		System.out.println ("MODIFIED STACK = "+st.peek ());
		System.out.println ("IS 10 FOUND ? "+st.search (new Integer (10)));
		Enumeration en=st.elements ();
		while (en.hasMoreElements ())
		{
			Object obj=en.nextElement ();
			System.out.println (obj);
		}
	}
};
```

Day - 71:

Dictionary:

Dictionary is an abstract class, whose object allows to retrieve to store the data in the form of (key, value) pair. An object of Dictionary never allows duplicate values as key objects and null values.

Hashtable:

Hashtable is the concrete sub-class of duplicates and where object allows us to store in the form of (key, value) pair. Hashtable object organizes its data by following hashing mechanism. We cannot determine in which order the Hashtable displays its data.

Hashtable API:

Constructor:

```
Hashtable ();
```

Instance methods:
```
public boolean put (Object kobj, Object vobj); →1
public void remove (Object kobj); →2
public Object get (Object kobj); →3
public Enumeration keys (); →4
```
Method-1 is used for obtaining value object by passing key object. If the key object is not found then the value object is null. Method-2 is used for extracting key objects from Hashtable.

Write a java program which illustrates the concept of Hashtable?

Answer:
```
import java.util.*;
class hashtable
{
	public static void main (String [] args)
	{
		Hashtable ht=new Hashtable ();
		ht.put ("AP","Hyd");
		ht.put ("Orissa","Bhuvaneshwar");
		ht.put ("Karnatake","Bng");
		ht.put ("TN","Chennai");
		ht.put ("Bihar","Patna");
		System.out.println (ht);
		Enumeration en=ht.keys ();
		while (en.hasMoreElements ())
		{
			Object k=en.nextElement ();
			Object v=ht.get (k);
			System.out.println (k+"   "+v);
		}
		if (args.length==0)
		{
			System.out.println ("PASS THE STATE");
		}
		else
		{
			String st=args [0];
			Object cap=ht.get (st);
			if (cap==null)
			{
				System.out.println (cap+" IS THE CAPITAL OF "+st);
			}
		}
	}
};
```

Properties class:

Properties is the sub-class of Hashtable class. Properties class object is used for reading of maintaining system environmental variables and reading the data from resource data file or properties file.

Properties API:

Constructor:
```
Properties ();
```

Instance Methods:
```
public void setProperty (Object kobj, Object vobj);
public Object getProperty (Object kobj);
public void load (InputStream);
```

Write a java program which illustrates the concept of Properties class?

Answer:
```
import java.util.*;
import java.io.*;
class properties
{
	public static void main (String [] args)
	{
		try
		{
			Properties p=new Properties ();
			FileInputStream fis=new FileInputStream ("x.prop");
			p.load (fis);
			Object vobj1=p.get ("dno");
			Object vobj2=p.get ("dname");
			Object vobj3=p.get ("pwd");
			System.out.println ("USER NAME : "+vobj2);
			System.out.println ("DEPT NUMBER : "+vobj1);
			System.out.println ("PASSWORD : "+vobj3);
		}
		catch (Exception e)
		{
			System.out.println (e);
		}
	}
};
```

Day - 72:

NETWORK PROGRAMMING:

Collection of interconnected autonomous or non-autonomous computers is known as network.

Autonomous represents independent processing power whereas non-autonomous represents dependent processing.

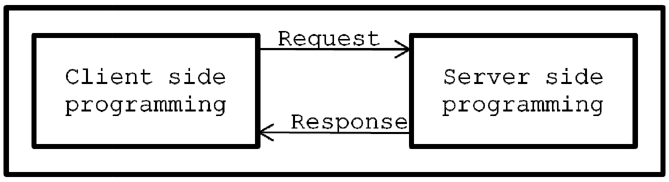
Aims of networking:

1. Fastest and reliable communication can be achieved.
2. Communication cost is so less.
3. Data can be shared either locally (centralized application) or globally (distributed application).

As a part of networking we write two types of programs. They are **client side programming** and **server side programming**.

- A *client side programming* is one which always makes a request to get the service from the server side program.

- A *server side programming* is one which receives client request, process the request and gives response back to the client.



In order to exchange the data between the client and server we must have a protocol. A protocol is the set of rules which exchange the data between client and server programs.

When we are developing any client and server application we must follow certain steps at client side and server side.

Steps to be performed at client side:

1. Connect to the server side program. To connect to the server the client side program must pass server name or domain naming services or IP address.
2. Make a request by passing data.
3. After performing some operations at server side, the client side program must receive the response from the server side.
4. Read the response data which is sent by the server.

Steps to be performed at server side:

1. Every server side program must run at certain port number (*a port number is a logical numerical ID at which logical execution of the program is taking place*).
2. Every server must have a physical name to access the services or programs which are running. Server can be access in two ways. **On name (localhost by default) called DNS** and **on IP address (127.0.0.1 default)**.
3. Every server side program must receive client request.
4. Every server side program must read request data (request parameter).
5. Process the data or request.
6. Send the response back to the client.

Socket: In order to develop the program at client side we must use the class Socket.

Socket API:

Constructor:

