BSIC INFORMATION REQUIRED FOR ADVANCED JAVA

Java Software Editions or Modules or Flavors:

Java software classified into three modules

A. JSE Module / J2SE Module.
B. JEE Module / J2EE Module.
C. JME Module / J2ME Module

JSE Module:

Java Standard Edition

Or

Java 2 Platform Standard Edition

It is basic module for JEE and JME.

The reason is if we want develop JEE and JME module we required JSE module.

It is installable software.

To work with JSE first we need to install one software that is JDK.

JDK: Java Standard Edition Development Kit.

By using this software we can only compile and execute the JSE related programs and operations but we can execute JEE and JME module applications.

The Last Version of JDK Version is JDK 1.9 / JSE 9.

Mainly software versions are classified into two types.

A. Major Version.
B. Minor Version.

Major Version: If we want to add new features to existing software then new software coming into market in the form of Major Version.

Minor Version: Once we add new features into existing software, there may be a chance of bugs.

To resolve the bugs, we need to write bug resolve code for Major Version, after adding bug resolve code we need to introduce new software into the market in the form Minor Version.

With the help of JSE, we can develop the following applications.

A. Standalone Applications.
B. Gamming Applications.
C. Networking Applications.
D. Applet programs.

Standalone Applications:

Here Standalone Applications means, only one end user can be interact with the resources at a time.

Types of Standalone Applications:

A. Character User Interface Applications / Console Interaction Applications.
B. Graphical User Interface Applications / Desktop Applications.

CUI Applications:

These types of applications are interacts with end user in the form of character by character.

Ex: If we want to read Employee related information, we need to provide some user friendly message like ENTER EMPLOYEE ID and EMPLYEE NAME etc...

By using these messages only end user can interact with applications.

These are not user friendly applications.

These applications not provide any look and feel to end users.

Some times may be face difficulty to interact with applications.

To avoiding these drawbacks java introduced one more type of application those are GUI applications.

GUI Applications:

These are very user friendly applications.

Provide more Look and Feels.

These applications are provides more facilities to interact with in easy manner.

Every person who doesn't have any idea on applications also can easily interact with applications.

These type of applications can develop by using some of JSE libraries those are AWT, SWING and APPLET.

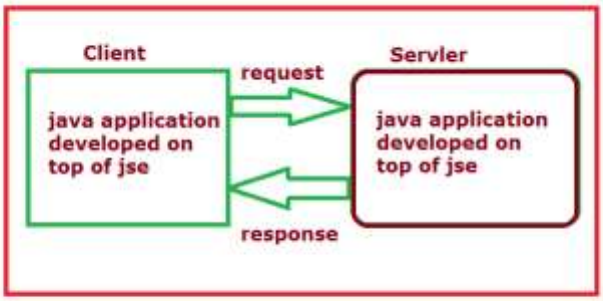
Networking applications or Socket Programming:

These applications are one type client and server applications.

Most of the peoples feel like client and server applications means client and server applications are running under different machine. That is totally wrong.

We can develop client and server applications within the single applications also.

We can develop networking applications or socket application by using JSE library that is java.net package.



Gaming Applications:

We can develop these applications by using AWT and SWING library; these are interacts with only one end user.

Q) Why should we learn JAVA Software?

There are lots of reasons we have to learn java software.

A. Platform Independent:

Developing one application under one operating system and executing under different applications.

Java provides response to all the end users within the same time and quickly.
The reason is it supports multiple Operating Systems.
In windows, with the support of multiple processors (Around 8 processors)
In Linux, with the support of multiple processors. (Around 64 microprocessors).

B. Open Source:

a. **Java software is open source.**
b. **Java software documentation is open source**
c. **Java software supported servers are open source.**
d. **Java software supported frameworks are open source.**
e. **Java software supported IDE's are open source.**

C. Supports All Scale Applications.

Java software will supports small and middle and large scale applications.

Ex: Banking applications, ATM applications, Amazon, Flipkart applications etc.

D. Security:

Java Byte code will provide security. Once we tampering the byte code that will be not executed.

E. Job Security:

The meaning of job security is java having long life in the market.

Now a day's java is the basic technology to develop and learn other technologies like Android, Hadoop, spring, Servlet, JSP, JDBC, Sales Force, Selenium and etc….

Java Standard Edition comes into the picture in 3 Parts.

A. Language:

It will provide basic infrastructure or information to develop applications. Technically we can calls as raw materials.

Languages are provides basic syntax and procedures.

Syntax means don't use keyword as class name, every statement is ended with semicolon etc…

Procedure means the way writing the program from top to bottom

Like in every java program

1. **Starts with documentation.**
2. **Package statements.**
3. **Import statements.**
4. **Interface.**
5. **Abstract class.**
6. **Concrete class.**
7. **Final class.**
8. **Enum.**
9. **Annotations.**
10. **Main method class.**

Programming language are supported to develop software technologies (JDBC, Servlet, JSP, frameworks (spring, Hibernate, Struts), tools (Ant, Maven, CVN, GIT).

Ex: C, C++, Java, VC++, C# etc

Java provides basic information in the form of oops, exception handling, applet, iostreams, multithreading, reflection-api.

B. Technology:

Java is a technology, it will provide specifications. Specification means set of rules and guide lines.

Rules we can develop with the support of interfaces and guide lines we can develop with the support of classes.

Java provides different types of technologies like JDBC, Servlet, JSP in the form of specifications.

