JE®

JavaEra.com

CoreJava | JSP | Servlets | JDBC | Struts | Spring | Hibernate
Projects | FAQs | Sample Programs | eBooks | Certification Stuff
Question Banks | Communities | Tutorials | Softwares | Sample Resumes
Interview Tips | Forums | Discussions | Online Test Engines | Jobs

www.JavaEra.com

A Perfect Place For All Java Resources

www.JavaEra.com

Nataraz(Sathya)

JDBC notes

JavaEra.com

Advanced Java

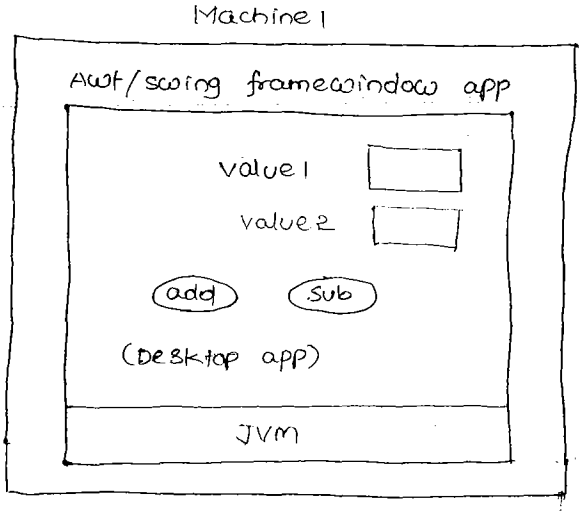
* Since Java can be used to develop all kinds of sofware applications so it is called as programming suite.
* Java can be divided into 3 modules. They are

(i) JSE – Java standard Edition
(ii) JEE – Java Enterprise Edition
(iii) JME – Java Mobile / Micro Edition

JSE:

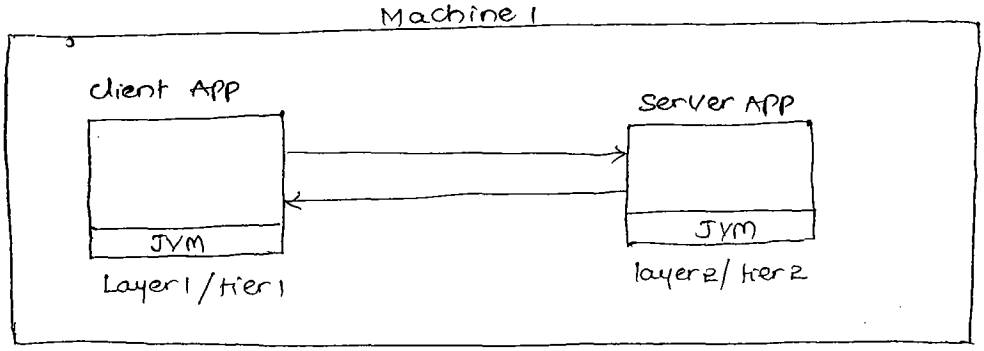
* It is installable software as jdk software.
* Latest version is 7.0 (dolphin)
* Java 6.0 is called as mustang and Java 5.0 is called as tiger.
* This module is given to develop stand-alone applications, desktop applications, two-tier applications....
* The application that is specific to one computer and contains main() is called as standard or stand-alone application.
* The stand-alone application that contains GUIness is called as desktop application.

Ex: Awt framewindow application, swing framewindow application.

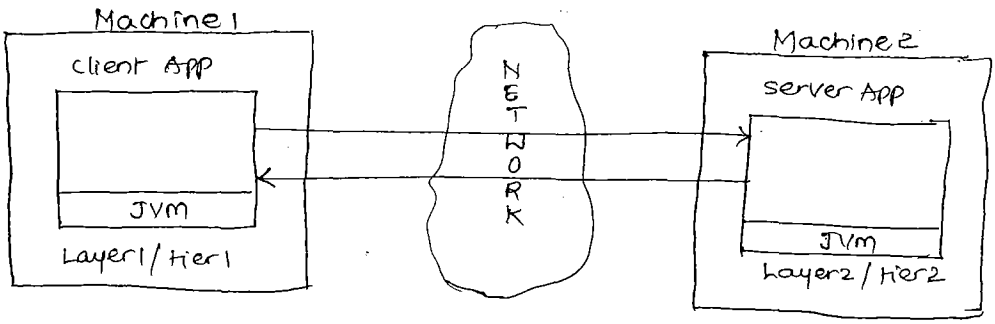


* The application that contains two layers communicating with each other is called as two-tier application.
* A layer represents logical partition in the application having logics.