```
Socket (String hname, int Portno) throws UnknownHostException, IOException
```

Instance methods:

```
public OutputStream getOutputStream (); →1
public InputStream getInputStream (); →2
public void close (); →3
```

Method-1 is used for establishing the connection with server socket by passing host name and port number of the server side program. Method-2 is used for writing the data to the server side

program. Method-3 is used for receiving or reading the response given by the server side program. Method-4 is used for closing socket or client communication with the server.
Socket s=new Socket (“localhost”, 7001);

<u>ServerSocket</u>: In order to develop the program at server side we must use the class ServerSocket.

<u>ServerSocket API</u>:
<u>Constructor</u>:
```
ServerSocket (int portno) throws IOException →1
```

<u>Instance methods</u>:
```
public Socket accept (); →2
public void close (); →3
```
Method-1 is used for making the server side program to run at certain port number. Method-2 is used for accepting socket data (client data). Method-3 is used for closing or terminating server side program i.e., ServerSocket program.

Write a java program which illustrates the concept of Socket and ServerSocket classes?
<u>Answer</u>:
<u>server.java</u>:
```
import java.net.*;
import java.io.*;
class server
{
	public static void main (String [] args)
	{
		try
		{
			int pno=Integer.parseInt (args [0]);
			ServerSocket ss=new ServerSocket (pno);
			System.out.println ("SERVER IS READY");
			while (true)
			{
				Socket s=ss.accept ();
				InputStream is=s.getInputStream ();
				DataInputStream dis=new DataInputStream (is);
				int n=dis.readInt ();
				System.out.println ("VALUE OF CLIENT = "+n);
				int res=n*n;
				OutputStream os=s.getOutputStream ();
				DataOutputStream dos=new DataOutputStream (os);
				dos.writeInt (res);
			}
		}
		catch (Exception e)
		{
			System.out.println (e);
		}
	}
};
```

client.java:
```
import java.io.*;
import java.net.*;
class client
{
	public static void main (String [] args)
	{
		try
		{
			String sname=args [0];
			int pno=Integer.parseInt (args [1]);
			Socket s=new Socket (sname, pno);
			System.out.println ("CLIENT CONNECTED TO SERVER");
			OutputStream os=s.getOutputStream ();
			DataOutputStream dos=new DataOutputStream (os);
			int n=Integer.parseInt (args [2]);
			dos.writeInt (n);
			InputStream is=s.getInputStream ();
			DataInputStream dis=new DataInputStream (is);
			int res=dis.readInt ();
			System.out.println ("RESULT FROM SERVER = "+res);
		}
		catch (Exception e)
		{
			System.out.println (e);
		}
	}
};
```

Day - 73:

Disadvantages of networking:

1. We are able to develop only one-one communication or half-duplex or walky talky applications only.
2. We are able to get only language dependency (client side and server side we must write only java programs).
3. There is no support of predefined protocol called http.
4. There is no internal support of third party servers such as tomcat, weblogic, etc.
5. We are able to develop only intranet applications but not internet applications.

DESIGN PATTERNS:

Design patterns are set of predefined proved rules by industry experts to avoid recurring problems which are occurring in software development. As a part of information technology we have some hundreds of design patterns but as a part of J2SE so far we have two types of design patterns. They are **factory method** and **Singleton class**.

- A method is said to be a *factory method* if and only if whose return type must be similar to name of the class where it presents.

  Rules for factory method:

  1. Every factory method must be public method.

2. Every factory method must be similar to name of the class where it presents.

- A *Singleton class* is one which allows us to create only one object for JVM. Every Singleton class must contain one object on its own and it should be declared as private. Every Singleton class contains at least one factory method. Default constructor of Singleton class must be made it as private.

Write a java program which illustrates the concept of factory method and Singleton process?

Answer:

```
class Stest
{
	private static Stest st;
	private Stest ()
	{
		System.out.println ("OBJECT CREATED FIRST TIME");
	}
	public static Stest create ()
	{
		if (st==null)
		{
			st=new Stest ();
		}
		else
		{
			System.out.println ("OBJECT ALREADY CREATED");
		}
		return (st);
	}
};
class DpDemo
{
	public static void main (String [] args)
	{
		Stest st1=Stest.create ();
		Stest st2=Stest.create ();
		Stest st3=Stest.create ();
		if ((st1==st2)&&(st2==st3)&&(st3==st1))
		{
			System.out.println ("ALL OBJECTS ARE SAME");
		}
		else
		{
			System.out.println ("ALL OBJECTS ARE NOT SAME");
		}
	}
};
```

Day - 74:

STRING HANDLING: A string is the sequence of characters enclosed within double quotes.

For example:

```
"Java is a programming language"
```

In order to deal with strings we have two classes. They are **java.lang.String** and **java.lang.StringBuffer**

What is the difference between String and StringBuffer classes?
Answer: String class is by default **not mutable** (non-modifiable) and StringBuffer class is **mutable** (modifiable).

String API:

Constructors:

```
String (); →1
String (String); →2
String (char []); →3
```

Constructor-1 is used for creating empty string. Constructor-2 is used for creating a String object by taking another string parameter. Constructor-3 is used for converting sequence of characters into string.

Instance methods:

```
public char charAt (int); →1
public int length (); →2
public boolean equals (String); →3
public boolean equalsIgnoreCase (String); →4
public String concat (String); →5
public boolean startsWith (String); →6
public boolean endsWith (String); →7
```

Method-1 is used for obtaining a character from a string by specifying valid character position. Method-2 is used for obtaining number of characters in the string. Method-3 is used for comparing two strings. If two strings are equal in its case and meaning then this method returns true otherwise false. Method-4 is used for comparing two strings by considering meaning by ignoring case. Method-5 is used for concatenating two strings and the result is stored in another string. Method-6 returns true provided the target String object present in first position of source String object otherwise false. Method-7 returns true provided the String object present in last position of source String object otherwise false.

Static methods:

```
public static String valueOf (byte);
public static String valueOf (short);
public static String valueOf (int);
public static String valueOf (long);
public static String valueOf (float);
public static String valueOf (double);
public static String valueOf (char);
public static String valueOf (boolean);
```

These methods are used for converting any fundamental value into String object and this method is overloaded method.

```
public String substring (int start);→1
public String substring (int start, int end); →2
```

Method-1 is used for obtaining the characters from specified position to end character position. Method-2 is used for obtaining those characters by specifying starting position to ending position.

What is the similarity between String and StringBuffer classes?

Answer: Both String and StringBuffer classes are public final. Hence, they are not extended by any of the derived classes and we cannot override the methods of String and StringBuffer class.

Day - 75:

StringBuffer class:

Whenever we create an object of StringBuffer we get 16 additional characters memory space. Hence an object of StringBuffer is mutable object.

StringBuffer API:

1. `StringBuffer ()`
2. `StringBuffer (String)`
3. `StringBuffer (char [])`
4. `StringBuffer (int size)`

Constructor-1 is used for creating an object of StringBuffer whose default capacity is 16 additional characters memory space.

For example:

```
StringBuffer sb=new StringBuffer ();
System.out.println (sb.capacity ());
```

Constructor-2 is used for converting String object into StringBuffer object.

For example:

```
String s="HELLO";
StringBuffer sb=new StringBuffer (s);
System.out.println (sb.capacity ());
System.out.println (sb.length ());
```

Constructor-3 is used for converting array of characters into StringBuffer object.

For example:

```
char ch [] = {'J', 'A', 'V', 'A'};
StringBuffer sb=new StringBuffer (ch);
System.out.println (sb);
```

Constructor-4 is used for creating an StringBuffer object with from specific size.

For example:

```
StringBuffer sb=new StringBuffer (256);
Sb="JAVA IS AN APPLICATION";
```

Instance methods:

1. `public int length ();`
   This method is used for determining the length of the string.

2. `public int capacity ();`
   This method is used for determining the capacity of StringBuffer object. Capacity of StringBuffer object is equal to the number of characters in the StringBuffer plus 16 additional characters memory space.

For example:
```
StringBuffer sb=new StringBuffer ("HELLO");
System.out.println (sb.length ());
int cap=sb.capacity ();
System.out.println (cap);
```

3. `public StringBuffer reverse ();`
   This method is used for obtaining reverse of source StringBuffer object.
   For example:
   ```
   StringBuffer sb=new StringBuffer ("HELLO");
   StringBuffer sb1=sb.reverse ();
   System.out.println (sb1);
   ```

4. `public void append (byte);`
5. `public void append (short);`
6. `public void append (int);`
7. `public void append (long);`
8. `public void append (float);`
9. `public void append (double);`
10. `public void append (char);`
11. `public void append (boolean);`
12. `public void append (String);`
    All the above methods mean (4 to 12) are used for appending the numerical data or the string data at the end of source StringBuffer object.

13. `public StringBuffer deleteCharAt (int);`
    This method is used for removing the character at the specified position and obtaining the result as StringBuffer object.
    For example:
    ```
    StringBuffer sb=new StringBuffer ("JAVA PROGRAM");
    StringBuffer sb1=sb.deleteCharAt (8);
    System.out.println (sb1);
    ```

14. `public StringBuffer delete (int start, int end);`
    This method is used for removing the specified number of characters from one position to another position.
    For example:
    ```
    StringBuffer sb=new StringBuffer ("JAVA PROGRAM");
    StringBuffer sb1=sb.delete (5,8);
    System.out.println (sb1);
    ```

15. `public StringBuffer replace (String, int, int);`
    This method is used for replacing the string into StringBuffer object form one specified position to another specified position.
    For example:
    ```
    StringBuffer sb=new StringBuffer ("JAVA PROGRAM");
    System.out.println (sb1);
    ```

16. `public StringBuffer insert (String, int, int);`
    This method is used for inserting the string data from one specified position to another specified position.
    For example:
    ```
    StringBuffer sb=new StringBuffer ("JAVA PROGRAM");
    StringBuffer sb1=sb.insert ("J2SE/, 0 ,3");
    System.out.println (sb1);
    ```

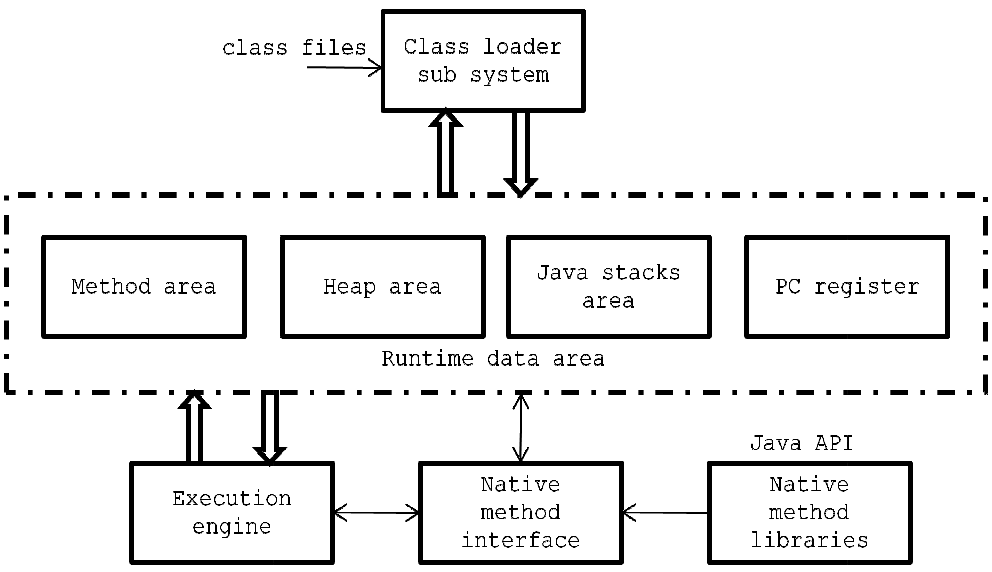
Day - 76:

In JVM we have three layers, they are:

1. **Class loader sub system:**
