

## **ADVANCE JAVA**

==> It is used to create a complete application.

==> By using Advance java we can create web application.

### **APPLICATION**

The collection of programs is known as application.

==> We have 4 types of applications.

1. Stand Alone Application
2. Web Application
3. Mobile Application
4. Distributed Application

#### **1.) STAND ALONE APPLICATION**

==> The application which will work without internet is known as Stand Alone Application.

==> To develop Stand Alone Application we require Core Java.

==> Stand Alone Applications are present in Client Machine (User System).

Example: Calculator, Notepad, EditPlus, Gallery, Music, Settings etc.

---

#### **NOTE:**

Stand Alone Applications are known as Independent Applications.

---

#### **2.) WEB APPLICATION**

==> The application which will work with internet and browser is known as Web Application.

==> To develop Web Application we require Web Technology and Advance Java.

==> Web Applications are not present in Client Machine.

Example: Facebook, Myntra, Flipkart, Amazon etc.

---

### **NOTE:**

==> Browser is an application which is used to access Web Applications by using Internet.

---

## **COMPONENTS OF WEB APPLICATIONS**

==> There are two components for Web Applications

1. Front-End

2. Back-End

### **1. FRONTEND**

==> The elements which are visible to the user is known as Front-End of an application.

==> To develop Front-End of an application we require Web Technologies.

### **2. BACKEND**

==> The elements which are not visible to the user is known as Back-End of an application.

==> To develop Back-End of an application we require Java, SQL.

## **TYPES OF WEB APPLICATIONS**

==> There are two types of Web Application

1. Static Web Application

2. Dynamic Web Application

## **1. Static Web Application**

==> The application which contains only Front-End is known as Static Web Application.

==> To develop Static Web Applications we have to use Web Technologies.

## **2. Dynamic Web Application**

==> The application which contains only Front-End as well as Back-End is known as Dynamic Web Application.

==> To develop Dynamic Web Applications we required Advance Java (which includes Java, SQL and Web Technologies).

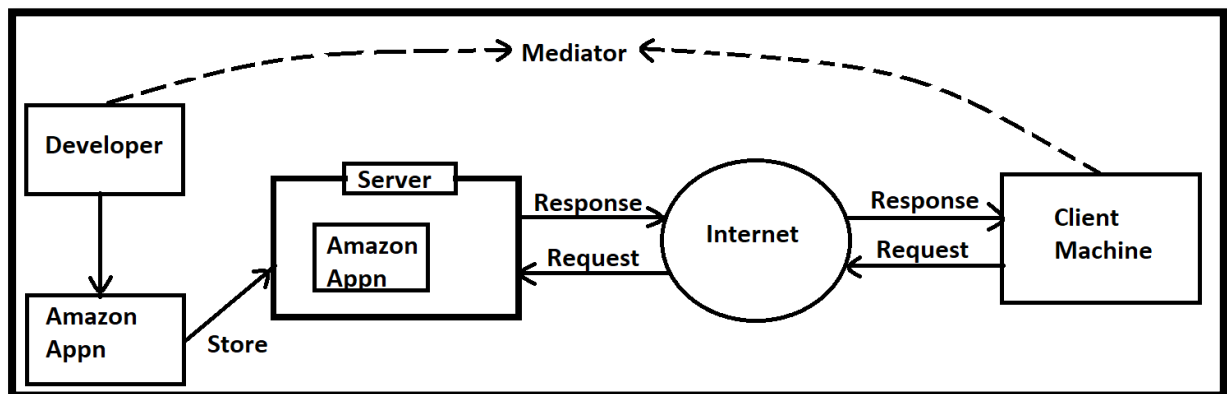
## **SERVER**

==> Server is a machine which is used to store Web Application.

==> Servers are the mediator between Developer and User.

==> Developer is using the server to store the Web Applications.

==> User is using the server access Web Applications by sending Request and Response.



**DATE : 09-08-2022**

## **3.) Mobile Application**

==> The Application which needs to download and install in Mobile is known as Mobile Application

==> Mobile Applications are developed by using FLUTTER Technology.

==> To use mobile applications in mobile we need Operating System.

Example: Android, IOS, Linux, Java

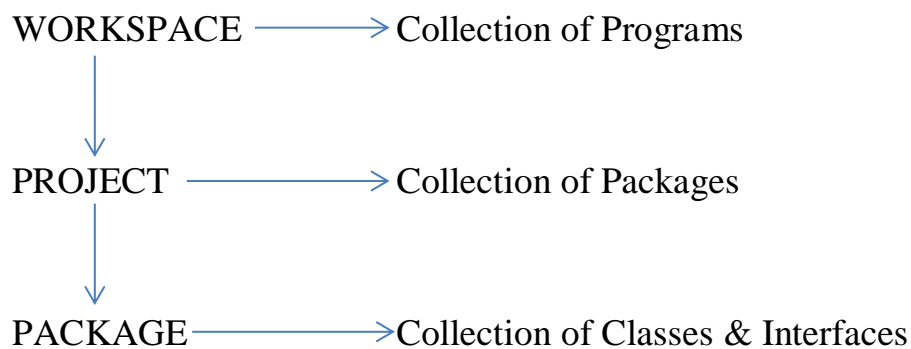
#### **4.) Distributed Application**

==> The applications which are interconnected is known as Distributed Applications.

==> Distributed Applications are developed by using Cloud Technology.

Example: Google(Gmail,GMap,GDrive,GPhotos,Youtube),  
Amazon(AmazonPay,Prime,Music,Shopping)

#### **WORKING WITH ECLIPSE**



#### **PROGRAM**

**package** demo;

**public class** Welcome {

**public static void** main(String[] args) {

        System.out.println("Welcome to Advance Java");

    }

}

#### **OUTPUT**

Welcome to Advance Java

## RANDOM CLASS

==> We can generate two types of numbers

1. Sequential Numbers
2. Random Numbers

### Sequential Numbers

==> The numbers which are present one after the another (or) one-by-one is known as Sequential Numbers.

==> To generate Sequential Numbers we have to use **for** loop.

Example:

```
for(int i=0 ; i<n ; i++)  
{  
    System.out.println(i);  
}
```

### Random Numbers

==> The numbers which are present in non-sequential order is known as Random Numbers

==> To generate Random Numbers we have to use **Random** class which is present in **java.util** package.

```
package demo;
```

```
import java.util.Random;
```

```
public class RandomNumbers {
```

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

```
        Random r = new Random();
```

```
        int n = r.nextInt(100);
```

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

```
    }
```

```
}
```

### OUTPUT

## **GENERATION OF OTP**

### **Q. WAP TO GENERATE 4 DIGITS OTP**

```
package demo;

import java.util.Random;

public class Generate4DigOTP {

    public static void main(String[] args) {

        Random r = new Random();
        int n = r.nextInt(10000);
        System.out.println("The Original OTP is : "+n);
        if(n<1000)
        {
            n+=1000;
        }
        System.out.println("The OTP is : "+n);
    }
}
```

### **OUTPUT**

The Original OTP is: 415

The OTP is: 1415

### **Q. WAP TO GENERATE 6 DIGITS OTP**

```
package demo;

import java.util.Random;

public class Generate6DigOTP {

    public static void main(String[] args) {

        Random r = new Random();
        int n = r.nextInt(1000000);
        System.out.println("The Original OTP is : "+n);
        if(n<100000)
    }
```

```

        {
            n+=100000;
        }
        System.out.println("The OTP is : "+n);
    }
}

```

## OUTPUT

The Original OTP is: 39853

The OTP is: 139853

## Q. WAP FOR OTP VERIFICATION

```
package demo;
```

```
import java.util.Random;
```

```
import java.util.Scanner;
```

```
public class OTP_Verification {
```

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

```
        Random r = new Random();
```

```
        int n = r.nextInt(1000000);
```

```
        if(n<100000)
```

```
        {
```

```
            n+=100000;
```

```
        }
```

```
        System.out.println("Your OTP is : "+n);
```

```
        Scanner sc = new Scanner(System.in);
```

```
        System.out.println("Enter the OTP : ");
```

```
        int newOTP = sc.nextInt();
```

```
        if(n == newOTP)
```

```
        {
```

```
            System.out.println("OTP verified Successfully....!");
```

```
        }
```

```
        else
```

```
        {
```

```
            System.err.println("Invalid OTP....!");
```

```
        }
```

```
    }
```

```
}
```

## OUTPUT

Your OTP is: 769274

Enter the OTP:

769274

OTP verified Successfully....!

==> Random Class is used to generate Random Numbers.

==> Random Class is one of the Inbuilt class which is present in java.util package.

==> To generate Random Numbers we have to use Random class nextInt() method.

==> nextInt() is the non-static method and with argument method.

==> nextInt() method can be called by using Random class reference variable.

==> The return type of nextInt() method is Integer.

==> nextInt() is a Polymorphic method which is present in Random class as well as Scanner class.

==> Random class nextInt() method is used to generate Random Numbers.

==> Scanner class nextInt() method is used to read the Integer values in RunTime.

==> To print a warning message in our program we have to use

System.*err*.println();

==> It will print output as an error message in red colour.



## **PACKAGE CREATION**

==> We can create an application in two ways based on industries.

1. Commercial Application
2. Organizational Application

### **1. Commercial Application**

==> The applications which can be accessed by anyone is known as Commercial Application.

**Example:** Facebook, Phonepe, Gpay etc

### **2. Organizational Application**

==> The applications which can be accessed only by authorized people is known as Organizational Application.

**Example:** Bank Application

### **Syntax to create a package**

Way of an application / Company\_name / Project\_name

**Example:** 1. com.google.gmail

2. org.jspiders.hr

---

### **NOTE:**

==> ‘.’ is used to create a folder inside a folder

---

## **JAR FILE**

==> JAR file stands for “**Java ARchive**”.

==> JAR files are the compressed version of Java Programs.

==> Compressed version of files (or) folders is known as ZIP file.

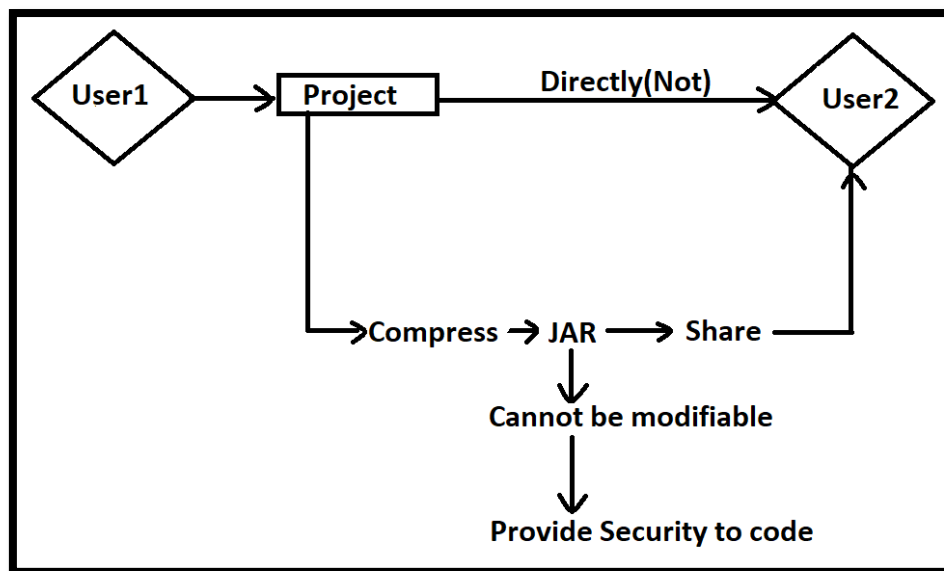
==> ZIP files are used to share large data files with one person to another person.

==> JAR files are the mediators between two developers.

==> The process of sharing java programs from one developer to another developer by converting as JAR file is known as Communication Medium.

==> JAR files are used to provide a security for Java Programs. Because JAR files cannot be modifiable.

==> JAR files are used to share Java programs from one developer to another developer.



## **STEPS TO CREATE JAR FILES**

1. Select the Project and Right click on the Project.
2. Select the option Export
3. In wizard or suggestion box search as ‘JAR’.
4. Select an option Java/JAR File.

5. Specify the location to save the JAR file
6. Mention the file name and click on Save.
7. Click on Finish.

==> Java project will contain two folders.

1. Source folder

2. Bin folder

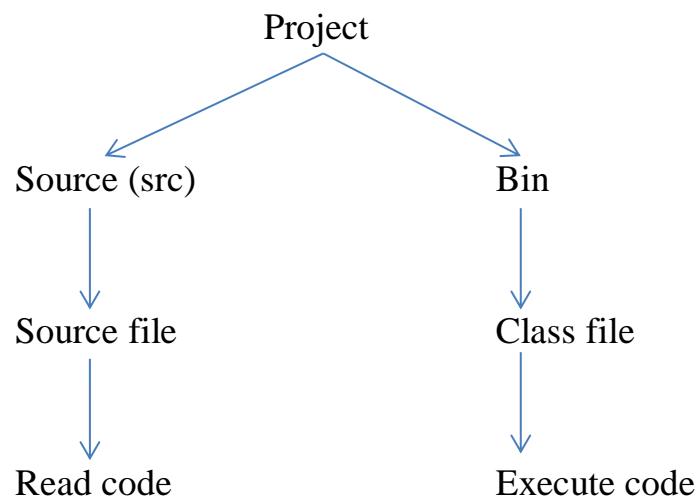
==> Source folder contain source file of java programs.

==> Source files are used to Read the code.

==> Bin folder contains class files of Java program.

==> Class files are used to execute the code.

==> JAR file will contain only Class files(Developer can only execute).



**DATE : 13-08-2022**

## **JAVA BUILT PATH**

==> We have to perform Java build path to make use of the JAR file which is shared from one developer to another developer.

==> Without performing we cannot use any type of JAR files.

### **STEPS TO PERFORM JAVA BUILD PATH**

1. Copy the JAR file and paste it in the project.
2. Right click on the project then Select the option 'Properties'.
3. Select the Java build path and Click on 'Libraries' (The option which is present on the top of the window).
4. Select 'Add JARs' and choose the Jar file which is present in respective project.
5. Click on 'Ok' and Click on 'Apply and Close'.

### **HOW TO REMOVE MODULE PATH ERROR**

1. Select the Project and Click on the project
2. Go for Properties and Select Java Build Path
3. Double click on JRE System Library and Select Java1.8 version in Execution environment
4. Click on 'Finish' and 'Apply and Close'.

**DATE : 15-08-2022**

## **TYPES OF LIBRARIES**

==> We have 2 types of Libraries

1. JRE System Library
2. Referenced Libraries

### **1. JRE System Library**

==> All predefined packages and classes are present in JRE System Library.

### **2. Referenced Libraries**

==> All external JAR files are stored in Referenced Libraries.

## CONVERSION OF FILES

==> There are 2 conversions in Java.

1. Source file to Class file.

2. Class file to Source file.

==> After creation of Java file we are using Compiler to create Class file.