* The layers of two-tier application can be there in a single computer (or) can be there in two different computers.



socker programming based client-server application (two-tier app)



socket programming based client-server application (two-tier app)

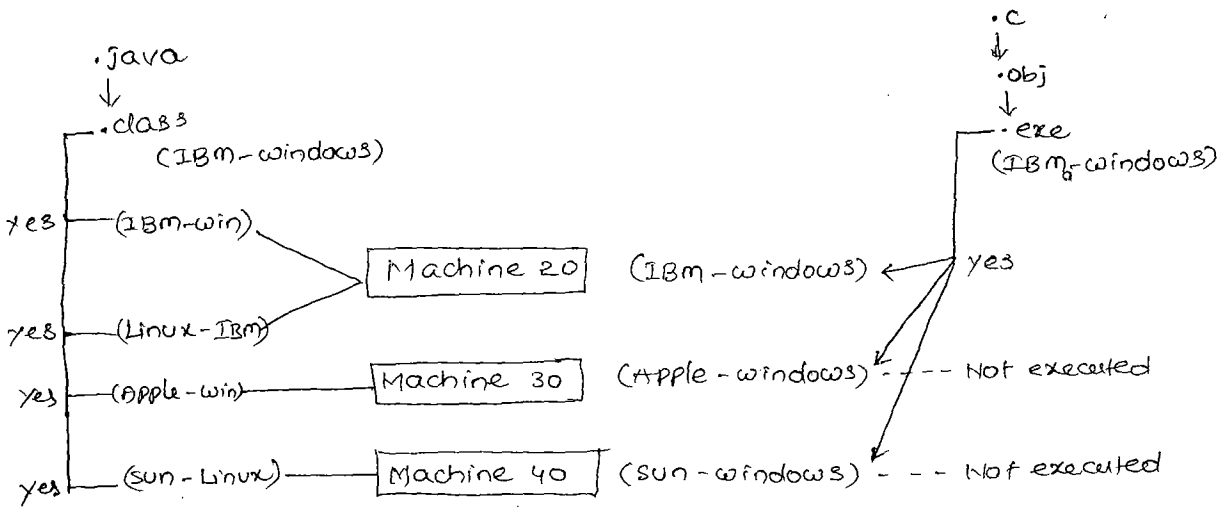
* To execute multiple Java applications parallelly (or) simulteneously from a single computer, multiple JVMs will be activated simulteneously (or) parallelly.
* Applet is not desktop application. It is a compiled java class that can be sent over the network as webpage giving alternate to the HTML based form pages.

<u>JSE</u> <u>Concepts</u> : awt, swings, collection frame work, JDBC, JNDI (Java Naming and Directory Interface) etc.

* The alternate technologies for JSE module are VB, VB.net, D2K. But these technologies based applications are platform dependent and architectural specific whereas the JSE module applications are architectural neutral and platform independent.
* As of now three popular architectures are there to manifacture computers. Each architecture use methodology and plan of manifacturing

computer. They are

(1) IBM architecture
(2) Apple architecture
(3) Sun architecture.



* The c, c++ generated .exe files not only contains operating system related instructions but they also contain instructions related to underlying computer architecture. Due to this we can say c, c++ languages are platform dependent and architecture specific whereas java programming language is platform independent and architectural neutral because the java compiler generated .class file doesn't contain underlying OS, computer architecture instructions.

JEE:

* JEE means Java Enterprise Edition.
* It is not installable software, it is given as specification.
* JEE specification contains rules and guidelines to develop webserver and application server software like weblogic, tomcat etc.

NOTE: working with JEE is nothing but working with one (or) other web-server (or) app server software to develop the app.

* These webserver, application server softwares are installable softwares.
* For JEE module JSE module is base module.

* Using jee module the following applications can be developed.

(1) web applications (websites)

(2) Distributed applications

(3) Enterprise applications

(4) n-tier applications

* The applications which can provide 24x7 access to their resources using internet environment are called as websites.

* The client-server applications which provide location transparency are called as distributed applications. If client application is able to reconize the server application location change dynamically are called as location transparency.

* A web application / website can be developed as distributed or non-distributed application.

* All credit card, debit card processing applications will be developed generally as distributed applications.

* A large scale application that deals with complex and heavy weight business logic by having middleware service support is called as enterprise application.

Ex: Banking application.

* The additional services that are configurable on the applications to make applications running perfectly in all the situations are called as middleware services.

