J2EE - Java 2 Enterprise Edition, where 2 refers to version. (also called as Advanced Java).

J2EE

JDBC Servlet

J2EE = JDBC + Servlet

Q- Why J2EE is needed?

1. To develop web application.

By using core java we can only develop standalone application.

Two types of application

1. Standalone Application – which do not require internet and restricted to one system. For ex- Notepad, MS-Office.
2. Web Application – which require internet. For ex- face-book, amazon etc

**jar file**

* Any file which has .jar extension is jar file
* .jar is an extension which is based on zip file format.
* jar stands for java archive.

Contents inside jar file

* .java files
* .class file
* config file

Need for jar file

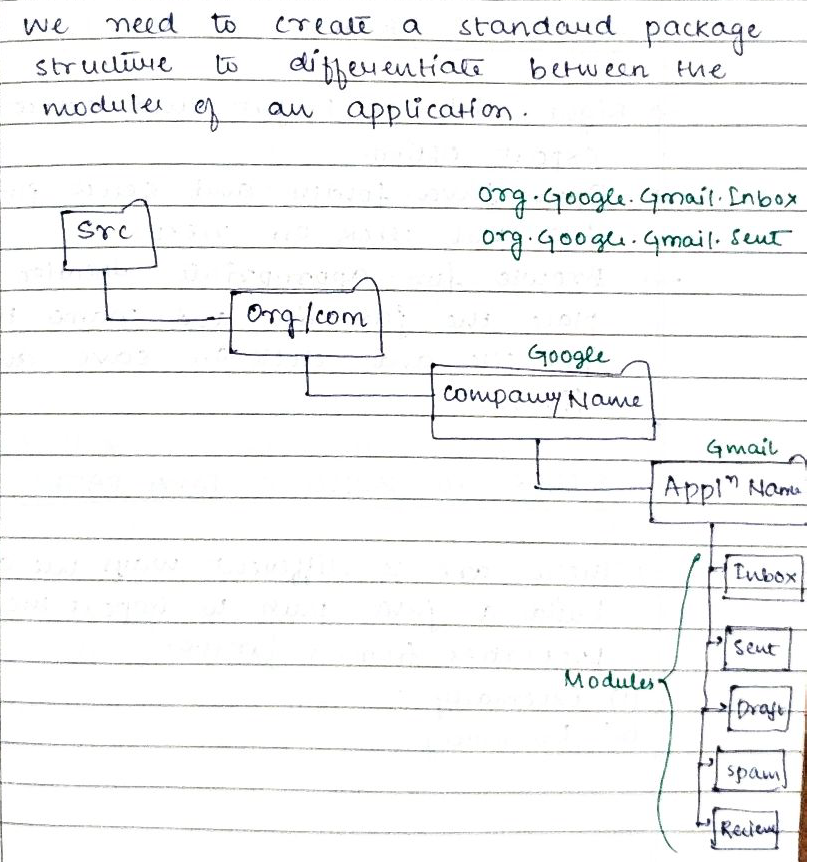
* jar files are basically used for distribution purpose, so that a developer can import these jar files into their program according to the requirement.

**STANDARD PACKAGE STRUCTURE**

What is Standard Package Structure ?

* It means way of maintaining your files and folders in a project.
* Root directory (main folder) 🡪 Package Name(sub folders)🡪 java files
* For ex🡪 com.jsp.JDBCApp

Why we need a standard package structure?



Steps to create a basic JDBC Project

Steps to build a Java Path

Internal Way

* Create a lib folder (inside your project) by right clicking on Project, Add the respective jar file inside that lib folder.
* Right click on project and properties option🡪java build path and then go to libraries tab.
* Click on add jar, select the jar file from the respective project and click on apply, apply close.

**Abstraction**

(what)

Hiding the implementation and showing the functionality with the help of interface is called abstraction.

(why)

We hide the implementation to reduce the complexity.(because user is least bothered about the implementation, he just wants to use the functionality without any complication).

(how)

In java we achieve Abstraction through interface and its implementation class

Q- What is **interface**?

* Interface is a java type definition, which is used to provide standardization and abstraction.
* In general, Interface is a medium between the user and the any device.
* It is a coding contract between the service and user.