By using these specifications we can develop software's.

For example JDBC is technology, which is used to develop JDBC drivers, these are software's.

Servlet is a technology, which is used to develop software's like Servers.
Ex: Apache Tomcat, Glass Fish, Web Logic etc..

Technologies are not installable software's. But technology based software's installable like JDBC drivers and servers and so on.

Every technology coming into the market in the form api and packages.

Package contains set of interface and class.

For Example JDBC interfaces are provides rules and classes provides guide lines.

When ever, any third party vendor wants to develop technology based software's, those vendors must be followed by the technology rules and guidelines.

Working with the technology means working with technology based software's.

Working with JDBC technology means working with JDBC technology based software's.

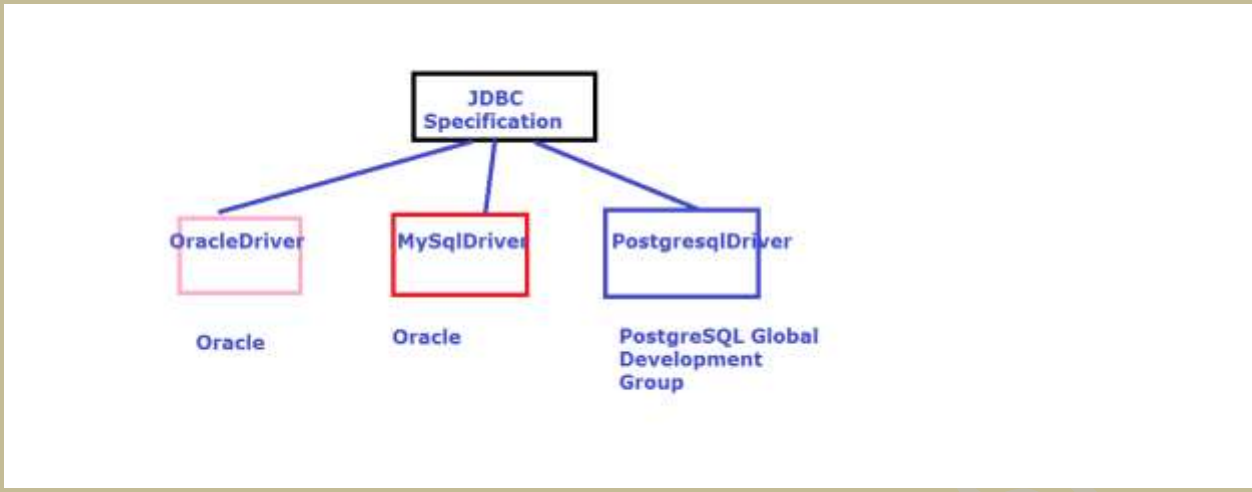
Technologies are mainly classified into two types.

1. **Open technologies.**
2. **Proprietary technologies.**

Open technologies are provides specifications to all third party vendors to develop their own technology based software's.

JDBC is one Open technology it provides specifications to all third party vendors.

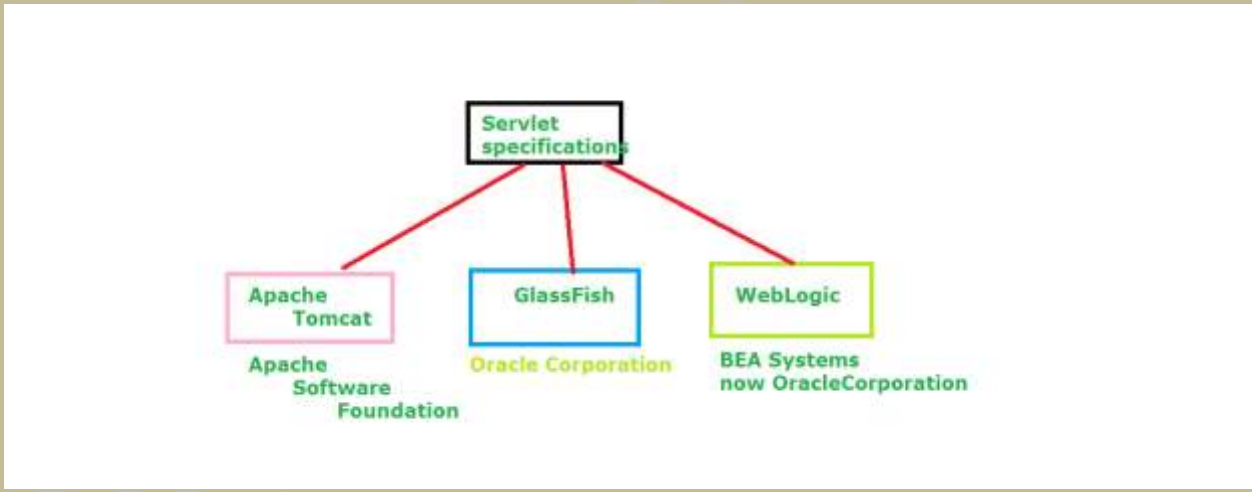
Different third party vendors develop different software's means different drivers like oracle driver, mysql driver, postgresql and so on.



In the same manner Servlet is one open specification to all third party vendors.

By using these specifications different third party vendors develop different server software's.

Like Apache Tomcat, Web Logic, Glass Fish etc...



If the rules and guide lines are open source then competition will be increases, then we will get best software product.

Cost of software product will be cheap.

Once technology is open source day by day new features will be added for providing best software.