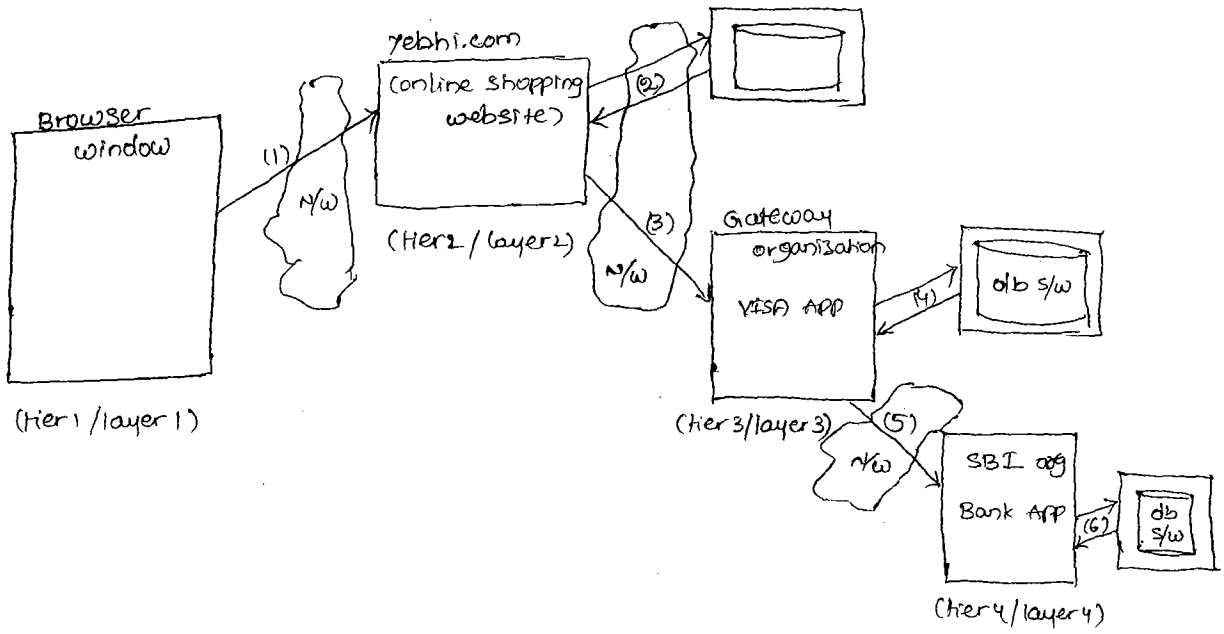
Ex: security service, Transaction management service, pooling service and etc...

NOTE: Middleware services are not minimum logics of the application, they are additional and optional logics to develop.

* The application that contains multiple layers / tiers (more than two) is called as n-tier applications.

Ex: online shopping website environment.

online shopping website environment (n-tier APP)



* Generally enterprise applications will be developed as n-tier applications.
* JEE module concepts are servlets, JSP, EJB, JTA (Java Transaction API), JMS (Java messaging service), Java mail, JAAS (Java Authentication and Autherization serive), JMX (Java management console), JCA (Java Connector API) and etc...
* struts, spring and hibernate technologies are not part of sunmicro systems JEE module. They are given by third party organizations as alternate and enhancement to JEE concepts.
* while working with all JEE concepts we must take support of webserver (or) application server softwares.
* majority projects of Java environment will be developed based on JEE module.
* Every project belongs one domain. This JEE module is suitable for developing banking financial service, Insurance, health care domain projects.
* JSE module is suitable to develop gaming, automation and retail domain projects.

* JME module is suitable for developing mobile gaming, tele communication domain projects.
* The latest version of JEE module is 6.

JME :

* JME means Java mobile/micro edition.
* for JME module JSE module is base module.
* It is installable software.
* Latest version is 2.x
* Given to develop mobile and micro applications in Java environment like sim cards, mobile games and etc.
* use wml (wireless markup language), wap in the application development.
* These applications can be embded with different types of devices (electronic, electrical, automobile and etc...).
* JSE, JEE module based applications can not be attached with (or) embedded with different types of devices by bringing the code outside the computer. But this is possible with JME module.
* The above process is called as embedded system programming and this is useful to add artifical intellegence to devices, that means we can make the devices to think and take the decisions.

Ex: A fully automated washing machine, Robot.

* To develop mobile related and artificial intellegence related applications in Java environment use JME module.

API (Application programming Interface) :

* It is the base for the programmer to develop certain software technology based software applications.
* Every software technology used built-in APIs. Using these APIs the programmers can develop user defined APIs and software applications.

* In "C" language API is set of functions which come in the form of header files.
* In "C++" language API is set of functions, classes which come in the form of header files.
* In "Java" language API is set of classes, Interfaces, enums, annotations which come in the form of packages.