**Loose coupling**- change in the implementation which do not affect the user is called loose coupling.

**Tight coupling**- change in the implementation which affects the user is called tight coupling.

**PORT NUMBER**

Port number helps us to get connected to a particular server.

Oracle Server 1521

MySql Server 3306

**API**

**-** API stands for Application programming interface.

- API is a medium by which two application can communicate with each other. Communicate means they can exchange the data and integrate the functionality of some other application into our own application without writing the code from scratch and without understanding the underlying implementation.

For ex- Google Map Api is used by Zomato App.

* In java APIs are given in the form jar file.
* Contents of API – interface, implementation classes and helper classes.

Through JDBC API which is given sun-microsystem we achieve loose coupling between the java application and database server.

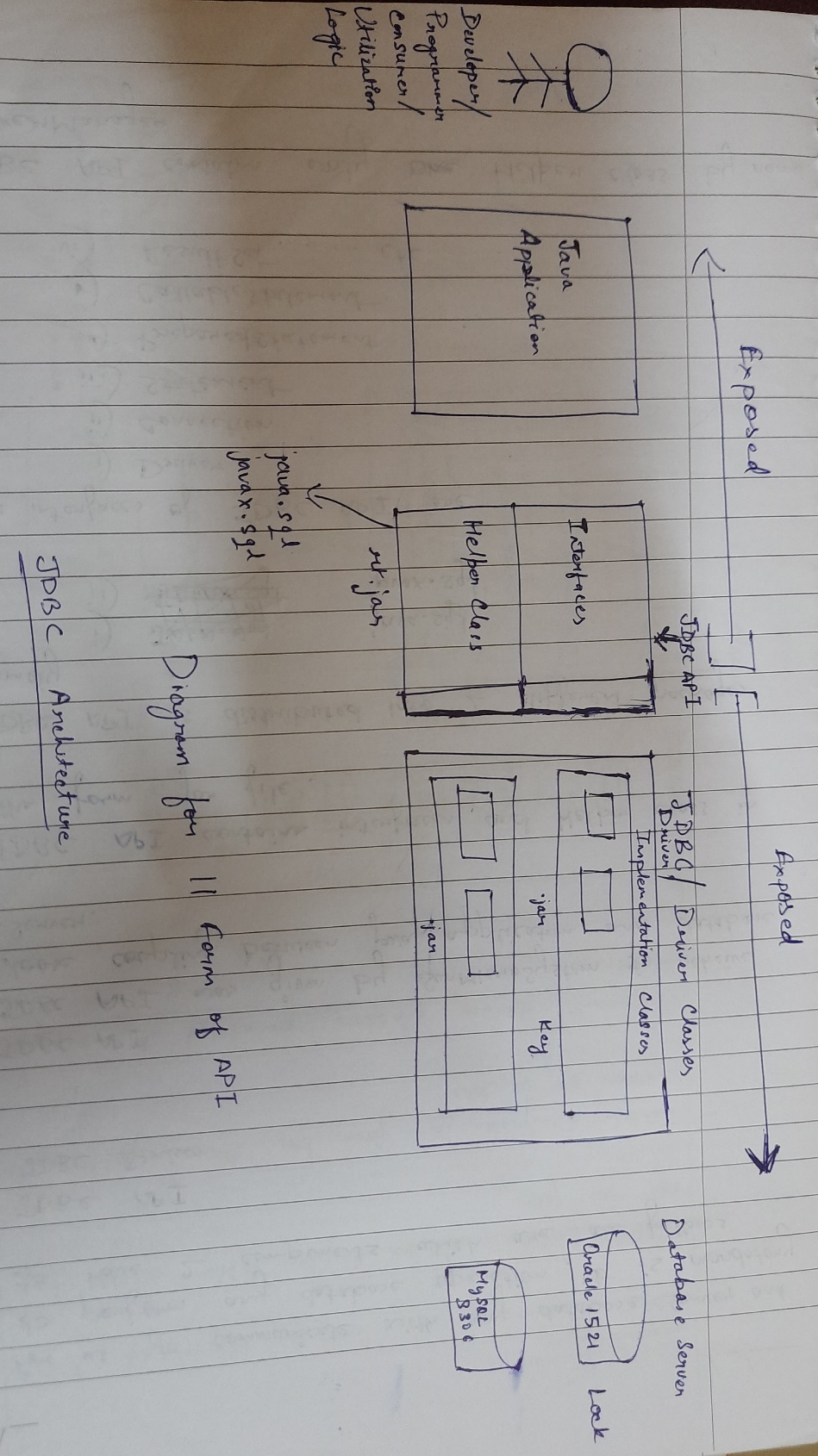
Types of API

Read from other source

I FORM OF API – It contains interfaces, implementation classes and helper class all in one single jar file.