Proprietary technologies are provides rule and guidelines for only one vendor.

That vendor itself provides technology based software's.

Like asp.net related to Micro Soft.

If vendor is same there is no competition.

The cost of the software will be more.

May be some we will get best product and sometimes not.

C. Platform:

Java has its own platform to execute the programs in the form JVM.

JEE Module:

The full form is Java Enterprise Edition.
New Version is JEE7.

JEE module is not installable software.
It is specification.
Provides rule and guide lines to develop server software's like Apache Tomcat, GlassFish, Web Logic etc.

JEE module specification coming into the market in the form of .jar files.

For example JDBC specification coming into the market in the form of jar files like ojdbc6.jar or ojdbc14.jar files.

Servlet specification coming into the market in the form of jar files like Servlet-api jar file.

By using JEE Module we can develop following applications
Like

A. **Web Applications.**
B. **Distributed Applications.**
C. **Enterprise Applications.**

Both Web and Distributed Applications are part of the Enterprise Applications only.

Web Applications:

It is one web program which will get the request from the end user and provides appropriate response to end user (client).

We can simply say this is client-server application.

To make communication between clients to server we use one protocol that is "http".

The collection of web application we can calls as web site.

For example web site is "Naresh I Technoloiges" and web application is "Course".

In generally we can say web site is one restaurant and individual menu items are web applications.

In this applications client is fixed client. (Browsers like Google Chrome, Opera Mini, Safari, Mozilla Firefox and Internet Explorer).

To make a communication between client to server either we can use LAN or MAN or WAN.

This architecture we can also called as "Thin Client and Fact Server".

Thin Client:

No need of any extra software architecture. Ex: Browsers.

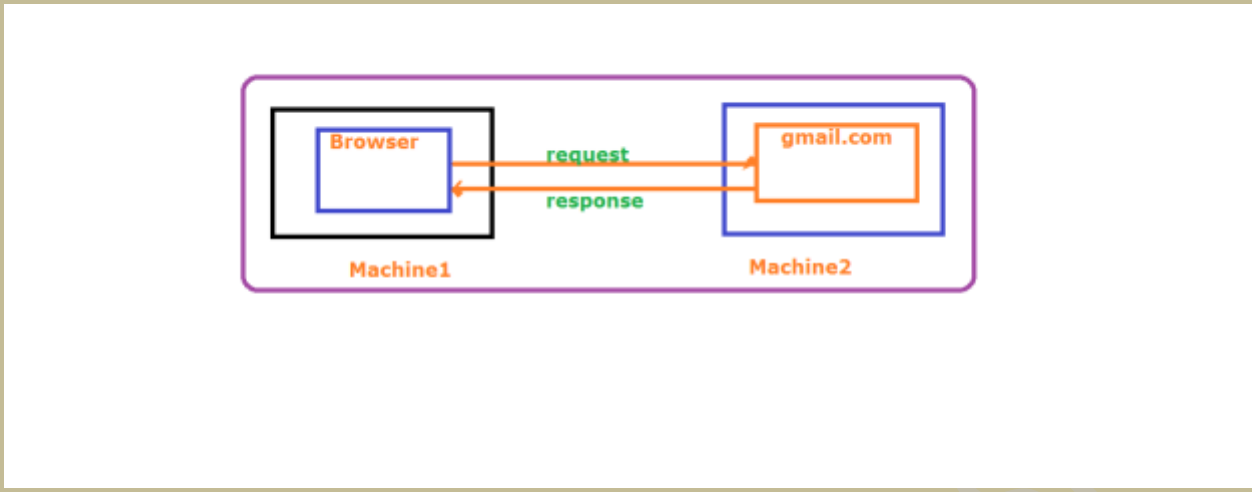
Thick Client/ Fat Client:

Required some extra software configurations.

Thick Server/ Fat Server:

We required some extra software's configurations.

Like gmail.com and facebook.com etc.



Distributed Applications:

The application resource common for all end user commonly throughout networks.

It is also one type client- server application.

To make communication between client and server we can use different type protocols.

Ex:

- **FTP (File Transfer Protocol).**
- **SMTP (Simple Mail Transfer Protocol).**
- **TCP (Transmission Control Protocol).**
- **Telnet (Teletype Network).**
- **HTTP (Hyper Text Transfer Protocol).**
- **HTTPs (Secure Hyper Text Transfer Protocol).**
- **MTP (Media Transfer Protocol).**

Initially client is not directly communicates with server applications.

First client must be gets the details about server application from repository.

If any application wants to provides services to client, that application details must be placed into repository.

Once getting the details from the repository, then client interact with server for appropriate services.