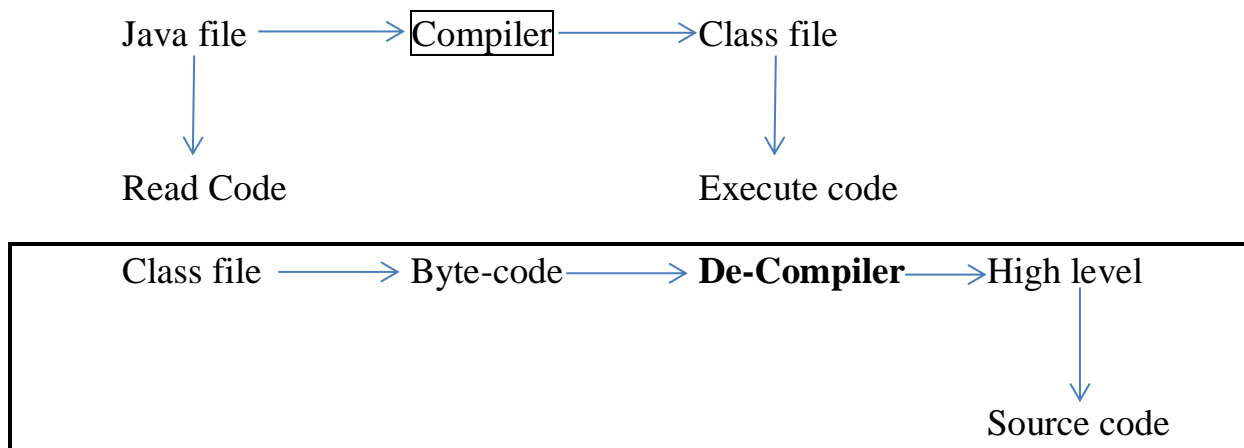
==> Project can contain two types of files, Source file as well as Class file.

==> Source file is to read the code.

==> Class file is used to execute the code

==> After converting the project into JAR file, JAR file will not contain Source code to read a code. So we need to go with De-Compiler.

==> De-Compiler is used to convert Class file into Source file for readability purpose.



## STEPS TO INSTALL DE-COMPILER IN ECLIPSE

1. Go to 'Help' in Eclipse and Select 'Eclipse Marketplace'.
2. Search as 'jd' in search bar.
3. Select and Install Enhanced Class De-Compiler.
4. While installing accept the license.
5. Accept all the folders which are selected.
6. Select 'Always Trust Content' and Click on 'Continue'.
7. Click on 'Finish'.
8. Restart the Eclipse.

## **HOW TO OPEN CLASS FILE WHICH IS PRESENT IN JAR FILE WITH THE HELP OF DE-COMPILER**

1. Open the Class file which is present in De-Compiler.
2. Select the class file and right click on it.
3. Select the option as 'Open Class with'.
4. Choose 'Fern-Flower'.
5. Your Class file will get open.

**DATE : 16-08-2022**

## **PERSPECTIVE**

==> Perspectives are used to decide which type of application we are developing.

==> Perspectives are used to decide type of an application which can be developed in Eclipse IDE.

==> There are 2 types of perspectives

1. Java Perspective
2. JavaEE perspective

### **1. Java Perspective**

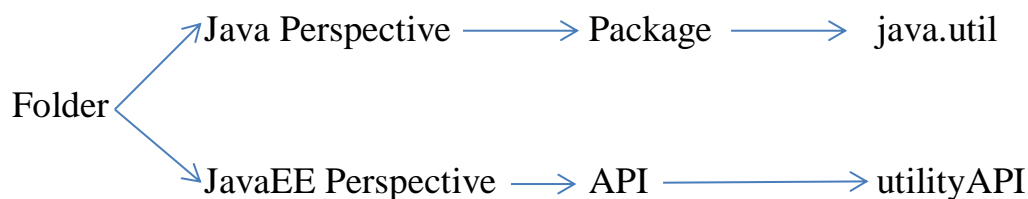
==> Java Perspective is used to develop the application which will work without internet (Standalone Application).

==> In Java Perspective folders are represented as Packages.

### **2. JavaEE Perspective**

==> JavaEE Perspective is used to develop the application which will work with internet (Web Application).

==> In JavaEE Perspective folders are represented as APIs.



## HOW TO SELECT JAVAEE PERSPECTIVE IN ECLIPSE

==> Open Eclipse —> Click the window Right side of 'Quick access'

==> Window —> Perspective —> Open Perspective

Open ← JavaEE ← Others

==> Help —> Install new softwares —> Drag the 'Work with' tagbar

Install ← Click on Web,XML ← Click on All available sites

---

### NOTE:

==> JavaEE stands for 'Java Enterprise Edition'.

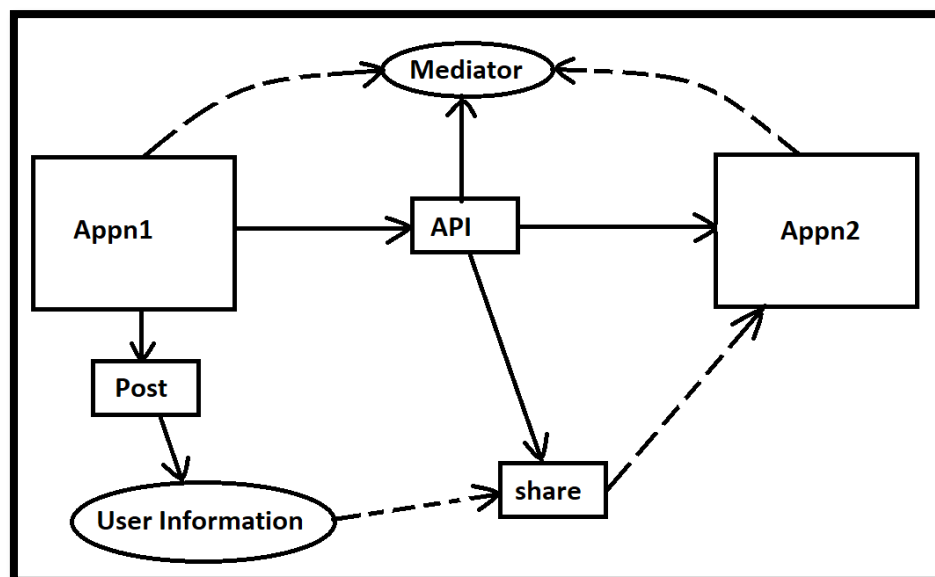
==> Advance Java can be called as JEE / J2EE.

---

**DATE : 17-08-2022**

## (API) APPLICATION PROGRAMMING INTERFACE

==> API is used to share the information from one application to another application.



==> API is a mediator between two applications.

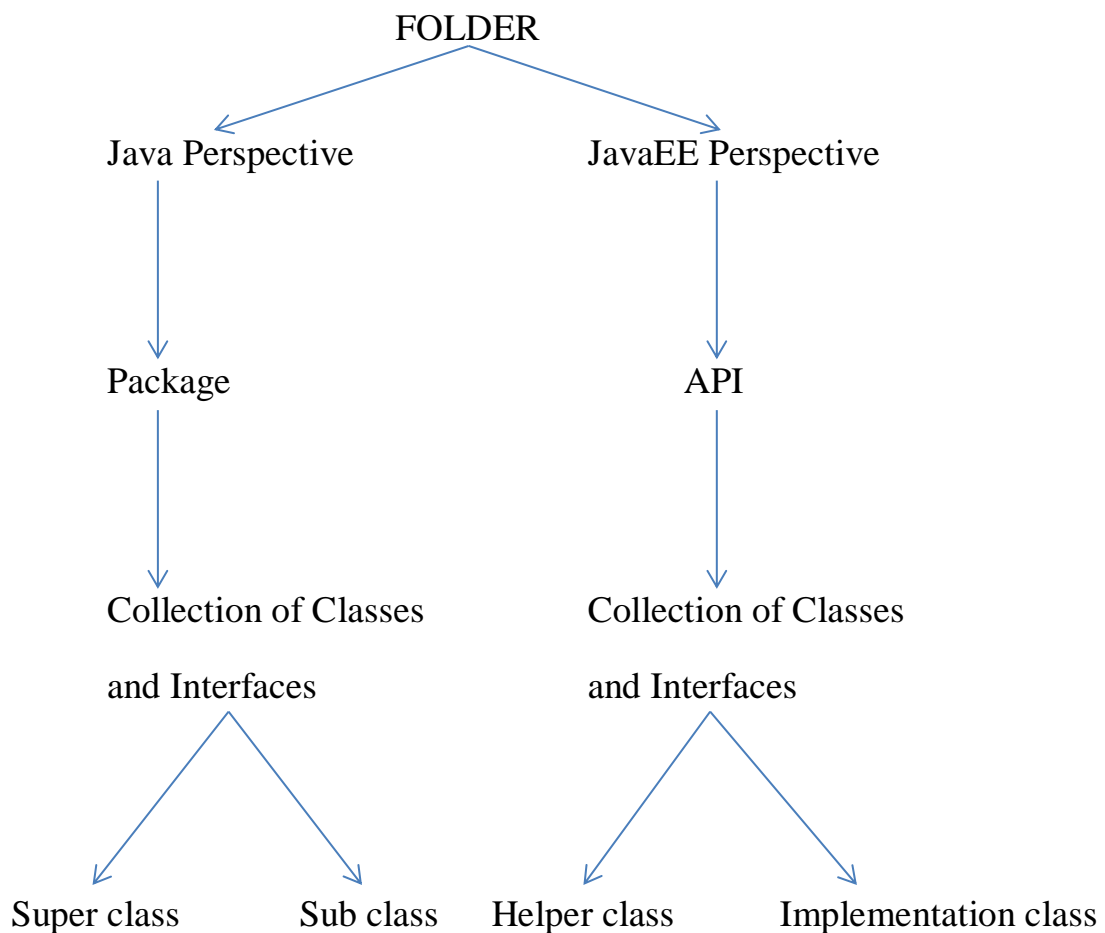
==> The process of sharing the information from one application with another application technically referred as Communication Medium.

---

**NOTE:**

==> The API terminology is present in JavaEE Perspective.

---



==> The API contains collection of classes and interfaces.

==> Classes are defined as Helper and Implementation class.

JAVA PERSPECTIVE	JAVAAE PERSPECTIVE
Super class	Helper class
Sub class	Implementation class
Interface	Interface
Package	API



==> The classes which contain common properties is known as Helper class.

==> The classes which contain specific properties is known as Implementation class.

==> The collection of non-static, abstract, static and final variables is known as Interface.

==> The combination of Helper class and Implementation class, Interface is known as Application Programming Interface.

## **DRIVERS**

==> Drivers are nothing but Translators.

==> Drivers are used to convert one type of language into another type of language.

==> The process of converting one type of language into another type of language is known as Translation.

Examples:

==> Source code —————> Compiler —————> Byte code

==> Byte code —————> De-Compiler —————> Source code

==> Byte code —————> JVM —————> Binary code

**DATE : 18-08-2022**

## **JDBC ARCHITECTURE**

==> JDBC stands for “**Java DataBase Connectivity**”

==> JDBC is used to connect Java application with Database application.

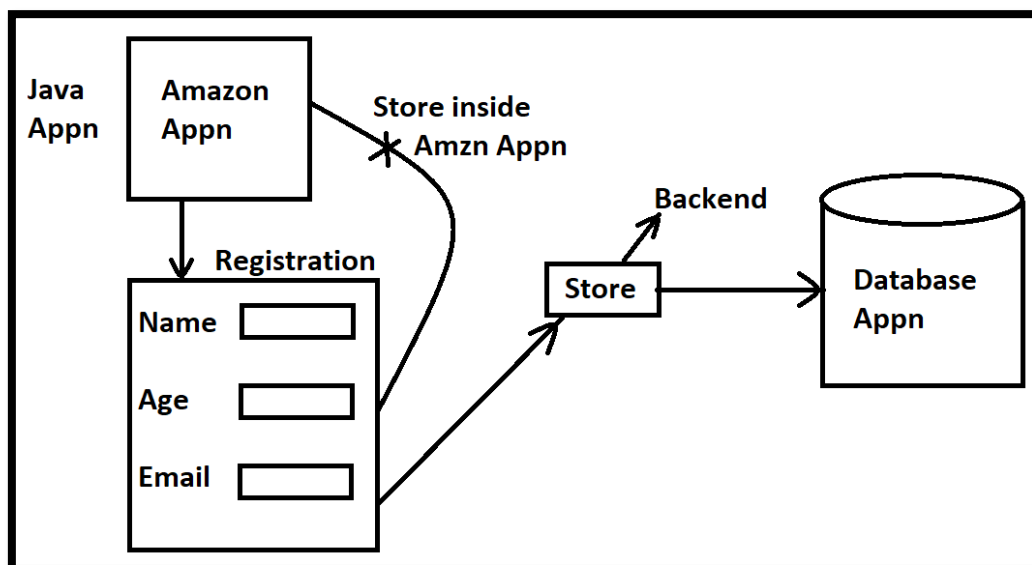
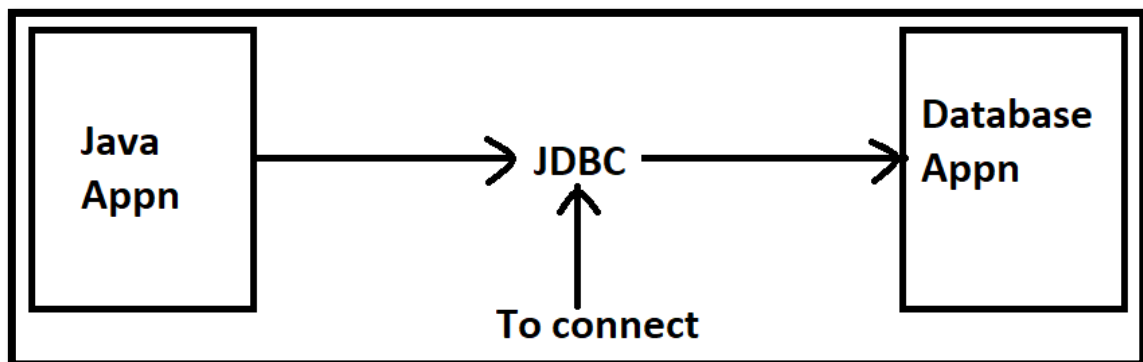
==> It is used to store the data inside Database.

==> The application which we are using to collect the user information it cannot be stored the data by itself. So we need to go with Database.

==> To understand the process of JDBC Architecture we have to follow major four components.

## COMPONENTS OF JDBC ARCHITECTURE

1. Java Application
2. JDBC API
3. JDBC Drivers
4. Database



### 1. Java Application

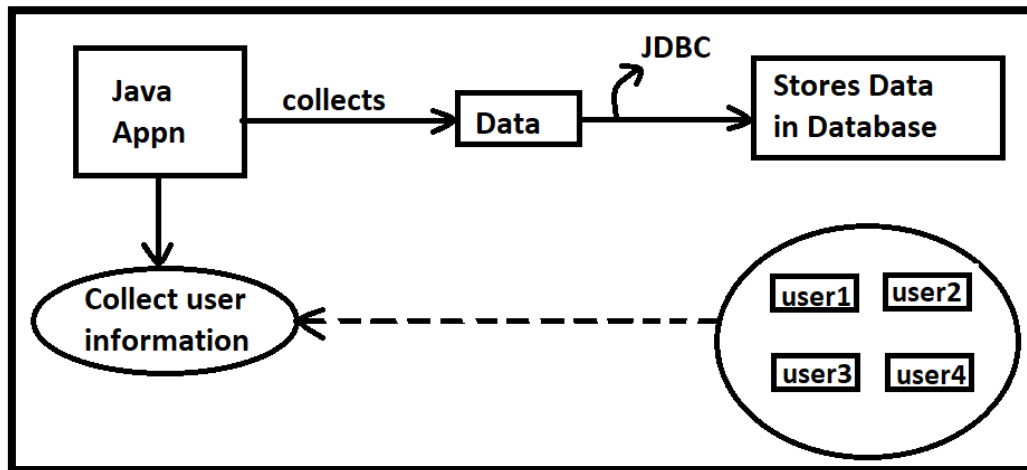
==> Java application is used to collect user information.

==> The data which is collected by the Java application we have to store it in database.

==> We can collect the data by the Java application in two ways.