Examples of Java APIs :

(1) AWT API → working with java.awt, java.awt.event packages.

(2) Reflection API → working with java.lang.reflect package

(3) JDBC API → working with java.sql, javax.sql packages

(4) Utility API → working with java.util package

* We can see three types of APIs in Java.

(1) Built-in APIs → given by sun micro system along with Jdk softwares.
Ex: AWT, JDBC etc...

(2) user defined APIs → given by programmers

(3) Third party APIs → given by third party organizations (other than sun micro systems and programmers).
Ex: Java Zoom API (given by JavaZoom organization, downloadable from www.javazoom.net).

* Generally the Java related APIs come as in the form of Jar files. (Java style zip file).
* JAVA_HOME\jre\lib\rt.jar file represents all the built-in APIs of Jdk software.
* The company who creates softwares is called as software vendor company.

Ex: IBM, sun micro system, oracle corporation and etc.

* The organization who gives their requirements to software development company is called as client organization.
  Ex: city bank, SBI, APSRTC etc..
* The company who takes the support of softwares given by software vendor companies and develop the project for client organizations is called as software development company.
  Ex: Infosys, wipro, polaries etc..

Software projects

- Software product
  Ex: Tally
  winzip tool
  NetBeans IDE
- Service sector project
  Ex: Bank App for citibank
  Ticket reservation App for APSRTC

* If project is designed and developed exclusively for one client organization by gathering requirements then it is called as service sector project.
* If project is developed based on common requirements and given to multiple clients then it is called as software product.
* For service sector project client organization will be there before development of the project.
* For software products client organizations will be there after the development of product.

What is the difference between static block and constructor?

A) static block is class level one time execution block. when JVM loads java class the static block of that class executes automatically. This block is useful to initialize static member variables of a class.

Constructor is object level one time execution block when JVM creates object for a class then the constructor of that class executes once automatically. Constructor is use ful to initialize instance variables of ~~java~~ class object.

* Class.forName (-) makes JVM of the current application to load given class into application from the memory. But this method does not create object of the loader class.

<u>Example application to understand above concepts.</u>

```
class Demo
{
    static
    {
        static System.out.println ("Demo: static block");
    }
    Demo()
    {
        system.out.println ("Demo: Constructor");
    }
}
class Test
{
    static
    {
        System.out.println ("Test: static block");
    }
    Test()
    {
        system.out.println ("Test: Constructor");
    }
public class TestApp
{
    static
    {
        system.out.println ("TestApp: static block");
    }
    public static void main (String args[])
    {
        sop (" TestApp: main(-)");
        Demo d1 = new Demo();
        Demo d2 = new Demo();
```

```
        Class.forName("Test");
        Class.forName("Test");
    }
}
```

O/P

```
TestApp : static block
TestApp : main(-)
Demo : static block
Demo : Constructor
Demo : Constructor
Test : static block
```

* with respect to above application we can see JVM loading java classes in the following situations.

(1) while creating first object for given java class

(2) when Class.forName () method is called.

(3) when java class is given to "java tool" for execution.

> java TestApp

Understanding Class.forName () :

Prototype:

public static Class forName(String name) throws ClassNotFoundException

* This method makes JVM to load given class or interface into application from memory dynamically at runtime. If they are not available to load this method throws checked exception called ClassNotFoundException.
* It is a static method (can be called with out object)
* available in java.lang.class (pre-defined)
* It is factory method
* A method of a java class that is capable of constructing and returning its own java class object is called as factory method.

Ex:

```
Thread t = Thread.currentThread();
String s1 = String.valueOf(10);
Calender c1 = Calander.getInstance();
Class c = Class.forName("Test");
```

```
Class c = Class.forName("Test");
```

* The above statement makes JVM to load class "Test" into application dynamically at runtime and returns object of (c) "java.lang.Class" referring the loaded class "test".
* In the above statement 'c' is not the object of the loaded class "Test", it is object of java.lang.Class referring the loaded class "Test". we can use this object 'c' to gather details about the loaded class "Test" to create object of loaded class "Test" and etc.. operations.
* If java method throws checked exception while calling that method we must handle the exception using try/catch blocks (or) we need to declare the exception to be thrown by using throws statement.

<u>Example application based on all the above statements</u>

```
// TestAppl.java
public class TestAppl
{
    public static void main (String args[]) throws Exception
    {
        Class c = Class.forName (args[0]);
        //gathered details and loaded class and Interface
        sop (args[0] + " is an interface ?" + c.isInterface());
        sop ("name of " + args[0] + " is :" + c.getName());
        sop ("super class of " + args[0] + " is :" + c.getSuperclass());
    } // main
} // Class
```

> javac TestAppl.java (Compile time)

> java TestAppl java.lang.String (run time)

* Command line argument args[0]
* Inorder to know class name of any java object programatically then call getClass() method on that object.