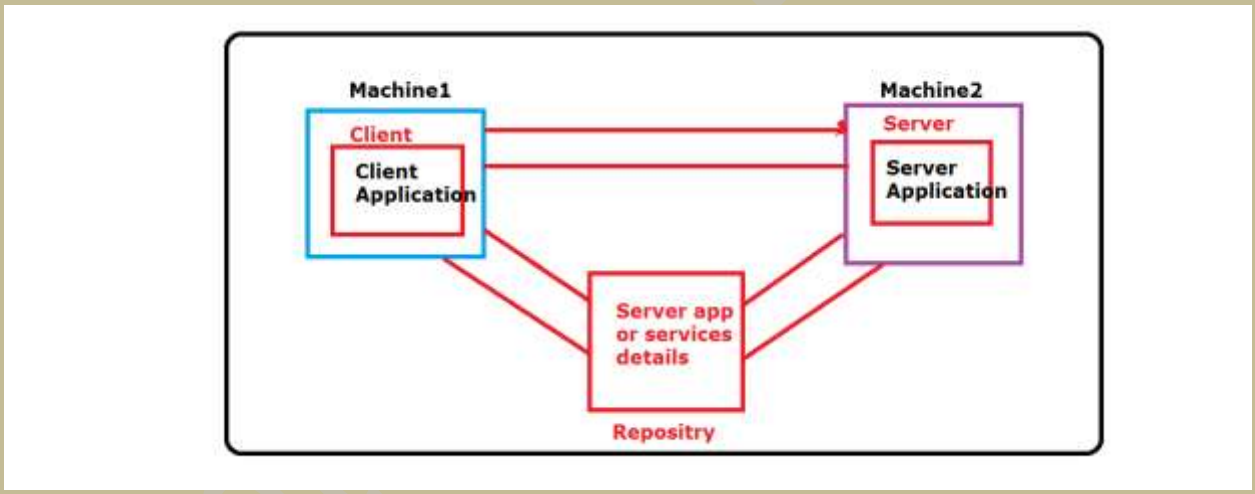
Ex: ATM Machine, Banking Applications, Health Care, Insurance etc.

These applications we can also called Fat Client and Fat Server Architecture.

Ex: If we want to withdraw the money from ATM machine.

First ATM Machine will communicate with appropriate server and provide the response to end user.

ATM machine is one client here. ATM machine definitely we required extra software equipment.



There are lot of difference between Web Applications and Distributed Applications.

In Web Application Architecture:

Client is thin client

Client can able communicate with the server directly.

No need of any extra repository for communicates with server.

In Distributed Application Architecture:

Client is Thick Client

We required repository services.

Once get service details from repository and communicate with server.

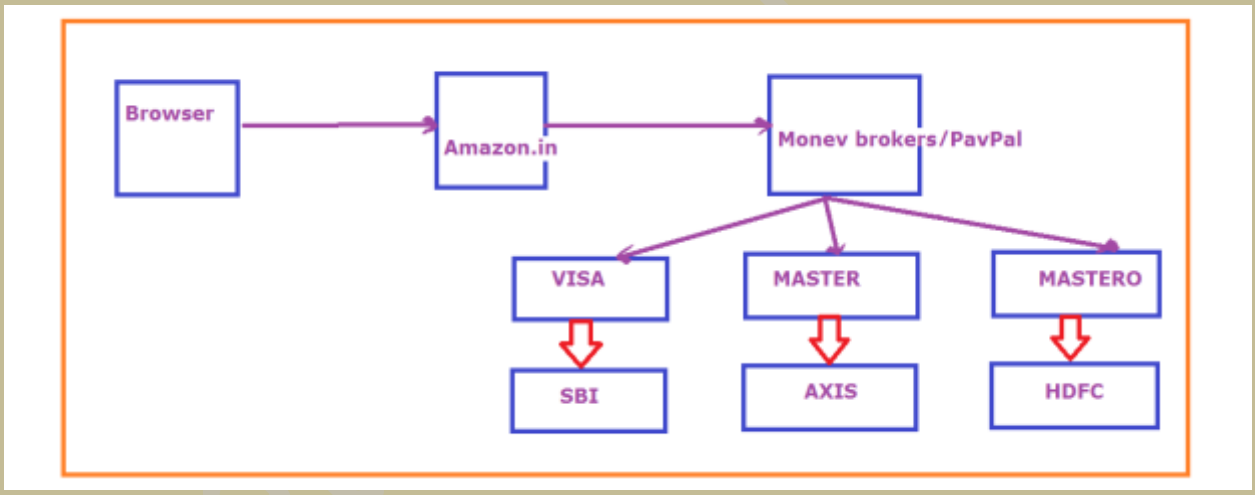
Server is always thick.

Enterprise Applications:

It is always deals with very large or complex, middle scale projects.

It is the combination of web and distributed applications.

Diagram:



PayPal Holdings is an American company operating a worldwide online payments system.

It has been supports online money transfers and serves as an electronic.

It is alternative of traditional papers and methods like checks and money orders.

It is the world largest internet payment companies.

It will provide more security.

It is communicate with different types of card and related organization (Banks).

JME Module:

JME full form is Java Micro Edition.

If we want to develop JME related applications we need install MDK software (Mobile Development Kit).

It is used to develop mobile related and electronic chip level applications.

These applications related programs develop on top of regular machine architecture and finally dumped into chip.

These applications performance is very higher but application wise smaller.

By using JME module we can develop fully function oriented application like refrigerator, washing machines, micro woven and also develop mobile oriented applications like mobile phonebook, phone call mechanism, music and video player, mobile plugin, mobile games etc.

Now days no body using these applications. Peoples are showing interest to develop above mobile related applications through Android and IPhone.

Real Time Terminologies:

Software Vendor/Developing Organization:

The Organization, which is develops the software and released into market either open or proprietary.

Ex: Data Base (Oracle), Operating System (Windows), Programming Language (Java, .NET), Frameworks (spring, hibernate).

Software Services Organization:

These Organization are used software's which are given by the vendors and develop software project or products.

Ex: Value Labs, TCS, Wipro etc.

Software Service Utilization Organization or Client Organization:

These are used the project or products for business services developed by software service organization.

Ex: Citi Bank, Amax etc..

The main intension of client organization is providing services to end user.