Scanner → Basic level

HTML → Higher level



## 2. JDBC API

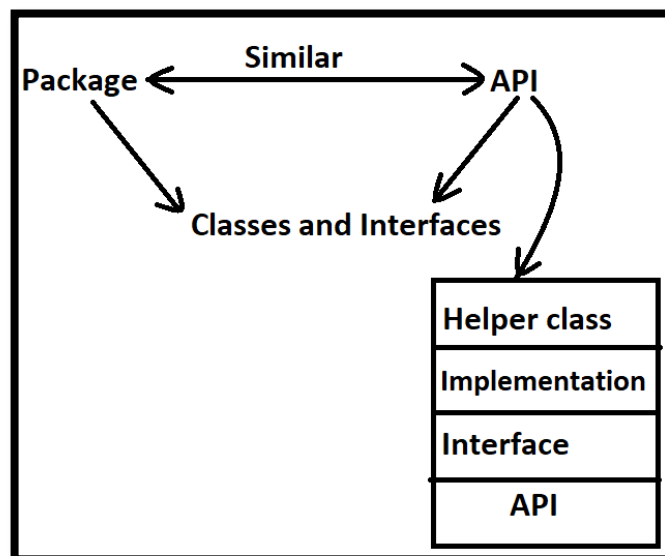
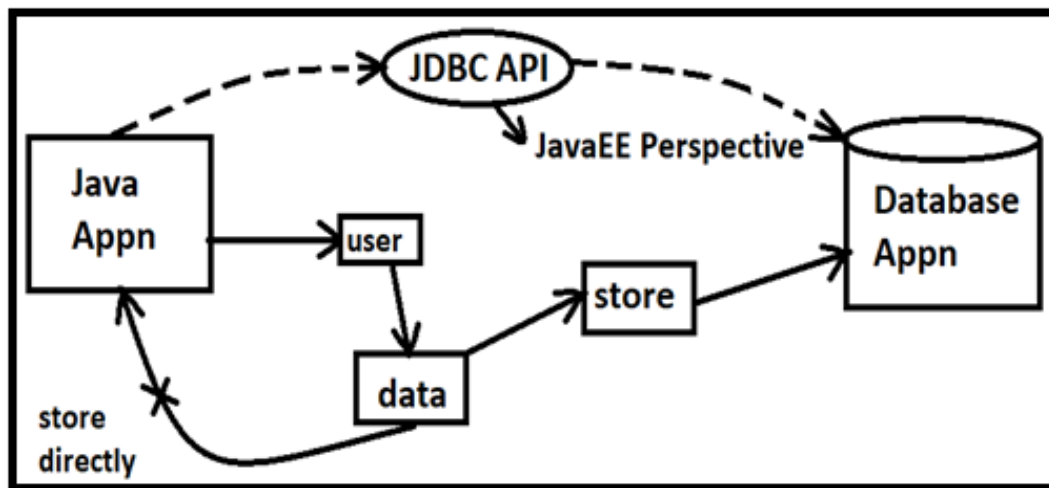
==> JDBC API is used to sharing the user information from Java application to Database application.

==> The JDBC API will represents in JavaEE perspective.

==> JDBC API is equal (or) corresponds of **java.sql** package.

==> Classes and Interfaces which are present in JDBC API (java.sql package).

- 1. Driver Manager → Helper class
  - 2. Connection
  - 3. Statement
  - 4. Prepared Statement
  - 5. Callable Statement
  - 6. Result Set
- } Interfaces



### **3. JDBC DRIVERS**

=> JDBC Drivers are used to convert Java information into SQL information (or) Java Language into SQL language (or) Java Calls into SQL Calls

#### **NOTE:**

The data which is collected in Java application is present in Java language, Java language cannot be understandable by Database, It is understand only Query language, So to make Database to understand Java instruction we need to convert into SQL instruction, So we make use of JDBC drivers.

#### **4. DATABASE**

==> Database is used to store the data permanently.

==> In Database we have two types.

1. Oracle
2. MySQL


==> To store the data in the Database we require DBMS(Data Base Management System) to access (or) manage the data.

==> To working with MySQL Database we need RDBMS(Relational Data Base Management System) to manage and store the data in table format.

==> To get the RDBMS in system and working with MySQL we have to install **SQLyog** Application.

==> To perform the operations on the Database we have to use SQL(Structured Query Language).

==> We can perform multiple operations by using SQL. They are

- |           |   |                        |
|-----------|---|------------------------|
| 1. Create |  | <b>CRUD</b> operations |
| 2. Read   |   |                        |
| 3. Update |   |                        |
| 4. Delete |   |                        |

**DATE : 19-08-2022**

#### **WORKING WITH SQLYOG APPLICATION**

==> We approach SQLyog application to communicate with MySQL database.

==> To communicate with any database we require the suitable RDBMS.

==> RDBMS cannot be accessed directly, they stored inside an application inorder to use that we have to install the specific application.

Types of Database	Software	DBMS
Oracle	SQL Plus	RDBMS with Oracle
MySQL	SQLyog	RDBMS with MySQL

**DATE : 20-08-2022**

## **SCHEMA**

==> The process of creating database inside a database is known as Schema.

### **STEPS TO CREATE DATABASE INSIDE MYSQL DATABASE**

1. Open the SQLyog application
2. Select root@localhost. Right click on it.
3. Select an option as “Create Database”.
4. Specify the database name and click on Finish.
5. We have to use datatype to store data inside a table.
6. Datatypes are different compare to Java language and MySQL database.

	<b>EmpId</b>	<b>EmpName</b>	<b>EmpSal</b>	<b>DeptNo</b>
Java	int	“String”	double	Int
SQL	int	“variable”	double	Int

7. Scale (or) Precision are mandatory when we are using “double” datatype.

### **STEPS TO CREATE TABLE INSIDE DATABASE**

1. Select the Database and Right click on it.
2. Select an option called as “Create Table”.
3. Specify the field values by providing datatype, fieldname, length, primary key, NOT NULL values.
4. After the field values click on “Create Table”.
5. Specify the table name and click on Finish.
6. We will get a message as “Table created successfully”.

**DATE : 22-08-2022**

==> On the database we can perform two operations.

1. Read operation
2. Write operation

==> To perform write operation we are using insert, update, delete queries

==> To perform read operation we are using select query.

==> To alter the table

Select Table → Right click → Alter table option

==> Primary key value which is used to allow unique values in the specific columns.

==> NOT NULL is used to doesn't allow NULL values in the specific column.

==> After creating the table user can see his data by setting "View data".

### **STEPS TO VIEW DATA**

Select Table → Right click → Select View Data option

==> To see the modification on the database for values we have to keep on refreshing the table.

### **SYNTAX FOR SELECT QUERY WITH EXAMPLE**

**SELECT \*/COLUMN\_NAME  
FROM DATABASE\_NAME.TABLE\_NAME  
WHERE <CONDITION>;**

**Eg:** select \* from teja15.employee where deptNo=10;

### **SYNTAX FOR INSERT QUERY WITH EXAMPLE**

**INSERT INTO DATABASE\_NAME.TABLE\_NAME  
INTO VALUES(V1,V2,.....,Vn);**

**Eg:** insert into teja15.employee values (101,'RAM',50000,'HYD');

### **SYNTAX FOR UPDATE QUERY WITH EXAMPLE**

**UPDATE DATABASE\_NAME.TABLE\_NAME  
SET COLUMN\_NAME = VALUE  
WHERE <CONDITION>;**

**Eg:** update teja15.employee set location = 'DLH' where location = 'HYD';

## SYNTAX FOR DELETE QUERY WITH EXAMPLE

**DELETE FROM DATABASE\_NAME.TABLE\_NAME**

**WHERE< CONDITION >;**

**Eg:** delete from teja15.employee where empId=101;

**DATE : 23-08-2022**

**Q. CREATE A CAR TABLE WHICH CONTAINS 5 COLUMNS**

**CARNO, CAR COLOUR, CAR BRAND, PRICE, MILEAGE**

**→ INSERT ABOVE 7 RECORDS**

**→ UPDATE THE CAR COLOUR AS WHITE IF CAR PRICE IS ABOVE 2CRORES**

**→DELETE THE CAR RECORDS IF THE CAR MILEAGE IS LESS THAN 10**

**→ UPDATE CARNO AS 'TS06EJ7813' IF CAR COLOR IS GREEN**

**→ DELETE CAR RECORDS IF THE CARNO ENDING WITH '0'.**

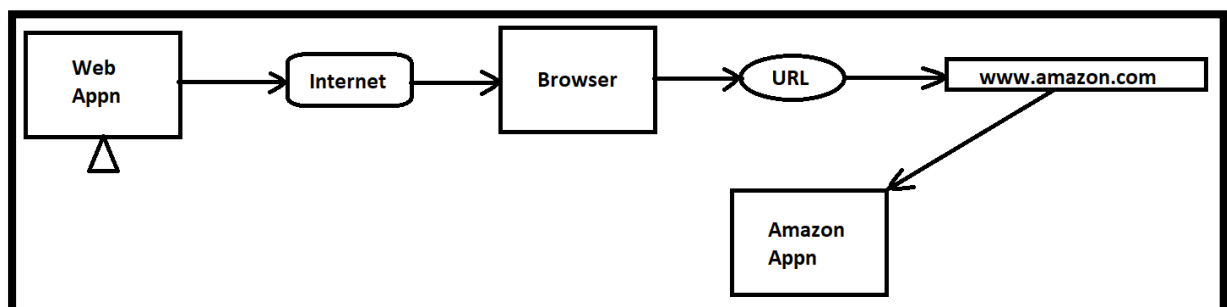
**DATE : 24-08-2022**

## URL

==> URL stands for 'Uniform Resource Locator'.

==> URL contains complete information about the application.

==> URL helps to the API to share the information from one application to another application.



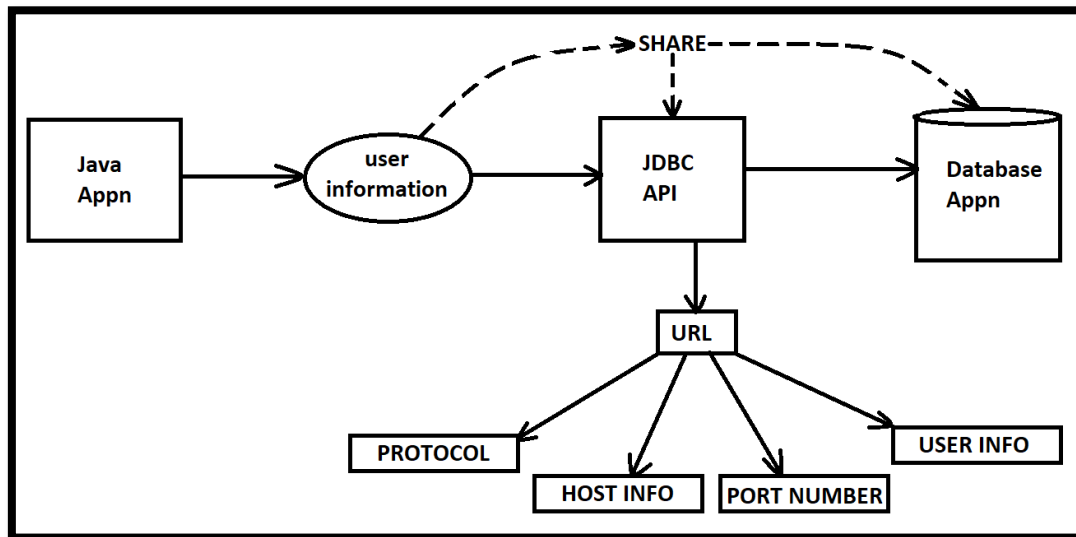
==> It is mandatory to use URL to connect with Java application and Database application.



==> Every database will contain four information.

1. Protocol
2. Host information
3. Port number
4. User information

==> These all the information present in the URL.



## 1. PROTOCOL

==> Set of rules which must be followed to access an application.

==> User information are storing inside database because we need to connect with database.

==> The database protocol is JDBC.

==> In database we have two types

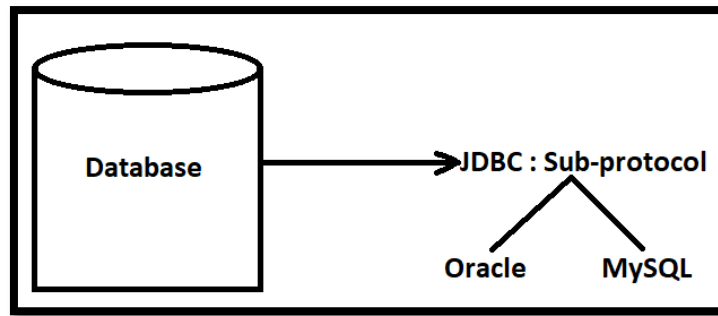
1. Oracle
2. MySql

==> To give more clarity about the URL we have to use Sub-protocol.

==> Syntax for Protocol is **Protocol : Sub-Protocol**

**Ex:** JDBC:MySql, JDBC:Oracle

==> The protocol for server is '**https**'.



## **2. HOST INFORMATION**

==> Host Information is the way of accessing an application.

==> We have two types of host information.

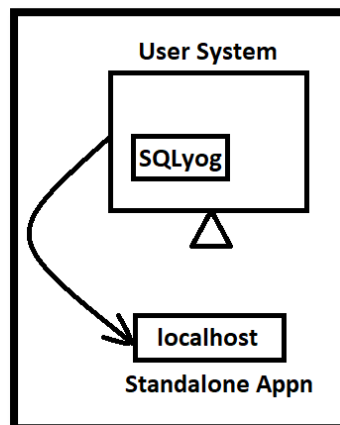
1. Local Host

2. Remote Host

### **1. Local Host**

==> If an application is present inside user system then we are going to use Local host.

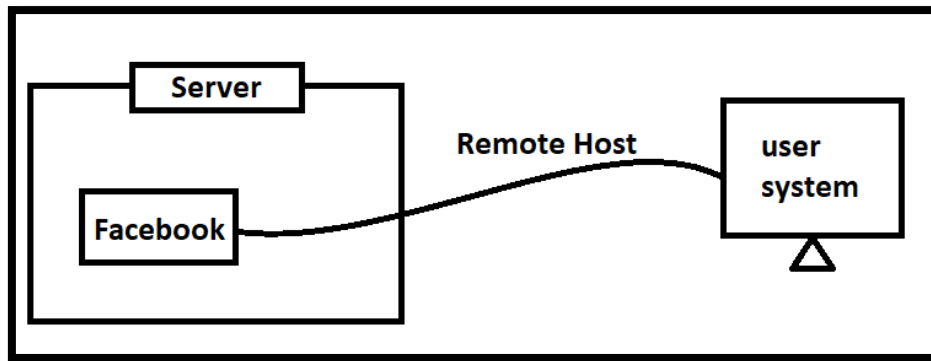
==> To access Standalone application we make use of Local host.



### **2. Remote Host**

==> If an application is present inside server we are going to use Remote host.

==> To access Web application we make use of Remote host.



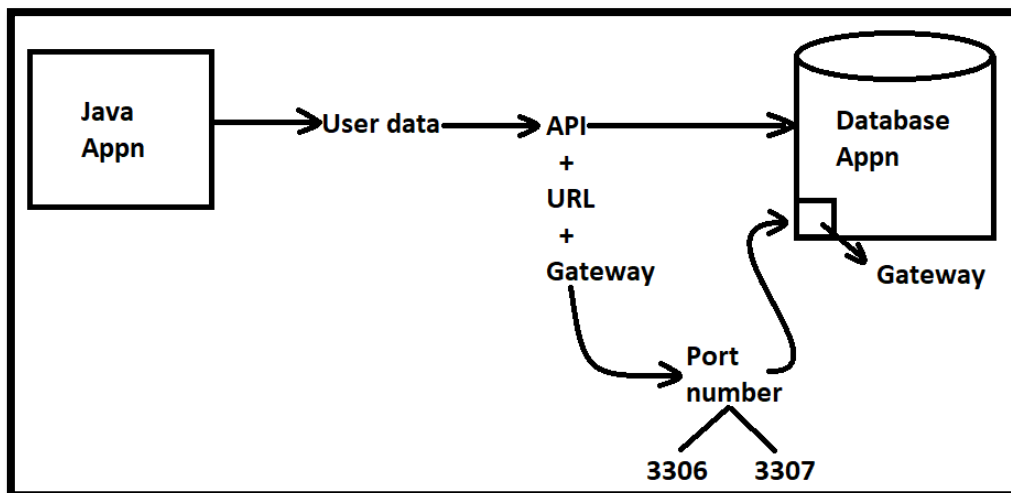
### **3. PORT NUMBER**

==> Port number is used to open the gateway of the Database.