For ex- Apache POI , JExcel.

II FORM API (JDBC ARCHITECTURE)



We need two things if we want our java application to communicate with respective database server

1. JDBC API – given by java(Sun Micro System)
2. JDBC Driver – given by respective database server

**1. JDBC API**

- given by java(SunMicroSystem) to achieve loose coupling

- JDBC API has only two things

1. interfaces (Driver, Connection, Statement, PreparedStatement,

CallableStatement, ResultSet)

2. Helper Class, only one helper class called DriverManager.

- JDBC API is has 2 different packages

1 java.sql – This is the main package which contains all the interfaces and one

Helper class.

(Driver, Connection, Statement, PreparedStatement,

CallableStatement, ResultSet, DriverManager).

2 javax.sql – This is the extended package which provide more

interface.

1. **JDBC DRIVER**

* JDBC Driver is a jar file which contains implementation classes of the interfaces present in JDBC API.
* JDBC Driver is provided by respective database server in the form jar file.

Names of the Driver classes given by respective database server

Database Name JDBC Driver Name Port Number

Oracle database Oracle Driver 1521

MySql Driver 3306

Fully qualified class name

com.mysql.jdbc.Driver -- provided by mysql

What is **JDBC**?

Java database connectivity is a specification (detailed description) which is given in the form of Abstraction API to achieve loose coupling between Java Application and Database Server.

**Advantage of JDBC**

* Achieve loose coupling between java application and database server. Which means without any major code changes we can switch the database.
* Platform independent

**SPECIFICATION OF JDBC**

Three different specification

* All the driver class must contain one static block in it.
* All the Driver classes must mandatorily implements java.sql.Driver interface.
* All the Driver Classes must mandatorily be registered with the DriverManager by using a static method called registerDriver().

**Class Loading**

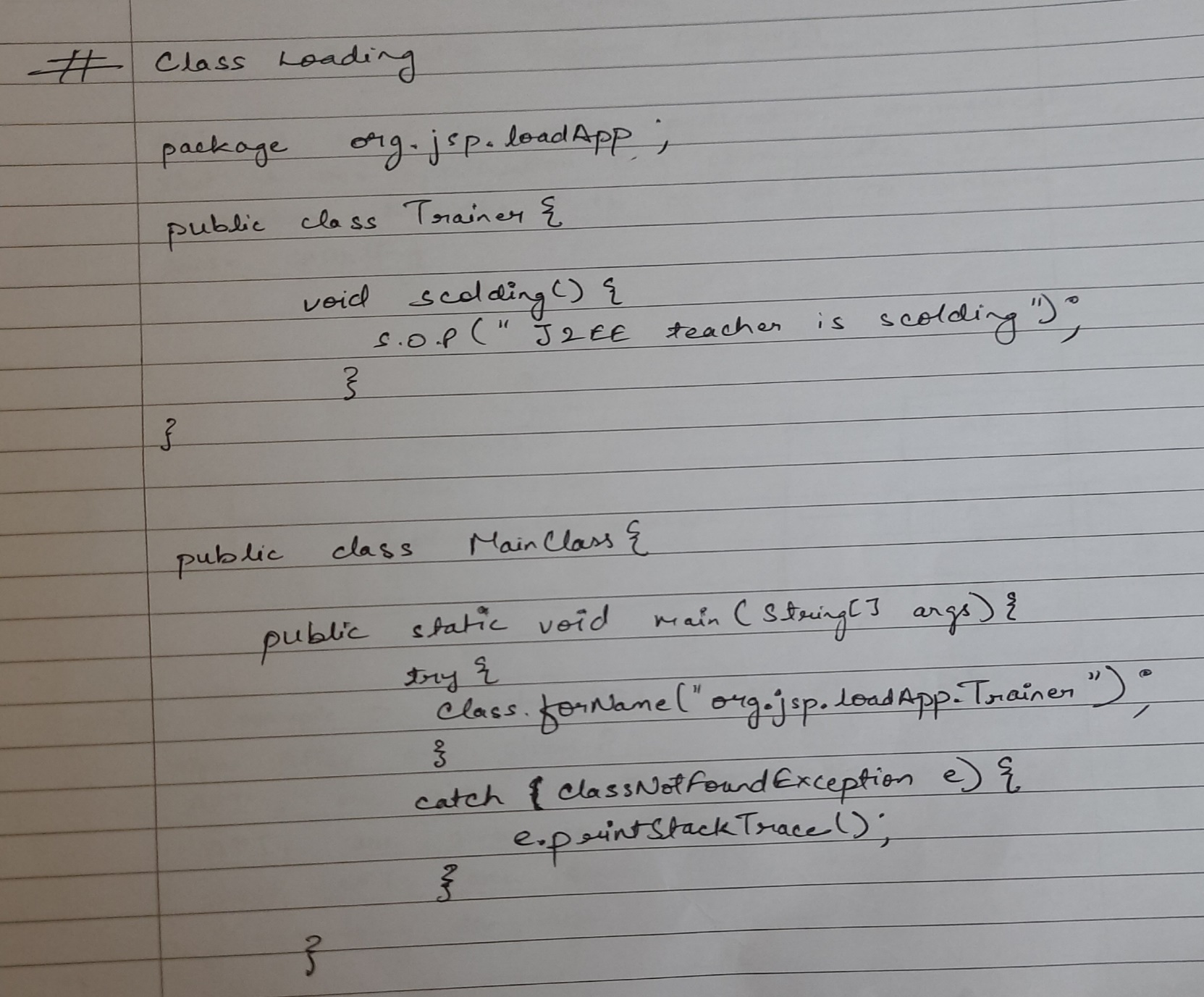
Loading a class into a JVM memory is called Class Loading.