Types of Projects:

Project is a collection of applications, it will provide services to end user with specific client.

Scratch Level Projects:

Each and every code of projects we need to develop.

Maintenance Projects:

After developing the projects, there may be chance of bugs. To solve these bugs we need to write bug resolve code to existing projects.

Based on client business requirements we need to add some new features and enhancement to existing project.

The above type of projects comes under Maintenance projects.

Migration Projects:

Based on new technologies and new features of software, sometimes we need to migrate our application from one version to another and one technology to other technologies.

Application Programming Interface:

To develop any application under any technology, we required some theoretical and programming knowledge.

These types of knowledge we can achieve through documentation and API.

API is communication channel between end user and program.

Here interface terminology is nothing but it is one English terminology. This word is no more related to java concept interface topic.

If we want to develop application on any programming language we required some predefine coding, those code are coming in the form of API's.

For example if we want to develop application on C language we required some predefine code, these code are coming in the form of header files.

Ex: <stdio.h> and <conio.h>

If we want to develop application on java language we required some predefine code coming in the form of classes, interface, enum and annotations.

All these coding concepts are coming in the form of API.

Ex:

If we want to develop general and common programs in java we required java.lang API.

If we want to develop graphic oriented applications we required java.awt or javax.swing or java.applet API.

If we want to develop database interaction application we required java.sql and javax.sql API.

If we want to develop web and distributed application we required javax.servlet API.

Types of API:

There are three types of applications.

Pre-Define API:

The API which coming along with technology, software and frameworks we can called as pre-define API.

User-Define API:

The API which is develops by the programmer or developer called as user-define API.

Third Party API:

The API, which is given by the third party vendors like

Ex: Apache tomcat, Glass Fish, Web Logic, Oracle corporations..

Environment Variables Setting:

Command prompt is comes with operating system, whereas java development tools (binary files) and library files are comes with java software.

So our command prompt is unable to recognize the development tools.

Then programmer should provide the information about binary and library files to command prompt, with the help of environment variables.

OS by default uses environment variables to recognize software.

For java we have two types of path settings.

1) Temporary Settings.

2) Permanent Settings.

Temporary Settings:

Whatever the setting which we are done at one command prompt these settings are applicable to that command prompt only not applicable to other command prompt and also if we did close the command prompt automatically all the settings are gone.

Syntax: set path=C:\Program Files\Java\jdk\bin;

With the help of above settings our command prompt recognizes the binary files. (javac, java, javap command).

Syntax": set classpath=.;C:\Users\lenovo\Desktop\ojdbc6.jar;

With the help of above settings our compiler and JVM can recognize the predefine .class files.

Binary files used to compile and execute the program, where as Library files are used develop the user define program.

Permanent Settings:

The settings which are applicable for entire system, these settings are called permanent settings.

Steps to permanent settings:

Right click on my computer →Properties →Advanced System Settings

→Advance →Environment Variables →under System variables.

Click on new button set the path.