==> Each and every database they contain the gateway.

==> To store the user data inside database first we need to get in, for that we have to open the gateway by using port number.

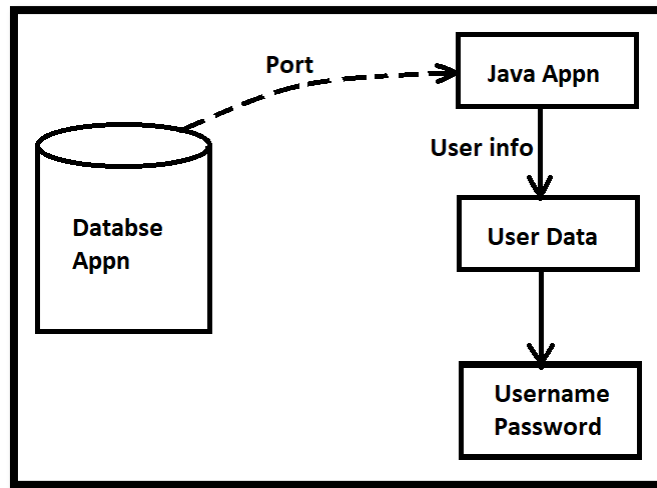
==> For a MySQL server, there are two port numbers 3306 and 3307



### **4. USER INFORMATION**

==> It is used to provide more security for an application.

==> All the applications are already present with basic security by port number. To increase more security we make use of user information.



==> The combination of protocol, host information, port number, user information is known as URL

==> Syntax for URL is

**Protocol : Sub-Protocol : // Host info : Port Number ? User info**

#### **For Example 1,**

If programmer is using MySQL database the protocol is “jdbc : mysql” and host information is “localhost” ( Because SQLyog application is present in the user system) and port number will be “3306 (or) 3307” and the username for MySQL database is “root” and password is “12345”.

`jdbc : mysql : // localhost : 3306 ? user = root & password = 12345`

#### **For Example 2,**

If programmer is using Oracle database the protocol is “jdbc : oracle” and host information is “localhost” and the port number of oracle database is “1521” and the username is “SCOTT” and password is “TIGER”.

`jdbc : oracle : // localhost : 1521 ? user = SCOTT & password = TIGER`

## **STEPS OF JDBC**

==> We can develop applications in two approach.

1. Without using Inbuilt code
2. With using Inbuilt code

==> Without using inbuilt code if we develop an application it will take more time.

==> With using inbuilt code if we develop an application it will take less time.

==> All the developers will always prefer to go with using inbuilt code.

## **STEPS**

1. Establishing Connection
2. Create a Platform
3. Execution of Query
4. Process of Resultant data
5. Close Connection

**Q.**

**→CREATE A COLLEGE INTERFACE WITH ADD STUDENT, REMOVE STUDENT, SEARCH STUDENT, DELETE STUDENT WITH INCOMPLETE METHODS**

**→CREATE COLLEGE IMPLEMETATION CLASS AND COMPLETE ALL THE COLLEGE INTERFACE METHODS**

**→CREATE A HELPER CLASS AS DEPARTMENT WITH A STATIC METHOD WHICH IS RESPONSIBLE TO CREATE AN OBJECT OF COLLEGE IMPLEMENTATION CLASS AND THE RETURN TYPE OF THE METHOD IS COLLEGE INTERFACE**

**→ CREATE A MAIN CLASS AND CALL THE DEPARTMENT CLASS STATIC METHOD AND STORE THE REFERENCE INSIDE COLLEGE INTERFACE REFERENCE VARIABLE AND CALL ALL THE METHODS OF COLLEGE INTERFACE.**

```
package org.jsp.jdbc;
```

```
public interface College {
```

```
    void addStudent();  
    void removeStudent();  
    void searchStudent();  
    void deleteStudent();
```

```
}
```

```
package org.jsp.jdbc;
```

```
public class CollegeImp implements College{
```

```
    public void addStudent()  
    {  
        System.out.println("Student Added successfully");  
    }  
    public void removeStudent()  
    {  
        System.out.println("Student Removed successfully");  
    }  
    public void searchStudent()  
    {  
        System.out.println("Student Searched successfully");  
    }  
    public void deleteStudent()  
    {  
        System.out.println("Student Deleted successfully");  
    }  
}
```

```
package org.jsp.jdbc;
```

```
public class Department {
```

```
    public static College getCollege()  
    {  
        College c = new CollegeImp();  
        return c;  
    }  
}
```

```
package org.jsp.jdbc;
```

```
public class CollegeMain {
```

```

public static void main(String[] args) {

    College c1 = Department.getCollege();
    c1.addStudent();
    c1.searchStudent();
    c1.removeStudent();
    c1.deleteStudent();

}
}

```

## **OUTPUT**

Student Added successfully  
 Student Searched successfully  
 Student Removed successfully  
 Student Deleted successfully

**DATE : 26-08-2022**

## **1. ESTABLISHING CONNECTION**

==> To pass the query from Java application to Database application we have to establish a connection first.

==> To establish a connection we are using Connection interface which is present in java.sql package.

==> For a Connection interface there is a helper class called as DriverManager.

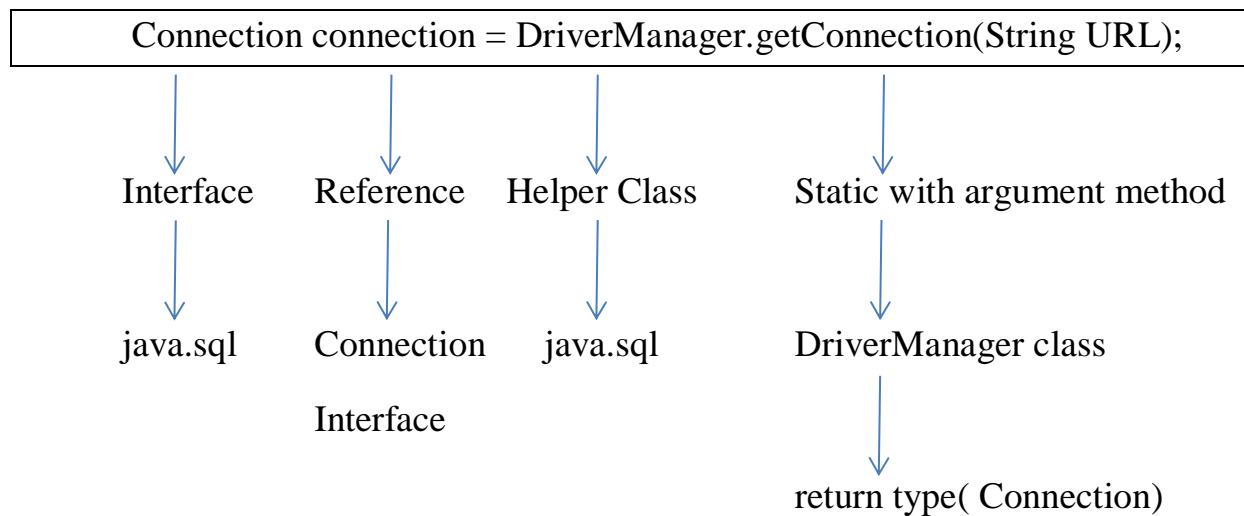
==> DriverManager having a method called as getConnection().

==> It can accept URL as argument.

1. getConnection( String URL);
2. getConnection( String URL, String user, String password);
3. getConnection( String URL , Properties prop);

==> getConnection() is responsible to throw checked exception called as SQLException.

==> The return type of getConnection() method is Connection interface.



```
package org.jsp.jdbc;
```

```
import java.sql.Connection;
```

```
import java.sql.DriverManager;
```

```
import java.sql.SQLException;
```

```
public class Establish {
```

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

```
        String url =
```

```
"jdbc:mysql://localhost:3306?user=root&password=12345";
```

```
        try
```

```
        {
```

```
            Connection c = DriverManager.getConnection(url);
```

```
            System.out.println("Connection Established
```

```
Successfully...!");
```

```
        }
```

```
        catch (SQLException e)
```

```
        {
```

```
            e.printStackTrace();
```

```
        }
```

```
    }
```

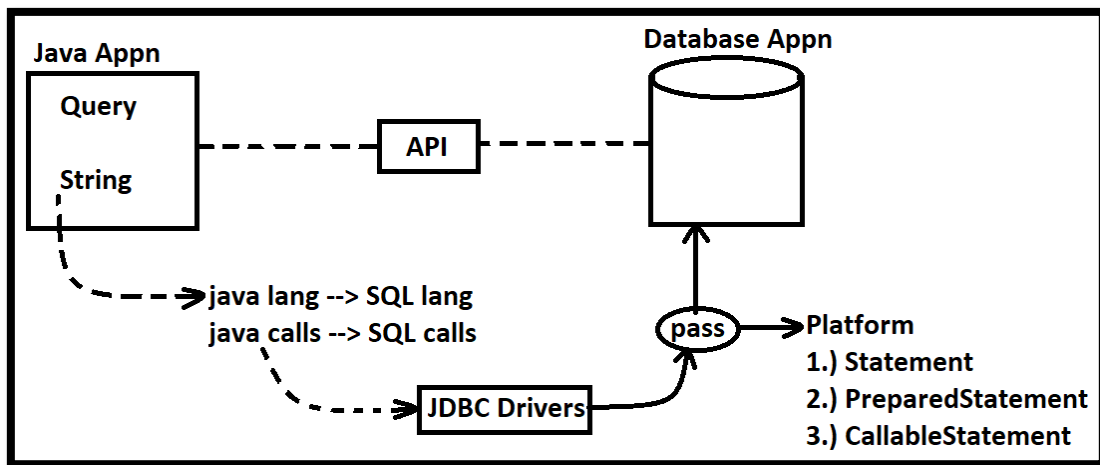
```
}
```

**OUTPUT**

Connection Established Successfully...!



## **2. CREATE A PLATFORM**



==> Platforms are used to carry the query from Java application to Database application.

==> Java application contains both Java code as well as SQL code.

==> Java compiler can understand only Java code.

==> To make SQL query to understand the compiler we make use of Platforms.

### **TYPES OF PLATFORMS**

1. Statement Platform
2. PreparedStatement Platform
3. CallableStatement Platform

==> The SQL queries which are present in Java program can be understandable by Database.

==> If a query contains hardcoded value, we prefer Statement platform.

==> If a query contains runtime values, we prefer PreparedStatement platform.

==> If a query contains hardcoded as well as runtime value, we prefer CallableStatement platform.

---

## NOTE :

The main use of platform is to carry the query from Java application to Database application.

---

## HARDCODED VALUE

=> The value which is entered by the Developer is known as Hardcoded value.

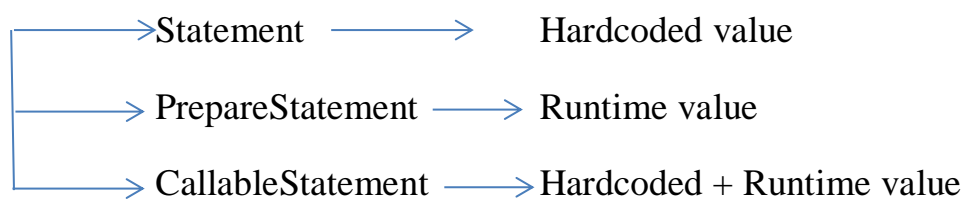
**Eg:** `int a = 50;`

## RUNTIME VALUE

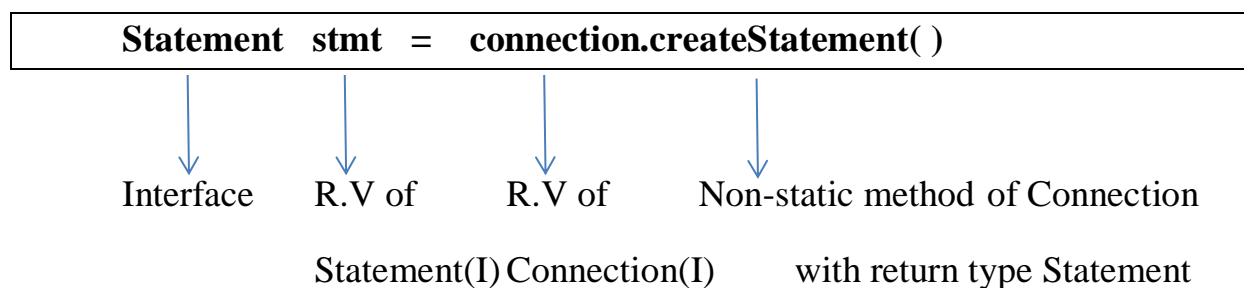
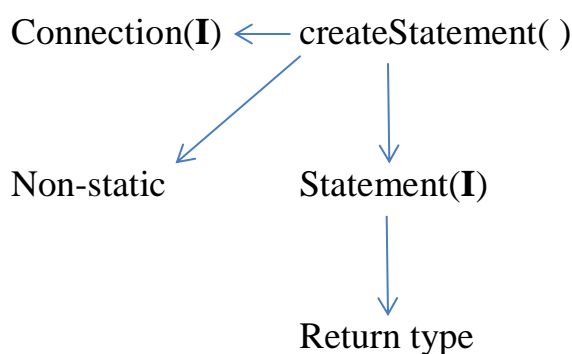
=> The value which is entered by the User is known as Runtime value.

**Eg:** `int a = sc.nextInt();`

## PLATFORMS



## HOW TO CREATE A STATEMENT PLATFORM



```

package org.jsp.jdbc;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

public class PlatformCreation {

    public static void main(String[] args) {

        String url =
"jdbc:mysql://localhost:3306?user=root&password=12345";
        try
        {
            Connection connection =
DriverManager.getConnection(url);
            Statement stmt = connection.createStatement();
            System.out.println("Platform Created...!");
        }
        catch (SQLException e)
        {
            e.printStackTrace();
        }
    }
}

```

### **OUTPUT**

Platform Created...!