   It is a part of JVM which loads the class files into main memory of the computer. While generating class files by the JVM it checks for existence of java API which we are using as a part of java program. It the java API is not found it generates compilation error.

2. **Runtime data area:**
   When the class files are loaded into main memory JVM requires the following runtime data area, they are
   Heap memory: It is used for allocating the memory space for the data members of the class at runtime.
   Java stacks/method area: In order to execute any method in java, JVM allocated some amount of memory space in the main memory on stack memory known as method area.
   PC register: PC stands for Program Counter which is known as general purpose register. The PC register always gives address of next instruction of java program to the JVM.

JVM Architecture



3. **Execute engine:**
   When a JVM is compiling in a java program in that context JVM is known as compiler. When a JVM is running a java program in that context JVM is known as interpreter.
   JVM reads the byte code of .class line by line. Byte codes are set of optimized instructions which are used by JVM for generating the result of the java programs.
   Method interface in the JVM architecture represents the middle man role between JVM (execution engine) and java API. Java API contains collection of packages.

TOOLS used in JDK:

Tools are nothing but the exe file which are developed by SUN micro system to run various applications of java, J2EE.

1. **appletviewer:**
   This is a tool used for running applet application when the browser is not supporting properly.

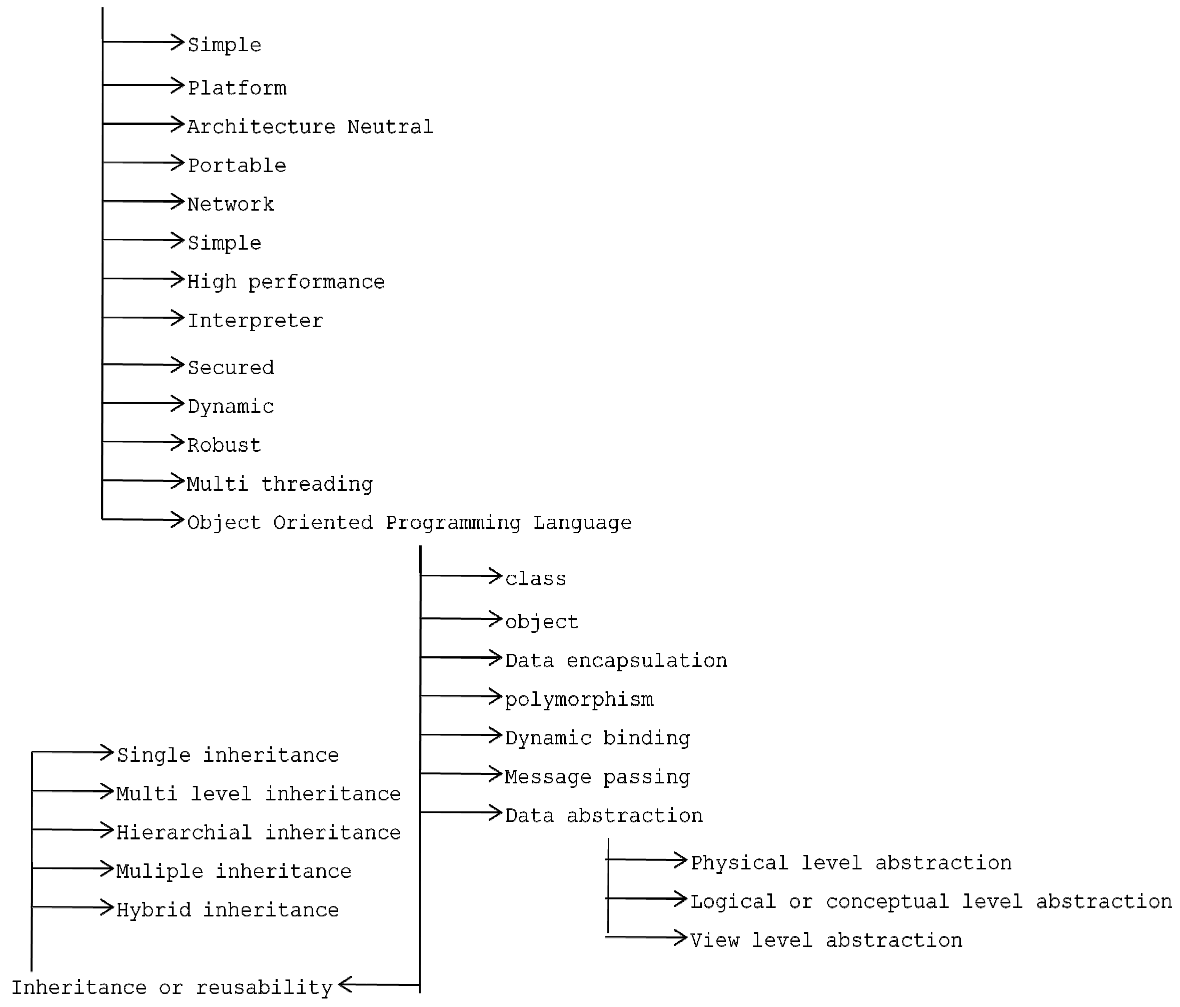
2. **<u>jar</u>:**
   jar stands for java archive. We use jar file for developing business component or application class (In J2EE applications we generate jar file for EJB applications).
   war stands for web archive used for developing web components (Servlets and JSP are called web components).
   <u>Syntax for create **war/jar** file</u>:
   ```
   jar cfv filename.war/jar *.class
   ```
   <u>For example</u>:
   ```
   jar cfv sathya.jar *.class
   ```

3. **<u>java</u>:** It is used to run a java program.

4. **<u>javac</u>:** It is used to compile a java program.

5. **<u>javap</u>:** It is used to see the API of the specific class or interface which belongs to a specified package.

<u>FEATURES OF JAVA</u>



Data types

- Integer category data types
  - byte (1-byte)
  - short (2-bytes)
  - int (4-bytes)
  - long (8-bytes)
- Float category data types
  - float (4-bytes)
  - double (8-bytes)
- Character category data types
- Boolean category data types

Constants

- final
  - At variable level
    - Final variable initialize