There are 2 ways in which we can load the class into JVM memory

1. By calling any members of that class (variable, constructor, function, block etc)
2. To load the class into a JVM memory we have a static method called forName() which is present in Class name as Class(this class is present java.lang package) which will load the class.

Syntax- Class.forName(“fully qualified class name”) ;

1. Whenever we use this forName() method, it will throw a checked exception called ClassNotFoundException.



**Standards Steps of JDBC**

6 steps

1. Load & register the class
2. Establish the connection between java application and database server
3. Create a platform to execute the query
4. Execute the sql queries
5. Process the resultant data(optional)
6. Close all the costly resource
7. **Load & register the Driver class**

In this step we have to load the Driver class which is provided by respective

database server.

There are two ways in which we can load and register this Driver class

First way

* By creating an object of Driver Class (this way class will be load into JVM memory) and then for registering the Driver Class, we have to use a static method called registerDriver() method, present in DriverManager Class(This class is a helper class present in JDBC API).

Syntax-

Driver d = new driver(); //load the class into JVM memory

DriverManager.registerDriver(d) // registering the Driver Class

* But this is not good practice as it will create tight-coupling between J.A and D.B

Second Way

* To load the class into a JVM memory we have a static method called forName() which is present in Class name as Class(this class is present java.lang package) which will load the Driver class (provided by respective database server) into JVM memory.

Syntax – Class.forName(“com.mysql.jdbc.Driver”);

//fully qualified class name

1. **Establish the connection between JAVA application and DATABASE Server**

So to make the connection between java application and database server we use Connection interface.

Connection Interface

Connection interface present in JDBC API inside java.sql package whose implementation will be present in JDBC Driver which is provided by respective database server.

Syntax

Connection con = DriverManager.getConnection(“jdbc:mysql://localhost:3306?user=root&password=admin”)

**DriverManager**

DriverManager is helper class present in JDBC API inside java.sql package .

DriverManager contains two static methods

1. getConnection() method

getConnection() is factory/helper method which will create and returns the implementation class object of Connection Interface that why return type of getConnection() method will be Connection interface.

getConnection() method has overloaded in 3 varients

* getConnection(String url)
* getConnection(String url, properties info)
* getConnection(String url, String user , String password)

Whenever we use this getConnection() method, it throw a checked exception called SQLException.