### **3. EXECUTION OF QUERY**

```

package org.jsp.jdbc;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

public class QueryExecution {

    public static void main(String[] args) {

```

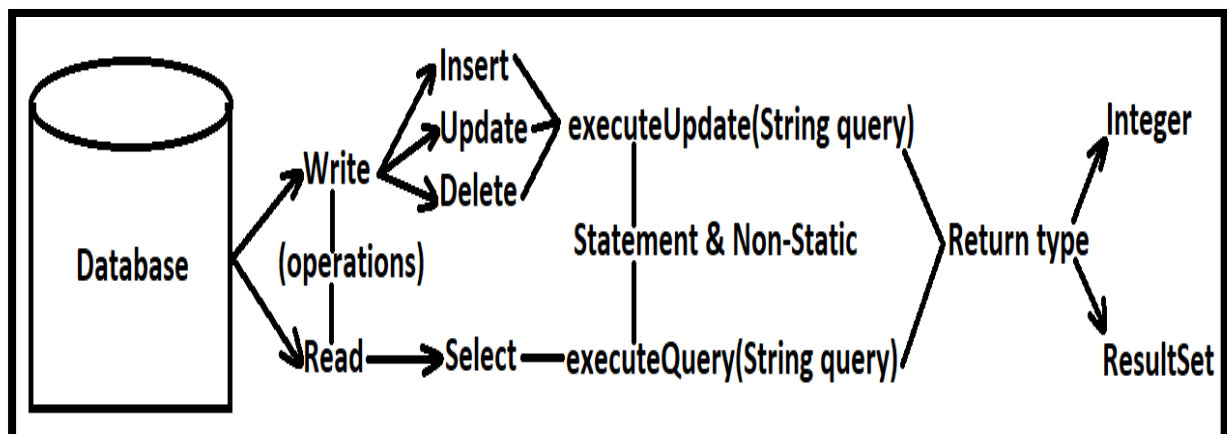
```

String url =
"jdbc:mysql://localhost:3306?user=root&password=12345";
String query = "insert into teja15.user
values(2,'TOM','9898765656','tom2222)";
try
{
    Connection connection =
DriverManager.getConnection(url);
    System.out.println("Connection Established...!");
    Statement stmt = connection.createStatement();
    System.out.println("Platform Created...!");
    stmt.executeUpdate(query);
    System.out.println("Query Executed...!");
}
catch (SQLException e)
{
    e.printStackTrace();
}
}

```

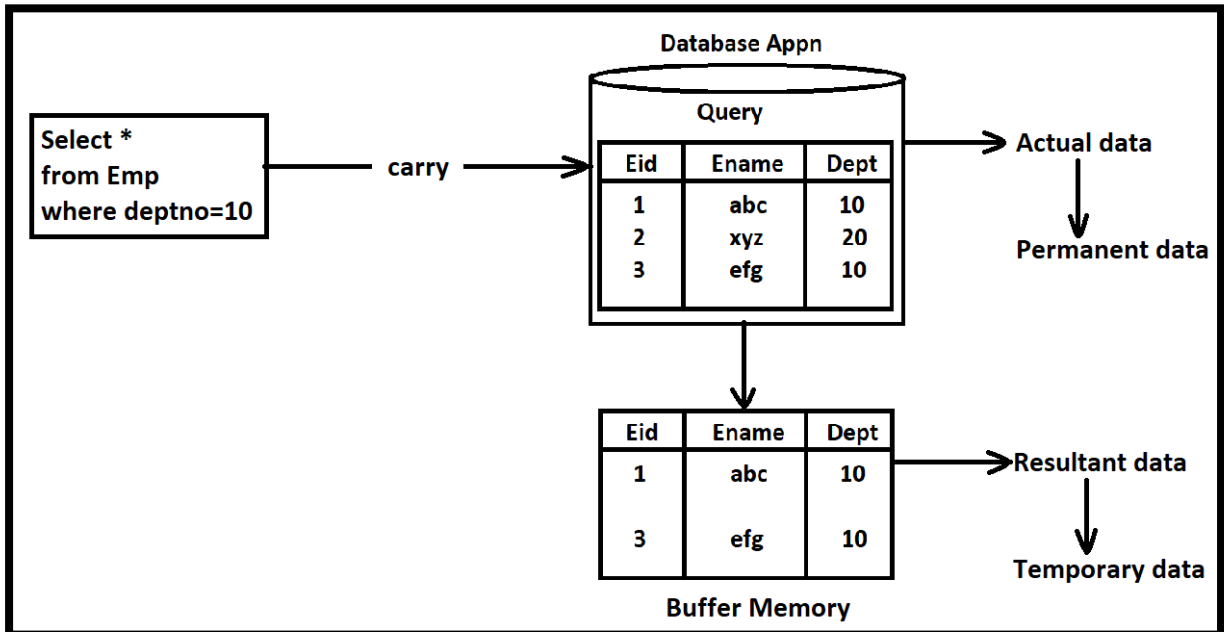
## OUTPUT

Connection Established...!  
 Platform Created...!  
 Query Executed...!



#### **4. PROCESS OF RESULTANT DATA**

==> It is an optional step in JDBC because resultant data will not be created for all the queries, it creates only for select query.



#### **NOTE:**

==> The data which is present in the database is known as Actual data.

==> Actual data is permanent data.

==> The query after reaching to the database there are three operations will get performed.

1. Compilation of query
2. Execution of query
3. Store the output in Buffer memory

==> Output will get generated only for read operation.

==> The data which is present in the Buffer memory is known as Resultant data.

==> Resultant data is temporary data.

---

## **BUFFER MEMORY**

==> Buffer memory will get created after performing the read operation.

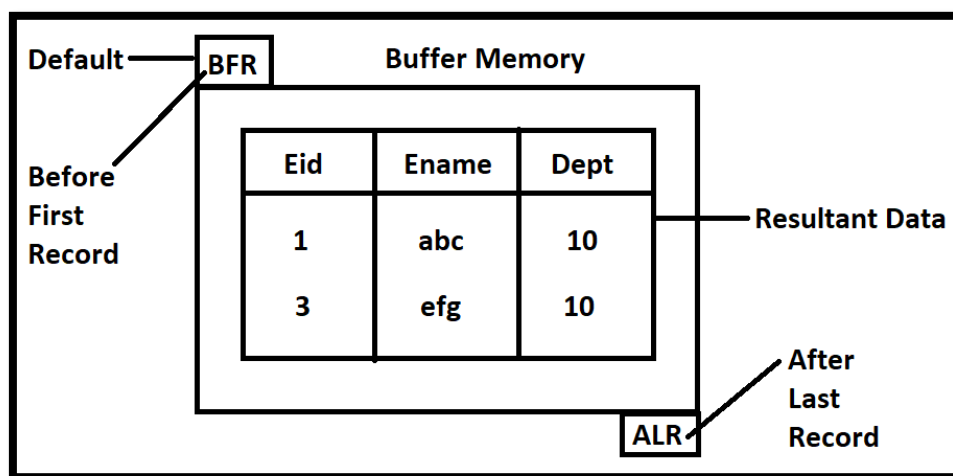
==> There are two pointers in Buffer memory

1. BFR (Before First Record)
  2. ALR (After Last Record)
- 

## **NOTE:**

==> Always cursor will present in the BFR pointer.

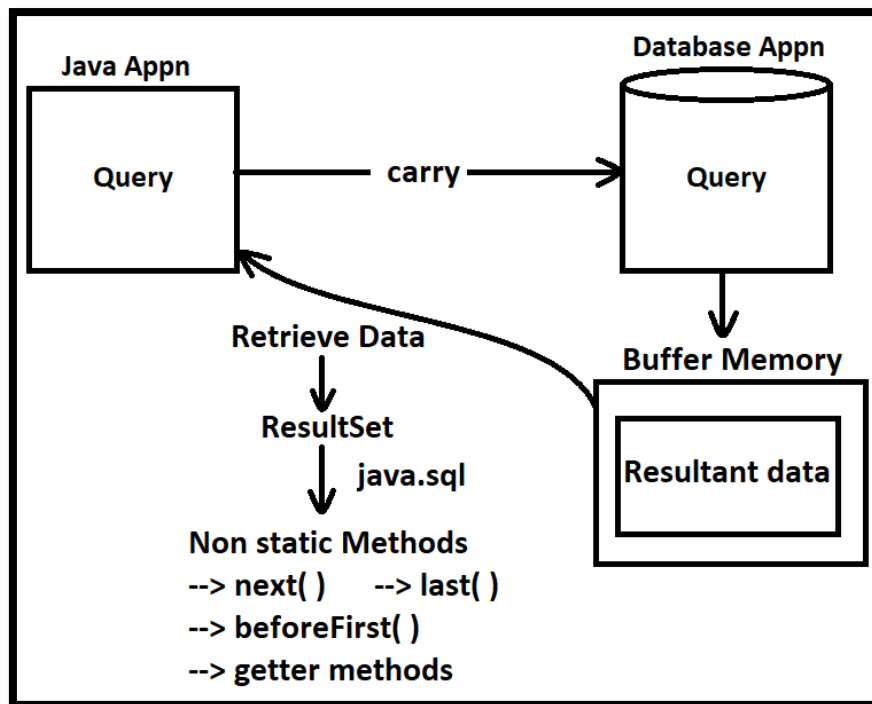
---



## **RESULT SET**

==> ResultSet is an interface which is present in java.sql package

==> ResultSet interface is used to store the resultant data.



## METHODS OF RESULT SET

1. next( ) → boolean
2. last( ) → boolean
3. beforeFirst( ) → void
4. getter → getInt(column\_number,column\_name)  
→ getString(column\_number,column\_name)  
→ getDouble(column\_number,column\_name)  
→ getFloat(column\_number,column\_name)

### 1. next( )

==> It is used to move the cursor from one record to another record, return type is boolean.

### 2. last( )

==> It is used to move the cursor to the last record which is present oin Buffer memory, return type is boolean.

### 3. beforeFirst( )

==> It is used to move the cursor from current record to BFR pointer, return type is void.

#### **4. getter methods**

==> They are used to get the data from the Buffer memory based on column data-type.

==> Getter methods are overloaded methods, we can pass any value as follows.

1. Column\_number

2. Column\_name

```
package org.jsp.jdbc;
```

```
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.ResultSet;  
import java.sql.SQLException;  
import java.sql.Statement;
```

```
public class Selection {
```

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

```
        String url =  
        "jdbc:mysql://localhost:3306?user=root&password=12345";  
        String query = "select * from teja15.employee where empId=105";  
        try  
        {  
            Connection connection =  
            DriverManager.getConnection(url);  
            System.out.println("Connection Established...!!!");  
            Statement stmt = connection.createStatement();  
            System.out.println("Platform Created");  
            ResultSet rs = stmt.executeQuery(query);  
            if(rs.next())  
            {  
                System.out.println("Valid Employee ID.....");  
                System.out.println("*****");  
                int id = rs.getInt("empId");  
                String name = rs.getString("empName");  
                double sal = rs.getDouble("empSal");
```



```

        int dept = rs.getInt("deptNo");
        String loc = rs.getString("location");
        System.out.println("Employee ID : "+id);
        System.out.println("Employee Name : "+name);
        System.out.println("Employee Salary : "+sal);
        System.out.println("Employee Dept No : "+dept);
        System.out.println("Employee Location : "+loc);
    }
    else
    {
        System.err.println("Invalid Employee ID...!!!");
    }
}
catch (SQLException e)
{
    e.printStackTrace();
}
}
}

```

## **OUTPUT**

```

Connection Established...!!!
Platform Created
Valid Employee ID.....
*****
Employee ID : 105
Employee Name : JAY
Employee Salary : 500000.0
Employee Dept No : 10
Employee Location : COCHIN

```

**DATE : 02-09-2022**

**Q. WRITE A JDBC PROGRAM FOR EMPLOYEE WHO IS WORKING IN DEPARTMENT 10**

```

package org.jsp.jdbc;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;

```

```

import java.sql.Statement;

public class SelectionMultiple {

    public static void main(String[] args) {

        String url =
"jdbc:mysql://localhost:3306?user=root&password=12345";
        String query = "select * from teja15.employee where deptNo=10";
        try
        {
            Connection connection =
DriverManager.getConnection(url);
            System.out.println("Connection Established...!");
            Statement stmt = connection.createStatement();
            System.out.println("Platform Created");
            ResultSet rs = stmt.executeQuery(query);
            System.out.println("Query Executed");
            if(rs.last())
            {
                rs.beforeFirst();
                while(rs.next())
                {
                    System.out.println("*****");
                    System.out.println("Employee ID :
"+rs.getInt(1));

                    System.out.println("Employee Name :
"+rs.getString(2));

                    System.out.println("Employee Salary :
"+rs.getDouble(3));

                    System.out.println("Employee Dept No :
"+rs.getInt(4));

                    System.out.println("Employee Location :
"+rs.getString(5));

                    System.out.println("*****");
                }
            }
            else
            {
                System.err.println("Invalid Employee ID...!!!");
            }
            connection.close();
        }
    }
}

```

```

        catch (SQLException e)
        {
            e.printStackTrace();
        }
    }
}

```

## **OUTPUT**

Connection Established...!

Platform Created

Query Executed

\*\*\*\*\*

Employee ID : 101

Employee Name : RAJA

Employee Salary : 100000.0

Employee Dept No : 10

Employee Location : DELHI

\*\*\*\*\*

\*\*\*\*\*

Employee ID : 103

Employee Name : RAM

Employee Salary : 50000.0

Employee Dept No : 10

Employee Location : CHENNAI

\*\*\*\*\*

\*\*\*\*\*

Employee ID : 105

Employee Name : JAY

Employee Salary : 500000.0

Employee Dept No : 10

Employee Location : COCHIN

\*\*\*\*\*

\*\*\*\*\*

Employee ID : 108

Employee Name : SAI

Employee Salary : 80000.0

Employee Dept No : 10

Employee Location : JAIPUR

\*\*\*\*\*

**Q. WRITE A JDBC PROGRAM TO PRINT STUDENT ID, NAME, STREAM IF MARKS GREATER THAN 70**

```

package org.jsp.jdbc;

```

```

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class SelectionMultipleStdnt {

    public static void main(String[] args) {

        String url =
"jdbc:mysql://localhost:3306?user=root&password=12345";
        String query = "select * from teja15.student where sMarks>=70";
        try
        {
            Connection connection =
DriverManager.getConnection(url);
            System.out.println("Connection Established...");
            Statement stmt = connection.createStatement();
            System.out.println("Platform Created...");
            ResultSet rs = stmt.executeQuery(query);
            System.out.println("Query Executed...");
            if(rs.last())
            {
                rs.beforeFirst();
                while(rs.next())
                {
                    System.out.println("*****");
                    System.out.println("Student ID :
"+rs.getInt(1));
                    System.out.println("Student Name :
"+rs.getString(2));
                    System.out.println("Student Stream :
"+rs.getString(3));
                    System.out.println("Student Marks :
"+rs.getInt(4));
                    System.out.println("*****");
                }
            }
        }
        catch (SQLException e)
        {

```

```

        e.printStackTrace();
    }

}
}

```

## **OUTPUT**

Connection Established...

Platform Created...

Query Executed...