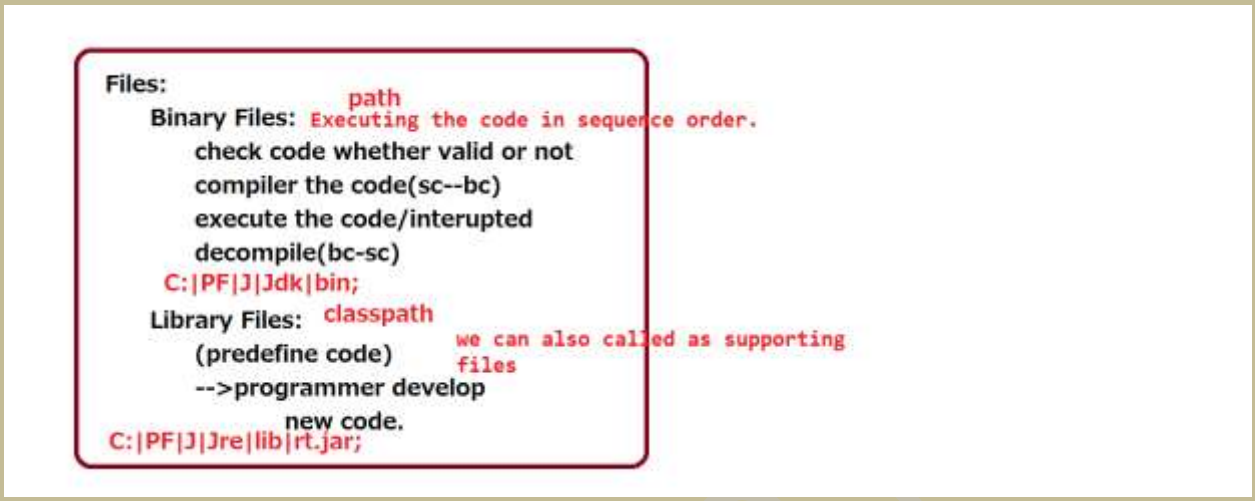
Variable Name: path

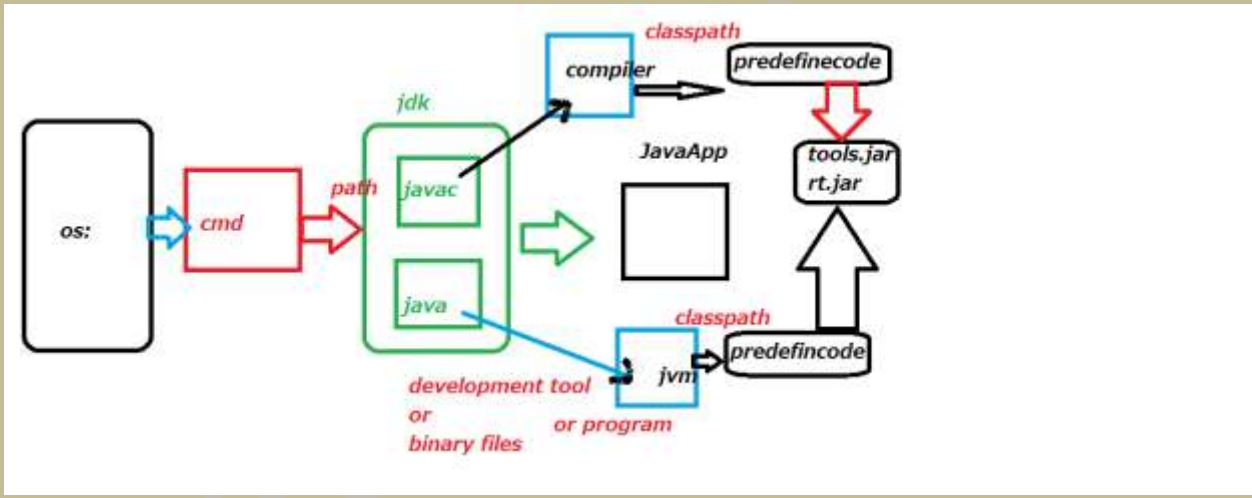
Variable Value: C:\Program Files\Java\jdk1.6.0\bin;

Click on new button set the classpath.

Variable Name: classpath

Variable Value: .;C:\Users\lenovo\Desktop\ojdbc6.jar;





Java archive: (jar):

It is one compressed file in java standard edition.

It contains java standard edition classes.

It can reduce file size.

With the help jar we can execute the programs also.

Web archive: (war):

It is also one compressed file in advanced java.

It contains the following files.

- **.servlet related classes**
- **.jsp file**
- **.html files**
- **.jpeg files**
- **.xml files**
- **.properties files**

Enterprise archive: (ear):

It is also one compressed file in enterprise edition.

This file contains the following files.

- **.servlet related classes**
- **.jsp**
- **.html**
- **.jpeg**
- **.xml files**
- **.properties files**
- **Enterprise java beans classes.**

Creating .jar file and add .class file:

```
public class A{
public static void main(String... args){
        System.out.println("jar file");
}
}
```

javac A.java

We are getting A.class file

jar -cf one.jar A.class

Delete the A.class file

Set the one.jar file in the class path then execute the program

java A

output: jar file.

JVM ARCHITECTURE:

Virtual Machine:

VM is a software/Application which will provide environment or memory areas for executing the programs.

JVM: JVM is a software/application which will provide environment or five individual identical memory areas for executing the java programs.

We have two types of JVM

1) Client JVM

2) Server JVM.

JVM is only provides abstraction.

JRE is provides implementation for that abstraction.

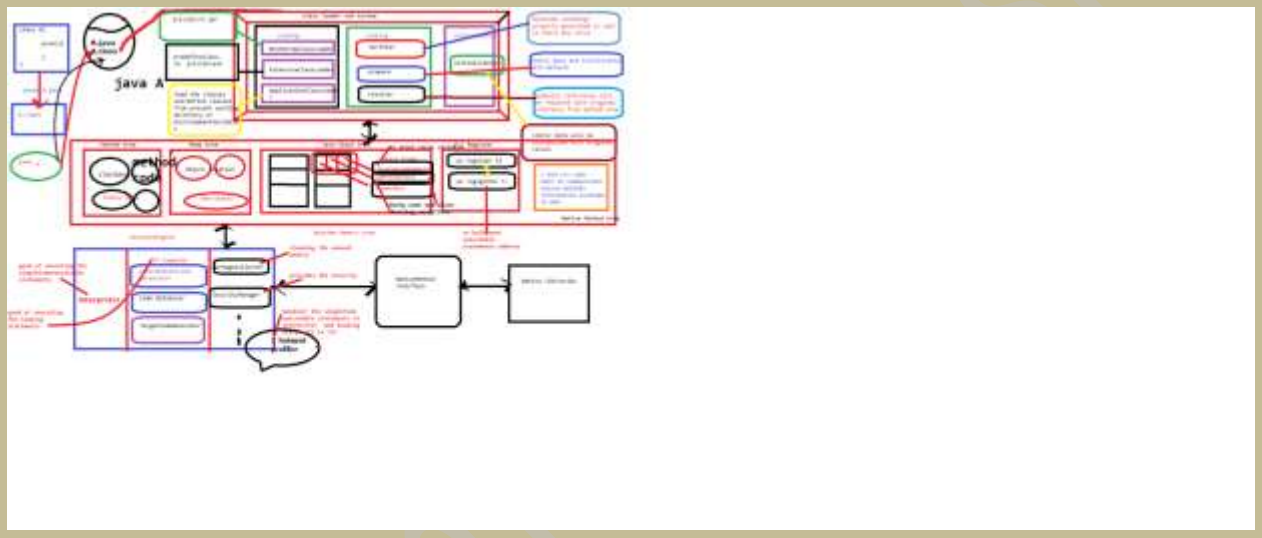
We have different JVM'S for different OS.