1. **Create a Platform to execute SQL queries**

Platform is created using Statement interface, PreparedStatement Interface, CallableStatement Interface

Statement Interface

It is a interface present in JDBC API inside java.sql package whose implementation class will be present in JDBC Driver which is provided by respective database server.

Syntax

Statement stmt = con.createStatement()

Reference Variable(hold address of Connection Interface)

createStatement() method is a non-static factory/helper method which is present in Connection interface, used to create and return object of implementation class of Statement Interface that why return type of createStatement() method will be Statement Interface.

Whenever we use createStatement() method, it will throw a checked exception called SQLException.

1. **Execute the SQL queries**

To execute the sql queries we have 3 methods, which are present in Statement interface(which is then inherited by PreparedStatement Interface, later by CallableStatement Interface)

1. execute() method
2. executeUpdate method()
3. executeQuery method()
4. execute() method

execute() method is a generic method which is used with any type of sql queries. Return type is boolean

in case of DDL/DML – false

in case of DQL – true

Method Signature – boolean execute(“generic SQL queries”)

1. executeUpdate()

executeUpdate() is specialised method which is used to execute only DML queries.

Its return type is int which shows the number of rows affected.

Method signature- int exceuteUpdate(“only DML queries”)

1. executeQuery()

* executeQuery() is a specialise method which is used to execute only DQL queries
* its return type is ResultSet interface (why)
* because whatever data comes as outcome of DQL queries is called processed data which is stored in cursor or buffer which can fetched through getXXX() method which is present in ResultSet interface.

Method Signature – ResultSet executeQuery(“only DQL query”)

Whenever we try to execute all these methods a checked exception will come called SQLException.

**ResultSet Interface**

* ResultSet Interface is present in JDBC API inside java.sql package whose implementation class is present in JDBC Driver which is given by respective database server.
* ResultSet interface has three methods

1. getXXX()

to fetch the data from cursor or buffer memory.(when we execute the DQL queries the data is stored in cursor or buffer memory)

1. overloaded varients of getXXX() method are present

returnType getXXX(int column number)

datatype of that column

returnType getXXX(String columnName)

1. next()

used to check whether the next record is present or not in cursor/buffer memory, its return type is boolean true if present , false if not present

Method Signature - boolean next()

Initially ResultSet cursor will not be pointing towards any record in cursor or buffer memory.

1. absolute()

used to check whether a particular record is present or not in the buffer memory.

Method Signature – boolean absolute(int rowNumber)

Q How to create object of implementation class of ResultSet interface

* There are two ways

1st way

boolean val = pstmt.executeQuery()

if(val){

ResultSet rs = pstmt.getResultSet()

**}**

2nd way

ResultSet rs = pstmt.executeQuery()

**Placeholder**

Placeholder is a parameter which holds dynamic value at run-time from the user.

Represented by ?

**3 Rule to set the data for placeholder**

1. Set the data in placeholder before execution
2. Number of data should match the number of placeholder
3. Use setXXX() method to set the data for placeholder.

setXXX() method

* It is present in PreparedStatement.

Method Signature

Void setXXX(Placeholder number,Data)

**PreparedStatement Interface**

* PreparedStatement Interface is present JDBC API inside java.sql package whose implementation class is present in JDBC Driver which is provided by respective database server.
* PreparedStatement supports the concept of placeholder.
* PreparedStatement is also called pre-compiled Statement because compilation of query happens at the times of object creation of implementation class of PreparedStatement Interface.
* In PreparedStatement query compile once and execute many times.

Syntax-

Prepared Statement = con.prepareStatement()

prepareStatement() method is non-static helper method which create and return the object of implementation class of PreparedStatement Interface.

prepareStatement() method is present in Connection interface.

**Statement interface**

**PrepareStatement interface**

**CallableStatement Interface**

**COSTLY RESOURCES**

* Resources which will use system properties(CPU,memory) in stream(means they are using system properties continuously rather than one-time operation) are known as costly resources. Ex – Scanner, PrintWriter
* These resources must be close within the finally block, by using if condition to avoid NullPointerException, so that our application performance do not decrease.

Note- All interfaces of JDBC API are costly resources(Driver, Connection, Statement, PreparedStatement, CallableStatement, ResultSet)

Insert single and multiple record without taking user input

Insert single record using user input

Get the single data without and with user input.

Get single record with user input

Get multiple records without and with user input