\*\*\*\*\*

Student ID : 2

Student Name : ASHOK

Student Stream : ECE

Student Marks : 77

\*\*\*\*\*

\*\*\*\*\*

Student ID : 3

Student Name : PREM

Student Stream : EEE

Student Marks : 83

\*\*\*\*\*

\*\*\*\*\*

Student ID : 4

Student Name : SINGH

Student Stream : CE

Student Marks : 75

\*\*\*\*\*

\*\*\*\*\*

Student ID : 5

Student Name : ROSHAN

Student Stream : CSE

Student Marks : 80

\*\*\*\*\*

**Q. WRITE A JDBC PROGRAM TO PRINT USER INFORMATION IF MOBILE NUMBER ENDING WITH '5'.**

**Q. WRITE A JDBC PROGRAM TO PRINT USER NAME, MOBILE NUMBER, PASSWORD WHOSE NAME CONTAIN TWO 'A's**

**Q. WRITE A JDBC PROGRAM TO UPDATE STUDENT STREAM IF STUDENT MARKS>80 AND STUDENT ID IS 'EVEN'**

**DATE : 03-09-2022**

**Q. WRITE A JDBC PROGRAM TO PRINT EMPLOYEE DETAILS IF EMPLOYEE ID IS PRIME NUMBER**

```
package org.jsp.jdbc;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class EmpIDPrime {

    public static void main(String[] args) {

        String url =
"jdbc:mysql://localhost:3306?user=root&password=12345";
        String query = "select * from teja15.employee";
        try
        {
            Connection connection =
DriverManager.getConnection(url);
            System.out.println("Connection Established...!");
            Statement stmt = connection.createStatement();
            System.out.println("Platform Created...!");
            ResultSet rs = stmt.executeQuery(query);
            System.out.println("Query Executed...");
            while(rs.next())
            {
                int num = rs.getInt(1);
                boolean isPrime = true;
                for(int i=2;i<num;i++)
                {
                    if(num%i==0)
                    {
                        isPrime = false;
                        break;
                    }
                }
                if(isPrime)
                {

```

```

"+rs.getInt(1));
"+rs.getString(2));
"+rs.getDouble(3));
"+rs.getInt(4));
"+rs.getString(5));
        }
    }
    }
    catch (SQLException e)
    {
        e.printStackTrace();
    }
}

```

## **OUTPUT**

```

Connection Established...!
Platform Created...!
Query Executed...
Employee Id : 101
Employee Name : RAJA
Employee Salary : 100000.0
Employee DeptNo : 10
Employee Location : DELHI
*****

Employee Id : 103
Employee Name : RAM
Employee Salary : 50000.0
Employee DeptNo : 10
Employee Location : CHENNAI
*****

Employee Id : 107
Employee Name : TEJA
Employee Salary : 30000.0
Employee DeptNo : 30
Employee Location : KOLKATA
*****

```

**DATE : 05-09-2022**

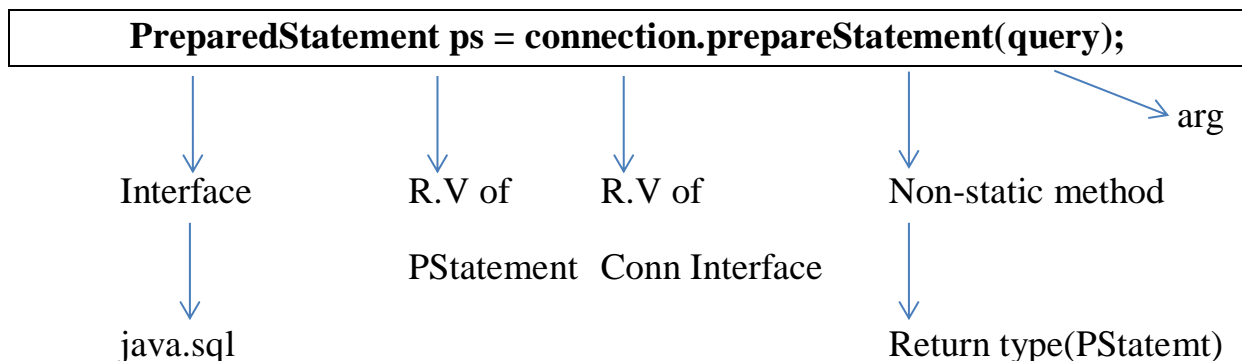
## **PREPARED STATEMENT**

insert into teja15.employee(101,'xyz',1000.0,10,'HYD') {Hardcode values}

insert into teja15.employee(?,?,?,?,:) {Placeholders → Runtime → Scanner class}

STATEMENT	PREPARED STATEMENT
We are using hard coded values	We are using run time values
Placeholders are not present in Statement platform	Placeholders are present in PreparedStatement platform
createStatement() is a no argument method	prepareStatement(query) is a argument method
Pass the query to the platform	Already query as passed
Return type is Statement platform	Return type is PreparedStatement platform

## **HOW TO CREATE PREPARED STATEMENT PLATFORM**



## **HOW TO ASSIGN VALUES TO THE PLACEHOLDERS**

setter → setInt(placeholder\_position,value)

→ setString(placeholder\_position,value)

→ setDouble(placeholder\_position,value)

→ setFloat(placeholder\_position,value)

**package** org.jdbc.prepared;

**import** java.sql.Connection;



```

import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.Scanner;

public class PStmtProgram {

    public static void main(String[] args) {

        String url =
"jdbc:mysql://localhost:3306?user=root&password=12345";
        String query = "insert into teja15.employee values(?,?,?,?) ";
        try
        {
            Connection connection =
DriverManager.getConnection(url);
            System.out.println("Connected...!");
            PreparedStatement ps =
connection.prepareStatement(query);
            Scanner sc = new Scanner(System.in);
            System.out.println("Enter the Employee ID : ");
            int id = sc.nextInt();
            ps.setInt(1,id);
            System.out.println("Enter the Employee Name : ");
            String name = sc.next();
            ps.setString(2, name);
            System.out.println("Enter the Employee Salary : ");
            double sal = sc.nextDouble();
            ps.setDouble(3,sal);
            System.out.println("Enter the Employee Dept : ");
            int dept = sc.nextInt();
            ps.setInt(4, dept);
            System.out.println("Enter the Employee Location : ");
            String loc = sc.next();
            ps.setString(5, loc);
            ps.executeUpdate();
            System.out.println("Data Inserted...!");

        }
        catch (SQLException e)
        {
            e.printStackTrace();
        }
    }
}

```

```
}  
}
```

## **OUTPUT**

Connected...!

Enter the Employee ID :

109

Enter the Employee Name :

SIMON

Enter the Employee Salary :

40000

Enter the Employee Dept :

10

Enter the Employee Location :

GURGAON

Data Inserted...!

**WRITE A JDBC PROGRAM TO UPDATE STUDENT GRADE AS 'A' IF THE MARKS BETWEEN 70-80**

**WRITE A JDBC PROGRAM TO DELETE THE CAR INFORMATION IF THE CAR PRICE IS MORE THAN 2CR**

**DATE : 06-09-2022**

**WRITYE A JDBC PROGRAM TO DELETE A USER RECORD BY THE MOBILE NUMBER**

```
package org.jdbc.prepared;
```

```
import java.sql.Connection;
```

```
import java.sql.DriverManager;
```

```
import java.sql.PreparedStatement;
```

```
import java.sql.SQLException;
```

```
import java.util.Scanner;
```

```
public class PStmtDelUMob {
```

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

```
        String url =
```

```
"jdbc:mysql://localhost:3306?user=root&password=12345";
```

```
        String query = "delete from teja15.user where uMobile=?";
```

```

        try
        {
            Connection connection =
DriverManager.getConnection(url);
            System.out.println("Connected...");
            PreparedStatement ps =
connection.prepareStatement(query);
            Scanner sc = new Scanner(System.in);
            System.out.println("Enter the Mobile Number to Delete : ");
            String mobile = sc.next();
            ps.setString(1, mobile);
            int num = ps.executeUpdate();
            if(num > 0)
            {
                System.out.println("Record deleted...");
            }
            else
            {
                System.err.println("Invalid Record...");
            }
        }
        catch (SQLException e)
        {
            e.printStackTrace();
        }
    }
}

```

## OUTPUT

```

Connected...
Enter the Mobile Number to Delete :
7345234955
Record deleted...

```

**DATE : 07-09-2022**

**WRITE A JDBC PROGRAM TO UPDATE SALARY WHO BORN AFTER 2005 TO UPDATE VALIDATE THEIR MOBILE NUMBER WITH OTP**

**package** org.jdbc.prepared;

```

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Random;
import java.util.Scanner;

public class PStmtUpdSalOTP {

    public static void main(String[] args) {

        String url =
"jdbc:mysql://localhost:3306?user=root&password=12345";
        String query = "select * from teja15.employee where empDOB>?";
        try
        {
            Connection connection =
DriverManager.getConnection(url);
            System.out.println("Connected");
            PreparedStatement ps =
connection.prepareStatement(query);
            Scanner sc = new Scanner(System.in);
            System.out.println("Enter the DOB : ");
            String dob = sc.next();
            ps.setString(1,dob);
            ResultSet rs = ps.executeQuery();
            if(rs.last())
            {
                rs.beforeFirst();
                while(rs.next())
                {
                    System.out.println("Emp Id : "+rs.getInt(1));
                    System.out.println("Emp Name :
"+rs.getString(2));
                    System.out.println("Emp Sal :
"+rs.getDouble(3));
                    System.out.println("Emp Dept : "+rs.getInt(4));
                    System.out.println("Emp Location :
"+rs.getString(5));
                    System.out.println("Emp DOB :
"+rs.getString(6));
                }
            }
        }
    }
}

```

```

"+rs.getString(7));

System.out.println("Emp Mobile :

System.out.println("*****");
System.out.println("Enter the Mobile Number :

");

String mobile = sc.next();
if(mobile.equals(rs.getString(7)))
{
    Random r = new Random();
    int otp = r.nextInt(10000);
    if(otp<1000)
    {
        otp+=1000;
    }
    System.out.println("OTP : "+otp);
    System.out.println("Enter the OTP for

verification : ");

    int user = sc.nextInt();
    if(user == otp)
    {
        query = "update teja15.employee
set empSal=empSal+empSal*0.1 where empMOB='"+mobile+"'and
empDOB> '"+dob+"'";

        PreparedStatement ps1 =
connection.prepareStatement(query);

        ps1.executeUpdate();
        System.out.println("Salary

Updated...!");
    }
    else
    {
        System.err.println("Invalid

OTP...!!!");
    }
}
else
{
    System.err.println("Invalid Mobile

Number...!!!");
}
}
}

```

```

        else
        {
            System.err.println("No Data Found");
        }
    }
    catch (SQLException e)
    {
        e.printStackTrace();
    }
}

```

## **OUTPUT**

Connected

Enter the DOB :

2004-12-31

Emp Id : 102

Emp Name : ASHOK

Emp Sal : 200000.0

Emp Dept : 20

Emp Location : BANGLORE

Emp DOB : 2006-05-31

Emp Mobile : 9876543210

\*\*\*\*\*

Enter the Mobile Number :

9876543210

OTP : 9688

Enter the OTP for verification :

9688

Salary Updated...!

Emp Id : 105

Emp Name : JAY

Emp Sal : 500000.0

Emp Dept : 10

Emp Location : COCHIN

Emp DOB : 2007-02-08

Emp Mobile : 8798634564

\*\*\*\*\*

Enter the Mobile Number :

8798634564

OTP : 4824

Enter the OTP for verification :

4824

Salary Updated...!  
Emp Id : 107  
Emp Name : TEJA  
Emp Sal : 30000.0  
Emp Dept : 30  
Emp Location : KOLKATA  
Emp DOB : 2007-12-28  
Emp Mobile : 6543277379  
\*\*\*\*\*  
Enter the Mobile Number :  
6543277379  
OTP : 1026  
Enter the OTP for verification :  
1026  
Salary Updated...!

## **WRITE A JDBC PROGRAM TO UPDATE EMPID,DEPT AND LOCATION BY PERFORMING TWO STEPS OF VERIFICATION**

- 1. By Mobile number**
- 2. By Name**

**DATE : 08-09-2022**

### **STORED PROCEDURES**

==> Stored procedures are used to store the queries in Database.

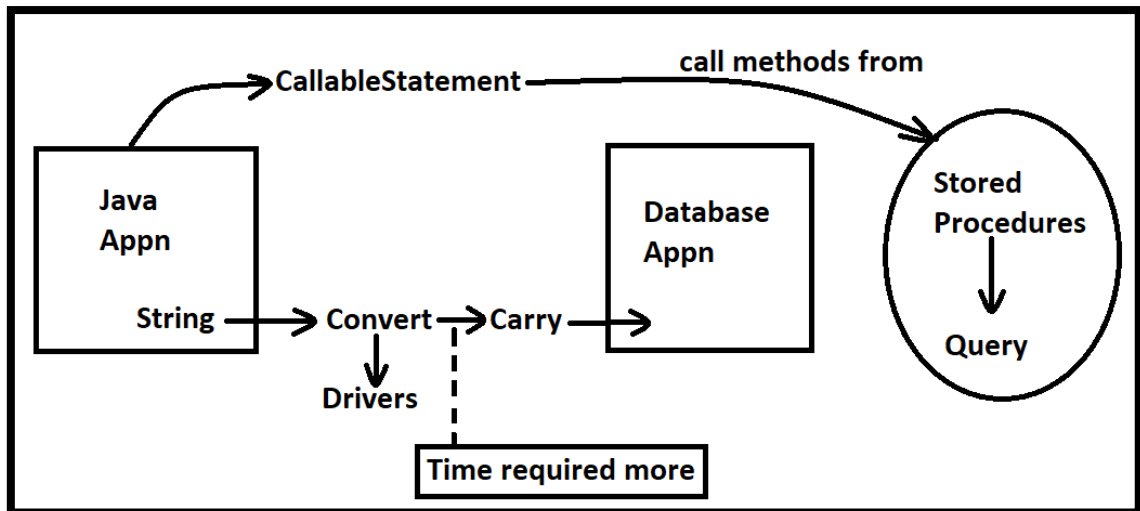
==> Stored procedures are nothing but methods.

==> The methods which is present in database to store the queries permanently is known as Stored procedures.

==> If a query is present in Java application in order to compile and execute it will take long process and more time.

==> If the query is present in the Java application it will affects to application performance.

==> If the query is present in the Database in order to compile and execute we will take short process and less time, So it will not affect to application performance.



---

### NOTE :

In order to call stored procedures we have to make use of platform called as CallableStatement

---

### HOW TO CREATE STORED PROCEDURES IN DATABASE

1. Select the Database and Right click on it.
2. Select an option called as Create Stored Procedure
3. Specify the Stored procedure name and Click on Create
4. Write the query in between **BEGIN** and **END**.

### SYNTAX:

```
CREATE PROCEDURE 'teja15'. 'EmpInsert' ( )  
  
BEGIN  
  
    insert into employee  
values(110,'UDAY',200000,30,'SRINAGAR','1997-08-22','7548351587');  
  
END $$
```

5. In order to compile stored procedures we have to make use of “Execute all queries” option.(>>)



## **CALLABLE STATEMENT**

==> It is an interface which is present inside **java.sql** package.

==> CallableStatement platform is used to call stored procedures.

==> To create a CallableStatement platform we make use of prepareCall() method.