    - Final variable declaration
  - At method level
  - At class level

Keywords

- instance
  - At variable level
  - At method level
- static
  - At variable level
  - At method level
- this
  - this ()
  - this (..)
- super
  - At variable level
  - At method level
  - At consrtuctor level
    - super ()
    - super (..)

String data into fundamental data

- → Wrapper classes
  - → Byte → public static Byte parseByte (String);
  - → Short → public static Short parseShort (String);
  - → Integer → public static Int parseInt (String);
  - → Long → public static Long parseLong (String);
  - → Float → public static Float parseFloat (String);
  - → Double → public static Double parseDouble (String);
  - → Character → public static Char parseChar (String);
  - → Boolean → public static Boolean parseBoolean (String);

PrintStream class

- → println methods
  - → public void println (byte);
  - → public void println (short);
  - → public void println (int);
  - → public void println (long);
  - → public void println (float);
  - → public void println (double);
  - → public void println (char);
  - → public void println (boolean);
  - → public void println (String);
- → print methods
  - → public void print (byte);
  - → public void print (short);
  - → public void print (int);
  - → public void print (long);
  - → public void print (float);
  - → public void print (double);
  - → public void print (char);
  - → public void print (boolean);
  - → public void print (String);

Object to fundamental data

- → public byte byteValue ();
- → public short shortValue ();
- → public int intValue ();
- → public long longValue ();
- → public float floatValue ();
- → public double doubleValue ();
- → public char charValue ();
- → public boolean booleanValue ();
- → Public String stringValue ();

Object to String type

- public String toString ();

String object into wrapper class object

- Wrapper classes
  - Byte → public static Byte valueOfByte (String);
  - Short → public static Short valueOfShort (String);
  - Integer → public static Int valueOfInt (String);
  - Long → public static Long valueOfLong (String);
  - Float → public static Float valueOfFloat (String);
  - Double → public static Double valueOfDouble (String);
  - Character → public static Char valueOfChar (String);
  - Boolean → public static Boolean valueOfBoolean (String);

Constructors

- Default constructor
- Overloaded constructors

$$\left.\begin{array}{l}\rightarrow \text{Number of parameters different}\\ \rightarrow \text{Types of parameter different}\\ \rightarrow \text{Order of parameter different}\end{array}\right\}\text{Signature}$$

Access specifiers

- private
- default
- protected
- public