That means we have different JRE software for different Operating Systems.

If we want to execute the java program, first we should interact with the JVM.

With the help of "java" command we communicate with JVM. "java" keyword is always followed by class name.

syntax: java Class_Name



Java is internally communicate the with JVM, then JVM will come into the picture, first JVM will read class name and JVM control will go to secondary memory, that class byte code will be loaded from secondary memory to primary memory.

JVM internally uses ClassLoaderSubSystem to load the .class files from secondary memory to primary memory.

In the ClassLoaderSubSystem, we have three separate phases. Those are

1) Loading phase.

2) Linking phase.

3) Initialization phase.

Loading Phase:

In this phase the byte code (predefine and user define class) will be loaded from secondary memory to primary with the help of bellow classloaders.

In this phase we have three ClassLoaderSubSystems.

1. ApplicationClassLoaderSubSystem
2. ExtensiveClassLoaderSubSystem
3. BootstrapClassLoaderSubSystem.

ApplicationClassLoaderSubSystem:

It will loads user define class files from current directly and environment variable path.

ExtensiveClassLoaderSubSystem:

It will loads predefine class files from jre\lib\ext folder.

BootstrapClassLoaderSubSystem:

It will load predefine class files from jre\lib\rt.jar file.

In the above three phases class is not available, then we will get one error cannot find or load main class <class_name>.

If available that .class file will loaded from secondary memory to primary memory and handover to linking phase.

Linking:

In this phase the loaded byte code will be checked by verifier.

In this phase we have three components.

1. Verifier

2. Preparer

3. Resolver.

Verifier will check whether the byte code is properly organized or not, is there any virus and hacking code or not. If yes verifier will give verifier error, if not that byte code will be handover to preparer.

Preparer will provide the default values to static variables in loading phase.

Resolver: it will convert symbolic reference into original references.

After this phase code will be handover to initializer.

Initialization: all default values of static data will be replaced with original or actual data.

Runtime Memory Areas:

JVM provides 5 runtime memory areas.

1) Method Area

2) Heap Area

3) Java Stack Area

4) PC Registers

5) Native Method Area

Method Area: In this area all class data is stored. that means all the static data is available in the method area.

Heap Area: in this area all object data is stored. that means all the non-static data is available in heap area.

All the objects are created in the heap area only.

Java Stack Area: Every thread have its own stack.

These stack will be interact with local data/method level data. Java stack have different slots, those slots are called stack frames.

Java Stack Frame:

It will convert into three parts.

1) Local Variable Storage Area:

In this all local variables are stored.

2) Operand Stack:

in this phase all operations and calculations will be happens.

3) Frame Data:

If method contains any exceptions, those exceptions will be thrown by frame data only.

PC Registers:

Every thread have its own PC Registers, these registers will hold the next execution statements address.

Native method area:

If java wants communicate with c and c++ code, that code will be available in Native method area.

If we interact with c and c++ code, we need libraries, those will be given by Native Library, that library will be handover by Native Method Interface to executable engine.

Execution Engine:

Internally JVM uses two translators to convert byte code to executable code.

1. Interpreter

2. JIT Compiler

Interpreter is good at execute the single time execution statements.

JIT compiler is good at execute the looping statements.

But these two are unable to find out behavior of statements.

In this one special component come into picture that is "profiler"

First profiler will identify the weather statements are single time or looping statements.

If single time execution statement then those statements handover to interpreter otherwise handover to JIT compiler.

Java is a high performance language, the reason java internally uses two translators to covert our byte code to executable. So java applications are executed with in the less time. If the application executed within the less time automatically the performance will increasers.

In this executable engine we have some other special components.

Those are Garbage Collector and Security manager.

Reflection API:

If we want know source code information from ".class" file (byte code) then we can go for Reflection API.

With reflection API we can read information from runtime loaded class(.class).

Reflection API provides mirror information about variables, methods, constructors, annotations, exception class information.

With the help of reflection API we can access both private and public data.

This Reflection API is coming under java.lang.reflect package.

Important classes under java.lang.reflect packages are

1. **Filed**
2. **Constructor**
3. **Method**

4)Modifer

Filed: It is used to store the variable information. Variables may public or private.

Method: It is used to store the methods information (public or private)

Constructor: It is used to store the information about constructors.

Modifier: It is used to store information about access modifier and modifier.

This class provides the modifier information in the form of int.

If want convert from int to string then Modifier class itself provides some predefine method (toString()).

ABSTRACT	**1024**
FINAL	**16**
INTERFACE	**512**
NATIVE	**256**
PRIVATE	**2**
PROTECTED	**4**
PUBLIC	**1**
STATIC	**8**
STRICT	**2048**
SYNCHRONIZED	**32**
TRANSIENT	**128**
VOLATILE	**64**

java.lang.Class: This class having capability to hold bytecode information.

Important method under java.lang.Class:

1)getDeclaredFields()

2)getFields()

3)getDeclaredMethods()

4)getMethods()

5)getDeclaredConstructors()

6)getConstructors()

forName():

It is an one static factory method, is used to loads the byte code of any class.

forName() always needs bytecode not source code.

Syntax : Class cls = Class.forName("class_name");

This method has been throwing one compile time exception i.e Java.lang.ClassNotFoundException.

forName() is throwing exception like ClassNotFoundException if we are not passing exsisting ".class" file name(classname).

newInstance():

This is one instance factory methods, which is used to create object with the help of class reference variable.