### **Syntax of prepareCall()**

```
prepareCall("call database_name.Stored_procedure_name");
```

### **Syntax to create CallableStatement platform**

```
CallableStatement cs = connection.prepareCall("call  
database_name.Stored_procedure_name");
```

where

**CallableStatement** is an Interface which is present in java.sql package

**cs** is reference variable of CallableStatement

**connection** is reference variable of Connection interface

**prepareCall** is non-static method of Connection interface

**package** org.jsp.callable;

**import** java.sql.CallableStatement;

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.SQLException;

**public class** CallStoreProcedure {

**public static void** main(String[] args) {

**try**  
{

Connection connection =  
DriverManager.getConnection("jdbc:mysql://localhost:3306?user=root&password=12345");

```

        System.out.println("Connection Established");
        CallableStatement cs = connection.prepareCall("call
teja15.EmpInsert");
        cs.executeUpdate();
        System.out.println("Stored procedure called successfully");
        connection.close();
    }
    catch (SQLException e) {
        e.printStackTrace();
    }
}
}

```

### **OUTPUT**

Connection Established  
Stored procedure called successfully

### **Q. CREATE A STORED PROCEDURE TO DELETE A RECORD FROM EMPLOYEE TABLE BY USING EMPLOYEE ID**

```

CREATE DEFINER=`root` @`localhost` PROCEDURE `EmpDelete`()
BEGIN
delete from employee where empId=110;
END$$

```

```

package org.jsp.callable;

```

```

import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

```

```

public class DelEmpId {

```

```

    public static void main(String[] args) {

```

```

        String url =
"jdbc:mysql://localhost:3306?user=root&password=12345";
        try

```

```

        {
            Connection connection =
DriverManager.getConnection(url);
            System.out.println("Connection Established");
            CallableStatement cs = connection.prepareCall("call
teja15.EmpDelete");
            cs.executeUpdate();
            System.out.println("Stored procedure called successfully");
            System.out.println("Record Deleted Successfully");
        }
        catch (SQLException e)
        {
            e.printStackTrace();
        }
    }
}

```

### **OUTPUT**

Connection Established  
 Stored procedure called successfully  
 Record Deleted Successfully

**Q. CREATE A STORED PROCEDURE TO UPDATE USER NAME, ADDRESS AND MOBILE IF EMAIL AND PASSWORD IS PRESENT**

**Q. CREATE A STORED PROCEDURE TO DELETE A RECORD FROM USER INFORMATION TABLE WHO BORN IN 1998 AND ADDRESS IS 'HYDERABAD'**

**DATE : 10-09-2022**

### **STORED PROCEDURE WITH ARGUMENT**

```

CREATE PROCEDURE teja15.`empDeletion`(IN id int(5))
BEGIN
    delete from employee where empId = id;
END$$

```

```

package org.jsp.callable;

import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.Scanner;

public class EmpDeletion {

    public static void main(String[] args) {

        String url =
"jdbc:mysql://localhost:3306?user=root&password=12345";
        try
        {
            Connection connect = DriverManager.getConnection(url);
            System.out.println("Cnnection Established");
            CallableStatement cs = connect.prepareCall("call
teja15.empDeletion(?)");
            Scanner sc = new Scanner(System.in);
            System.out.println("Enter the Employee ID to delete the
Record : ");

            cs.setInt(1, sc.nextInt());
            cs.executeUpdate();
            System.out.println("Record deleted...!!!");
        }
        catch (SQLException e)
        {
            e.printStackTrace();
        }
    }
}

```

## OUTPUT

```

Cnnection Established
Enter the Employee ID to delete the Record :
108
Record deleted...!!!

```

**DATE : 12-09-2022**

**CREATE A STORED PROCEDURE TO UPDATE EMAIL IF MOBILE NUMBER AND PASSWORD PRESENT IN DATABASE AND PERFORM THE VERIFICATION BY DATE OF BIRTH**

```
package org.jsp.callable;
```

```
import java.sql.CallableStatement;
```

```
import java.sql.Connection;
```

```
import java.sql.DriverManager;
```

```
import java.sql.PreparedStatement;
```

```
import java.sql.ResultSet;
```

```
import java.sql.SQLException;
```

```
import java.util.Scanner;
```

```
public class MailUpdation {
```

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

String url =

```
"jdbc:mysql://localhost:3306?user=root&password=12345";
```

try

 $\{$ 

```
Connection connect = DriverManager.getConnection(url);
```

```
String query = "select * from teja15.user where uMobile=?";
```

and uPwd =?";

```
PreparedStatement ps = connect.prepareStatement(query);
```

Scanner **sc** = **new** Scanner(System.*in*);

```
System.out.println("Enter the Mobile Number : ");
```

```
ps.setString(1,sc.next());
```

```
System.out.println("Enter the Password : ");
```

```
ps.setString(2,sc.next());
```

```
ResultSet rs = ps.executeQuery();
```

```
if(rs.next())
```

 $\{$ 

```
System.out.println("Enter DOB for 2nd Verification : ");
```

") ;

```
String dob = sc.next();
```

```
if(dob.equals(rs.getString(6)))
```

 $\{$ 

```
System.out.println("Valid DOB...!");
```

```

        CallableStatement cs =
connect.prepareCall("call teja15.updMail(?,?)");
        System.out.println("Enter the New Email : ");
        cs.setString(1, sc.next());
        cs.setString(2, rs.getString(3));
        cs.executeUpdate();
        System.out.println("Mail Updated");
    }
    else
    {
        System.err.println("Invalid DOB...!!!");
    }
}
else
{
    System.err.println("Invalid Mobile OR
Password...!!!");
}
}
catch (SQLException e)
{
    e.printStackTrace();
}
}
}

```

## **OUTPUT**

Enter the Mobile Number :

8775534955

Enter the Password :

raja666

Enter DOB for 2nd Verification :

1999-03-20

Valid DOB...!

Enter the New Email :

raj@gmail.com

Mail Updated

**DATE : 13-09-2022**

**CREATE A STORED PROCEDURE TO UPDATE STUDENT STREAM AND MARKS BY STUDENT ID AND NAME CONSIDER ONLY EVEN DIGIT STUDENT ID**

```
package org.jsp.callable;
```

```
import java.sql.CallableStatement;
```

```
import java.sql.Connection;
```

```
import java.sql.DriverManager;
```

```
import java.sql.SQLException;
```

```
import java.util.Scanner;
```

```
public class UpdSTD {
```

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

```
        String url =
```

```
"jdbc:mysql://localhost:3306?user=root&password=12345";
```

```
        try
```

```
        {
```

```
            Connection connect = DriverManager.getConnection(url);
```

```
            CallableStatement cs = connect.prepareCall("call
```

```
teja15.updSTD(?,?,?,?)");
```

```
            Scanner sc = new Scanner(System.in);
```

```
            System.out.println("Enter the New Stream : ");
```

```
            cs.setString(1,sc.next());
```

```
            System.out.println("Enter the New Marks : ");
```

```
            cs.setDouble(2,sc.nextDouble());
```

```
            System.out.println("Enter Student Name : ");
```

```
            cs.setString(3,sc.next());
```

```
            System.out.println("Enter the Student ID : ");
```

```
            int temp = sc.nextInt();
```

```
            if(temp%2==0)
```

```
            {
```

```
                cs.setInt(4,temp);
```

```
                int num = cs.executeUpdate();
```

```
                if(num>0)
```

```
                {
```

```
                    System.out.println("Record Updated...!");
```

```
                }
```

```
            else
```

```

        {
            System.err.println("INVALID STUDENT
NAME OR ID...!!!");
        }
    }
    else
    {
        System.err.println("STUDENT ID IS ODD...!!!");
    }
}
catch (SQLException e)
{
    e.printStackTrace();
}
}
}

```

## **OUTPUT**

Enter the New Stream :

CIVIL

Enter the New Marks :

78

Enter Student Name :

SINGH

Enter the Student ID :

4

Record Updated...!

**WRITE A JDBC PROGRAM TO PRINT THE COUNT OF THE STUDENTS WHOSE MARKS IS GREATER THAN 60 AND LESS THAN 60**

```
package org.jsp.callable;
```

```
import java.sql.Connection;
```

```
import java.sql.DriverManager;
```

```
import java.sql.ResultSet;
```

```
import java.sql.SQLException;
```

```
import java.sql.Statement;
```

```
public class STDCount {
```



```

public static void main(String[] args) {

    String url =
"jdbc:mysql://localhost:3306?user=root&password=12345";
    String query = "select * from teja15.student";
    try
    {
        Connection connect = DriverManager.getConnection(url);
        Statement stmt = connect.createStatement();
        ResultSet rs = stmt.executeQuery(query);
        int gcount = 0;
        int lcount = 0;
        if(rs.last())
        {
            rs.beforeFirst();
            while(rs.next())
            {
                if(rs.getDouble(4)>60)
                {
                    gcount++;
                }
                else
                {
                    lcount++;
                }
            }
            System.out.println(gcount+" Students have greater
than 60 marks");
            System.out.println(lcount+" Students have less than
60 marks");
        }
        else
        {
            System.err.println("NO RECORDS PRESENT IN
TABLE");
        }
    }
    catch (SQLException e)
    {
        e.printStackTrace();
    }
}
}

```

## OUTPUT

5 Students have greater than 60 marks

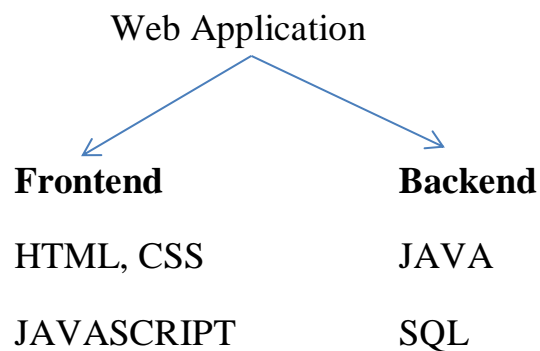
2 Students have less than 60 marks

**DATE : 14-09-2022**

## SERVLETS

==> Servlet is not a server, it is a Class.

==> By using Servlets we are going to develop Web applications.



==> Servlet is used to connect Frontend with Backend.

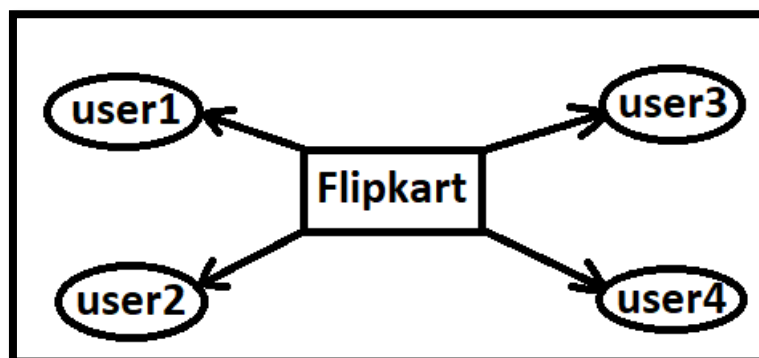
==> Servlet will provide few additional features for an application

1. Multi-Threaded

2. Session

### 1. MULTI THREADED

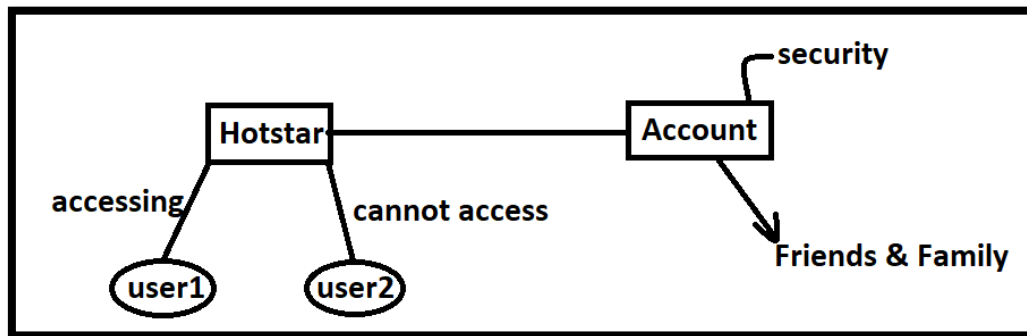
==> If an application which can be used by multiple users at the same time is known as Multithreaded.



## SINGLE THREADED

==> If an application can access by only user where other user have to wait to access an application is known as Single Threaded.

==> Single threaded is opposite behaviour of Multithreaded.

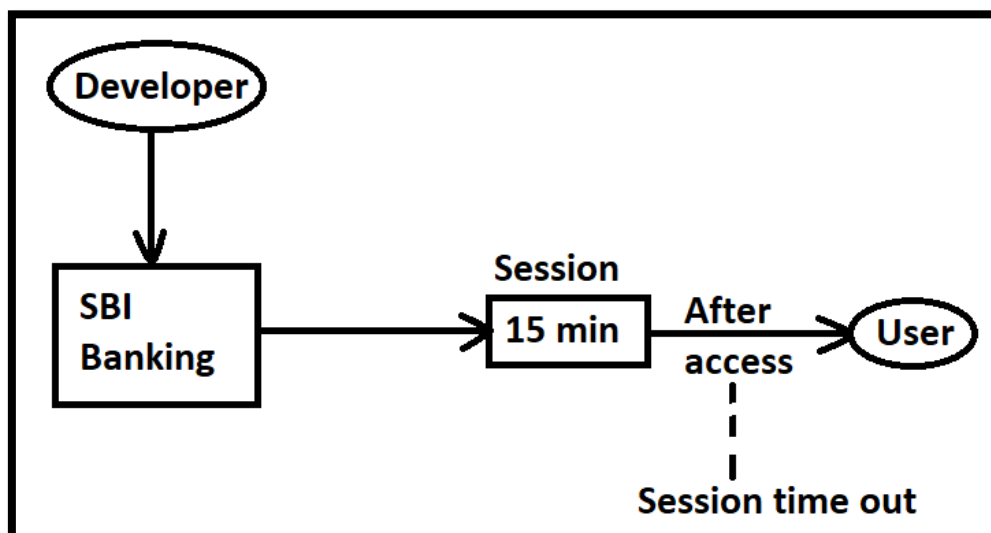


## 2. SESSION

==> Session is nothing but time interval which is given to the application by the Developer.

==> The applications which are developed by using Session feature the operation should be complete within given particular time.

==> If we tried to access an application after the time interval we will get "Session Time Out".



## **DEPLOYMENT**

==> Storing an application to the server technically referred as Deployment.

==> Deployment can be done in two ways

1. Manual Deployment
2. Automated Deployment

### **1. MANUAL DEPLOYMENT**

==> In Manual Deployment programmers are responsible for writing the code and giving the application to the server.

### **2. AUTOMATED DEPLOYMENT**

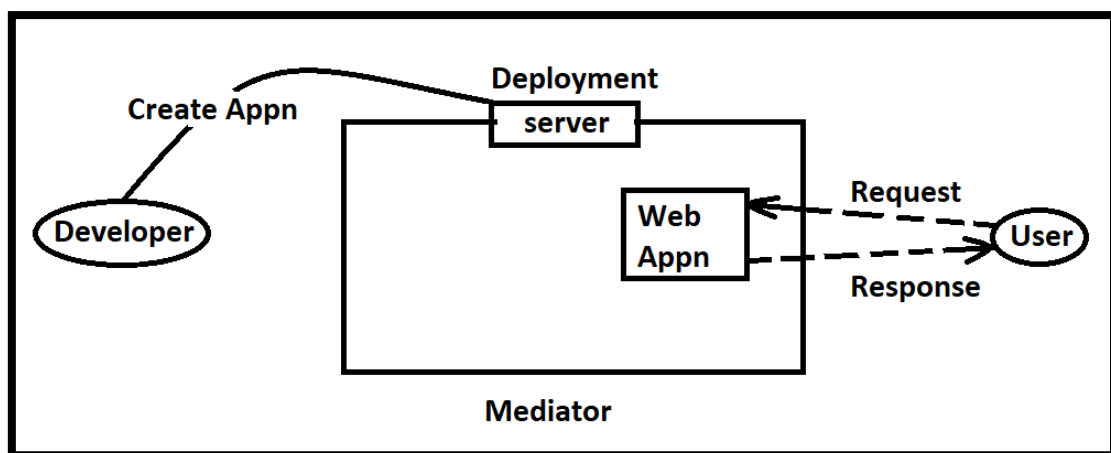
==> In Automated Deployment programmers are responsible to write the code and but the application will be given to the server by the Maven Tool.

---

### **NOTE:**

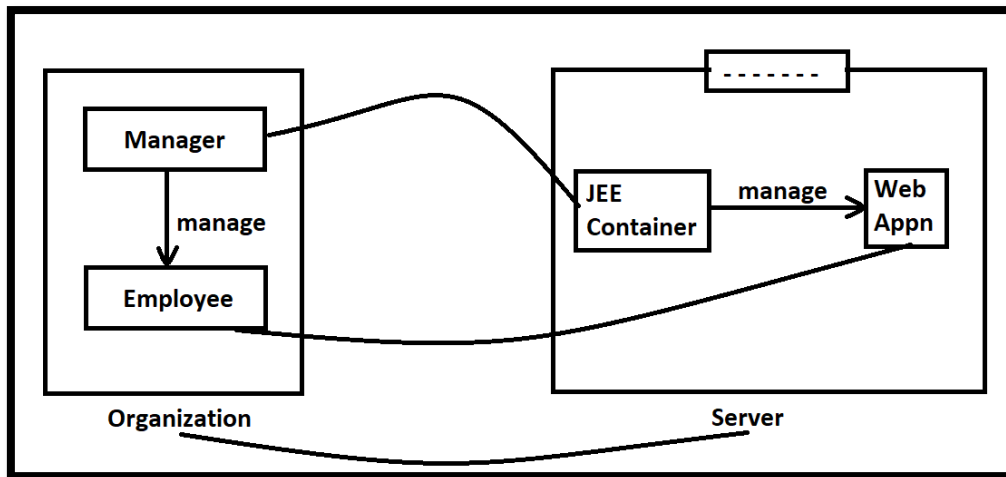
Developing a Web application means writing a code for Frontend and Backend

---



## **JEE CONTAINER**

==> JEE Container is an important component which is present inside Server to maintain Web applications.

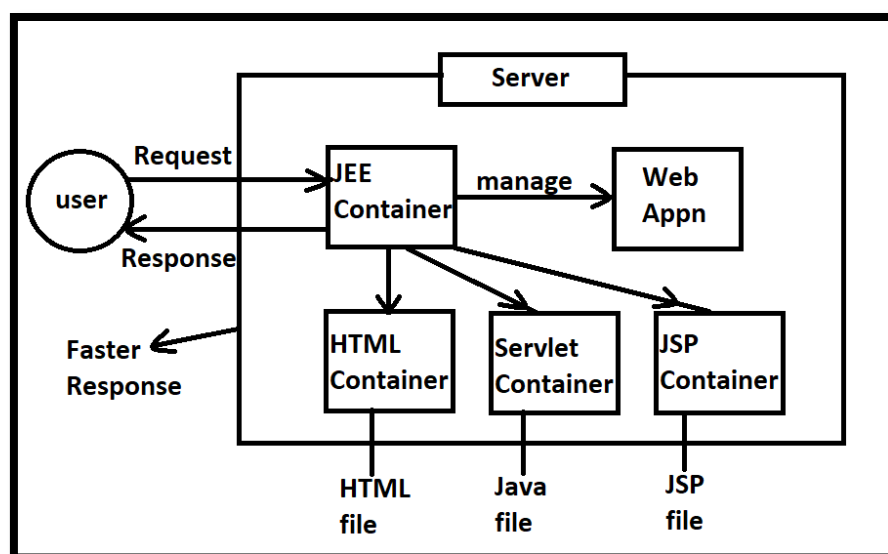


==> To maintain the Web application JEE container is going to follow 5 functionalities.

1. Segregation of files
2. Process of an user request
3. Manage Servlet life cycle
4. Create config & context object
5. Managing Servlet chaining

**DATE : 15-09-2022**

### SEGREGATION OF FILES



==> Once after request sent to JEE container the functionalities performed by JEE container are

1. Once after Deployment, JEE container will segregate the file of an application into separate container based on the execution of the files which results in faster response.
2. Once after segregation of file user can make a request and JEE container provides response. This process is technically referred as Process of an User request.
3. JEE container manages Servlet Life cycle
4. JEE container is responsible for creating config and context object
5. JEE container will handle Servlet chaining

## **TYPES OF PROJECTS**

1. Java Project —————> Java Perspective{.java} —————> Backend

2. Dynamic Web Project



JavaEE Perspective{.html, .java , .jsp} —————> Frontend + Backend

==> In Eclipse there are multiple types of applications can be created based on requirement.

1. Java project
2. Dynamic Web project

==> Programmers prefer Java project if requirement consists of only based on Backend file.

==> Programmers prefer Dynamic Web project if requirement consists of both Frontend and Backend file.

==> Dynamic Web projects consists of 4 important folders

1. Build folder
2. Src folder

3. Web content folder

4. Lib folder

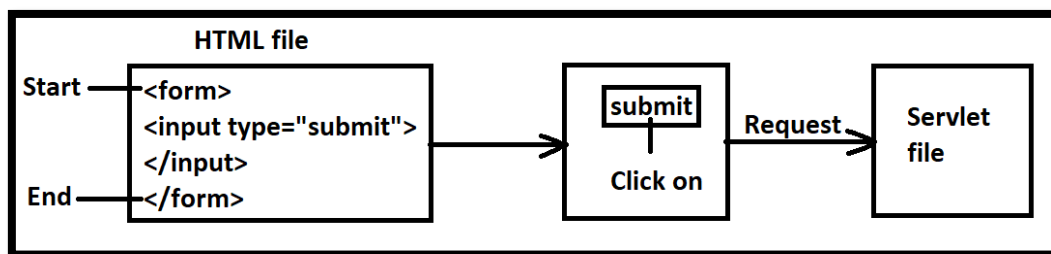
==> Source folder or Java folder is use of Backend file whereas Web content or Web app folder is used for Frontend file.

==> Lib folder consists of JAR files which are required for execution.

## **TYPES OF CLASSES**

1. Normal class ———> execution ———> main( ) method

2. Servlet class ———> execution ———> HTML file



==> We have many types of classes which can be created inside project.

1. Normal class

2. Servlet class

==> If an execution start from main( ) method is known as Normal class.

==> If an execution start from HTML file is known as Servlet class.

==> All the Servlet classes will get execute once after request sent from HTML file.

==> The basic way of making request from HTML file is by creating a button.

## **SYNTAX:**

**`<input type="submit"> </input>`**

## **HOW TO CREATE DYNAMIC WEB PROJECT**

==> Open Eclipse ———> Press Ctrl+N

==> Search Dynamic Web project.

==> Select the project which is under Web —> Dynamic Web project

==> Mention the project name, Click on Next

==> Create Java file to store Backend file by removing **src** which is already present.

==> Create a folder src/main/Java —> Click on Next

==> Create Web app folder to store Frontend file —> src/main/webapp

==> Click on Finish.

**DATE : 16-09-2022**

### **TYPES OF SERVLET CLASSES**

==> We have two types of Servlet classes

1. Generic Servlet class
2. HTTP Servlet class

==> Generic Servlet class will support only Multithreaded, it will not support Session.

==> In HTTP Servlet class they can use Multithreaded as well as Session feature.

---

### **NOTE:**

In Servlet class request will send from HTML file by using Submit button.

---

### **HOW TO RUN HTML FILE**

==> After HTML code click on Run button.

==> Select Apache Tomcat version

==> Click on Next



==> Select the version which is present inside  
ThisPC/ProgramFiles/ApacheSoftwareVersion/Select the version.

==> Click on Next and Select the Web Dynamic project.

==> Click on Finish

## **HOW TO SOLVE “SERVER IS CURRENTLY IN USE” ERROR**

### **METHOD-1**

==> Go to search —> Search as Task Manager

==> Select services —> Find the Tomcat version which is installed in the  
system —> Check the status of Server —> If it is already in run(running)  
—> Select the server and Right click on it —> Select Stop

==> Close the Window.

### **METHOD-2**

==> Press Windows+R button.

==> Search as services.msc —> Find the Tomcat version which is installed in  
the system —> Check the status of Server —> If it is already in run  
(running) —> Select the server and Right click on it —> Select Stop

==> Close the Window.

## **GENERIC SERVLET CLASS**

==> Generic Servlet class is an Abstract class which is present in **javax.servlet**  
/ jakarta.servlet package.

==> To print an output or to perform Backend operations we have to make use  
of service( ) method which is present inside Generic Servlet class.\

==> service( ) method is an abstract method which is present with two  
arguments

arg0→ ServletRequest

arg1→ ServletResponse

==> ServletRequest and ServletResponse are the interfaces which are present in  
javax.servlet

==> ServletRequest interface will provide User Request to the Server.

==> ServletResponse interface will provide Server response to the User.

### **SYNTAX FOR SERVICE() METHOD**

```
public(+) void service(ServletRequest arg0, ServletResponse arg1)
{
    //Backend code
}
```

### **//Example Program**

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>First Program</title>
</head>
<body style="background-color:springgreen">
    <center>
        <form action = "FirstGeneric">
            <br><br><br><br>
            <input type="submit"></input>
        </form>
    </center>
</body>
</html>
```

```
package org.jsp.servletPrograms;
```

```
import java.io.IOException;
```

```
import javax.servlet.GenericServlet;
```

```
import javax.servlet.ServletException;
```

```
import javax.servlet.ServletRequest;
```

```
import javax.servlet.ServletResponse;
```

```
import javax.servlet.annotation.WebServlet;
```

```
@WebServlet("/FirstGeneric")
```

```
public class FirstGeneric extends GenericServlet {
```

```
    @Override
```

```
public void service(ServletRequest req, ServletResponse res) throws  
ServletException, IOException {  
    System.out.println("Backend Code");
```

```
    }  
}
```

## OUTPUT

Backend Code

**DATE : 19-09-2022**

## TO LINK FRONTEND WITH BACKEND

==> We need to provide some important information

1. In HTML file we have to provide Servlet class name for annotation

For Example,

```
<form action="FirstServletProgram"></form>
```

2. We have to provide annotation for Servlet class

3. Annotations are always present above the class name

4. Annotations are declared by PASCAL casing.

## WHAT IS ANNOTATION?

==> Annotations are used to provide more information about class, interface, variables and methods.

==> Annotations are used to give the information for JVM, JEE Container (or) Compiler.

==> Required annotation is **@WebServlet( )**

==> Not Required annotation is **@override( )**

## getParameter( ) METHOD

==> It is a non-static method which is present inside ServletRequest interface.

==> It is used to get the data from the webpage which is entered by the user.

==> getParameter( ) method is an argument method which will accept String values.

==> We have to pass name attribute value as an argument to the getParameter( ) method.

Example:

```
<input type="text" name="uname"></input>
```

```
String name = req.getParamter(uname);
```

==> the return type of getParameter( ) method is String.

### //Example Program

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>To get data from Web Page</title>
</head>
<body style="background-color:cyan">
    <center>
        <form action="WebPageData">
            <label>Enter User Name :</label>
            <input type="text" name="uname"></input><br><br>
            <label>Enter User Password :</label>
            <input type="text" name="upwd"></input><br><br>
            <input type="submit"></input>
        </form>
    </center>
</body>
</html>
```

```
package org.jsp.servletPrograms;
```

```
import java.io.IOException;
```

```
import javax.servlet.GenericServlet;
```

```
import javax.servlet.ServletException;
```

```
import javax.servlet.ServletRequest;
```

```
import javax.servlet.ServletResponse;
```

```
import javax.servlet.annotation.WebServlet;
```

```

@WebServlet("/WebPageData")
public class WebPageData extends GenericServlet
{
    @Override
    public void service(ServletRequest req, ServletResponse res) throws
ServletException, IOException {
        System.out.println("Backend Code");
        String name = req.getParameter("uname");
        String pwd = req.getParameter("upwd");
        System.out.println("User Name : "+name);
        System.out.println("User Password : "+pwd);
    }
}

```

### OUTPUT

Backend Code  
User Name :Dinga  
User Password : 12345

**DATE : 20-09-2022**

### **WAP TO CREATE REGISTRATION FORM AND PRINT THE DATA IN THE BACKEND**

```

<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Registration Page</title>
</head>
<body style="background-color:green">
    <center>
        <form action="RegPage">
            <label> First Name :</label>
            <input type="text" name="fname"></input><br><br>
            <label> Last Name :</label>
            <input type="text" name="lname"></input><br><br>
            <label> DOB :</label>
            <input type="date" name="date"></input><br><br>
            <label> Email :</label>
            <input type="email" name="email"></input><br><br>
            <label> Mobile :</label>
            <input type="text" name="mob"></input><br><br>
        </form>
    </center>

```

```

        <label> Password :</label>
        <input type="text" name="pwd"></input><br><br>
        <input type="submit" value="Submit"></input>
        <input type="reset" value="Cancel"></input>
    </form>
</center>
</body>
</html>

```

```

package org.jsp.servletPrograms;

```

```

import java.io.IOException;

```

```

import javax.servlet.GenericServlet;

```

```

import javax.servlet.ServletException;

```

```

import javax.servlet.ServletRequest;

```

```

import javax.servlet.ServletResponse;

```

```

import javax.servlet.annotation.WebServlet;

```

```

@WebServlet("/RegPage")

```

```

public class RegPage extends GenericServlet
{

```

```

    @Override

```

```

    public void service(ServletRequest req, ServletResponse res) throws
ServletException, IOException

```

```

    {
        System.out.println("Backend Code");
        String first = req.getParameter("fname");
        String last = req.getParameter("lname");
        String dob = req.getParameter("date");
        String mail = req.getParameter("email");
        String mobile = req.getParameter("mob");
        String pass = req.getParameter("pwd");
        System.out.println("First Name : "+first);
        System.out.println("Last Name : "+last);
        System.out.println("Date of Birth : "+dob);
        System.out.println("Mail-Id : "+mail);
        System.out.println("Mobile : "+mobile);
        System.out.println("Password : "+pass);
    }

```

```

}

```

**OUTPUT**

Backend Code

First Name : Ram

Last Name : Kiran

Date of Birth : 1993-12-20

Mail-Id : ramkiran@gmail.com

Mobile : 9878787654

Password : rkiran@123

**// To print the data in database table**

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Registration Page</title>
</head>
<body style="background-color:green">
    <center>
        <h1 style="color:cyan"><marquee
direction="left"><u>REGISTRATION FORM</u></marquee></h1>
        <form action="RegPage">
            <label> First Name :</label>
            <input type="text" name="fname"></input><br><br>
            <label> Last Name :</label>
            <input type="text" name="lname"></input><br><br>
            <label> DOB :</label>
            <input type="date" name="date"></input><br><br>
            <label> Email :</label>
            <input type="email" name="email"></input><br><br>
            <label> Mobile :</label>
            <input type="text" name="mob"></input><br><br>
            <label> Password :</label>
            <input type="text" name="pwd"></input><br><br>
            <input type="submit" value="Submit"></input>
            <input type="reset" value="Cancel"></input>
        </form>
    </center>
</body>
</html>
```

**package** org.jsp.servletPrograms;

```

import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;

import javax.servlet.GenericServlet;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.annotation.WebServlet;
@WebServlet("/RegPage")
public class RegPage extends GenericServlet
{

    @Override
    public void service(ServletRequest req, ServletResponse res) throws
ServletException, IOException
    {
        System.out.println("Backend Code");
        System.out.println("*****");
        System.out.println("First Name : "+req.getParameter("fname"));
        System.out.println("Last Name : "+req.getParameter("lname"));
        System.out.println("Date of Birth : "+req.getParameter("date"));
        System.out.println("Mail-Id : "+req.getParameter("email"));
        System.out.println("Mobile : "+req.getParameter("mob"));
        System.out.println("Password : "+req.getParameter("pwd"));
        String first = req.getParameter("fname");
        String last = req.getParameter("lname");
        String dob = req.getParameter("date");
        String mail = req.getParameter("email");
        String mobile = req.getParameter("mob");
        String pass = req.getParameter("pwd");

        String url =
"jdbc:mysql://localhost:3306?user=root&password=12345";
        String query = "insert into teja15.userdetails values(?,?,?,?);";

        try
        {
            Class.forName("com.mysql.jdbc.Driver");
            Connection connect = DriverManager.getConnection(url);
            PreparedStatement ps = connect.prepareStatement(query);

```



```

        ps.setString(1,first);
        ps.setString(2,last);
        ps.setString(3,dob);
        ps.setString(4,mail);
        ps.setString(5,mobile);
        ps.setString(6,pass);
        ps.executeUpdate();
        System.out.println("REGISTRATION SUCCESSFUL...!");
        connect.close();
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
}

```

## **OUTPUT**

### Backend Code

\*\*\*\*\*

First Name : Ram

Last Name : Kiran

Date of Birth : 1995-03-20

Mail-Id : ramkiran@gmail.com

Mobile : 9878787654

Password : rkiran@123

REGISTRATION SUCCESSFUL...!

### **IMPORTANT QUESTIONS**

1. WRITE A SYNTAX FOR INSERT, UPDATE ,DELETE ,SELECT QUERY.
2. EXPLAIN ABOUT JAR FILE AND STEPS TO CREATE JAR FILE.
3. EXPLAIN ABOUT TYPES OF PERSPECTIVE AND API.
4. EXPLAIN ABOUT TYPES OF APPLICATIONS.
5. WRITE STEPS TO PERFORM JAVA BUILD PATH.
6. EXPLAIN ABOUT JDBC ARCHITECTURE AND DRIVERS.