Object obj = cls.newInstance().

newInstance() has been throwing two compile time excepitons.

1) IllegalAccessException

2) InstantiationException

Jvm will throwing Instantiation Exception, if given ".class "file (class) doesn't contains zero argument constructor.

Private variables by defaultly having one background method like setAccessible(false), so this method doesnot provide permission to access the data outside of the class.

If we want to access private data outside of the class we need to convert from setAccessible(false) to setAccessible(true), otherwise we will IllegalAccessException

```
import java.io.FileNotFoundException;

import java.io.IOException;

public class A {

	public int b=2000;

	A(){	//getDeclatedConstructor or Constructor

		System.out.println("A class construc");

	}

	A(int x){

		System.out.println("A class paramconstruc");

	}

	private static int c =3000;

	int m1()throws IOException,FileNotFoundException{

		return 10;

	}

	public void m2(int x){

		System.out.println("hi");

	}

}
```

****

```
import java.lang.reflect.Constructor;
```

```
import java.lang.reflect.Field;

import java.lang.reflect.Method;

import java.lang.reflect.Modifier;

public class B  extends A{

	public static void main(String[] args)

			throws ClassNotFoundException,
InstantiationException, IllegalAccessException {

		java.lang.Class cls = java.lang.Class.forName("A");
			java.lang.Object obj = cls.newInstance();

		Field f[]= cls.getDeclaredFields();

		for(Field f1:f){

			String modi = Modifier.toString(f1.getModifiers());

			if(modi.contains("private")){

				System.out.println(f1);

				f1.setAccessible(true);

				System.out.println(f1.get(obj));

				Object type = f1.getType();

				System.out.println(type);

			}


			System.out.println(f1);

		}

		System.out.println("*****************");

		Constructor[] c = cls.getDeclaredConstructors();

		for(Constructor c1:c){
```

```
                System.out.println(c1);
            }
            System.out.println("*****************");
            Method[] m = cls.getDeclaredMethods();
            for(Method a:m){
                System.out.println(a);
                System.out.println("**************");
                Class[] e = a.getExceptionTypes();
                for(Class e1:e){
                    System.out.println(e1);
                }
            }
        }
}
```

Block:

Blocks are used to hold the logic.

Doesn't have identity.

Automatically executing.

Method:

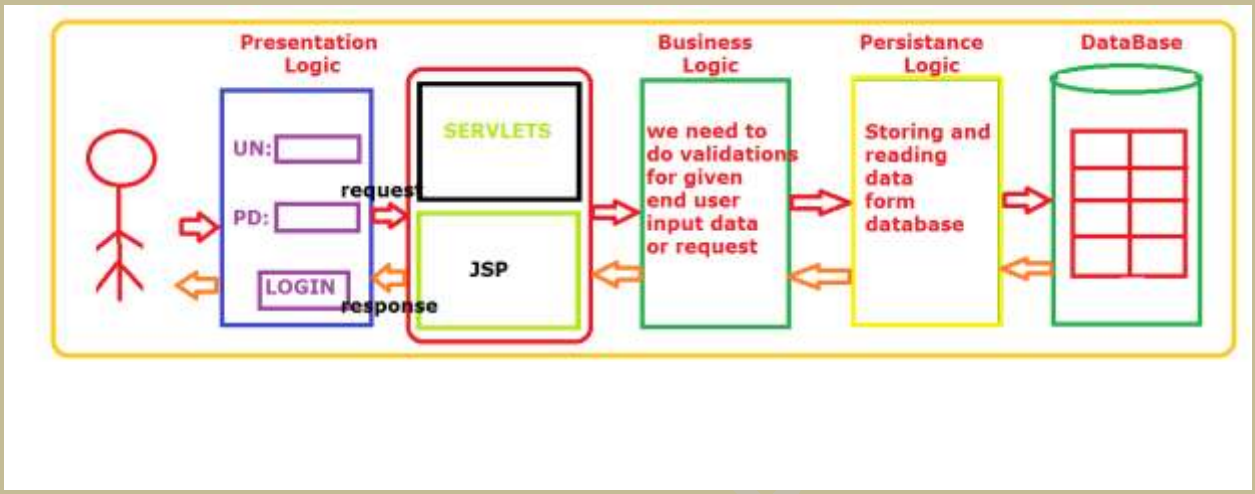
Methods are used to hold the logic.

Not executing automatically.

If we want execute, we need to call.

Methods are having identity.

Website architecture by using JDBC and SERVLETS and JSP:



To develop above architecture we required the following operations like.

a. **Reading and storing data form end user.**
b. **Doing validation and calculations using input and to generate output.**
c. **Storing output or response permanently in data base.**

To develop above operations we can use any type of language and technologies.

The above three operations calling they as

a. **Presentation Logic.**
b. **Business Logic.**
c. **Persistence logic.**

Presentation Logic:

The logic, which is used to developing GUI for reading the data from end user and providing response or output to end user, is called Persistence logic.

To develop the presentation logic we have rendering technologies like html, servlet, jsp, php etc.

Business Logic:

This is the main logic of the application. That logic contains validation and calculations for storing and generates response to end user.

We don't have any technologies for developing business logics, the reason is business type is different from one to other as well as business logic is different from business to another.

Ex: Bank , Naresh I Technologies.

Persistence Logic:

The logic which is used to store the data permanently is called persistence logic.

Ex: JDBC, JPA, PHP, ADO.NET, Servlet, JSP etc….