* This method is available as public method of "java.lang.Object" class

Ex:

```
String s1 = new String("ok");
sop("class name of s1 obj" + s1.getClass());
                                  → java.lang.String
java.util.Date d1 = new java.util.Date();
sop("class name of d1 obj" + d1.getClass());
                                  → java.util.Date
class c = Class.forName("java.awt.Button");
sop("class name of obj 'c' is:" + c.getClass());
                                  → java.lang.Class
```

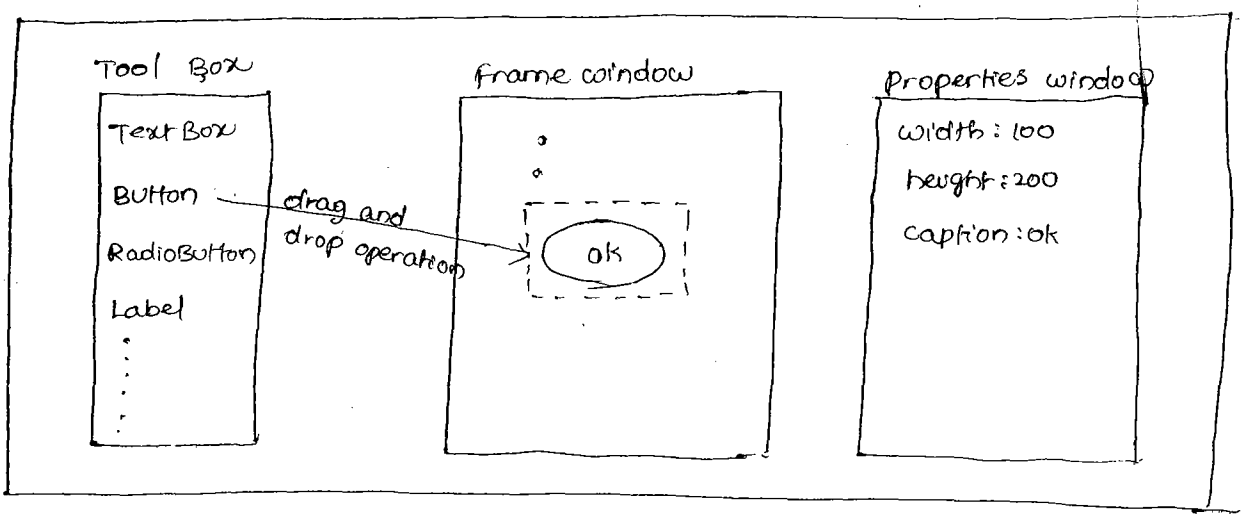
Problem with "new" keyword:

* "new" keyword creates the object of a java class at runtime but it expects the presence of the class (or) availability of the class at compile time.

```
Test t = new Test();
```

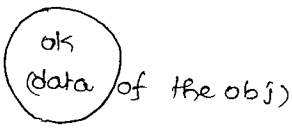
* In the above statement object 't' will be created at run time but it expects the presence of class "Test" at compile time.
* This indicates new keyword is not suitable to create the object of a class whose class name comes to application dynamically at runtime. To overcome this problem use "newInstance()" method of java.lang.Class.
* The newInstance() method can create object of given java class at runtime even though class name is supplied to the application dynamically at runtime. This kind of object creation process is required while are performing drag and drop operations in GUI builder applications.

GUI Builder Application (Part of IDE s/ws)



String s = new String ("ok");

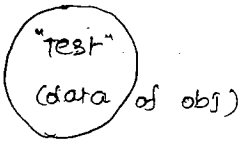
s (String class obj)



class c = Class.forName("Test");

(The given class/interface to load)

c (obj of java.lang.class)



* In a running java application if we want to refer a java class or interface then we need to use the object of java.lang.Class
* When you call java method catching and using the return value of java method is purely programmer choice.

Class.forName ("Test"); // valid statement

Class c = Class.forName ("Test"); // valid statement

* When java object is created constructor will execute and the instance variables will be assigned with default values.

creating object of java class by using newInstance() method by getting class name into application dynamically at runtime.

```
//Test.java
public class Test
  {
      int a;
      float b;
      public Test()
        {
          System.out.println("Test: 0-arg Constructor");
        }
      public String toString()
        {
          return "a="+a+ "b=" +b;
  } //class

>javac Test.java
```

```
//objTest.java
public class objTest
  {
    public static void main (String args[])
    {
      try
        {
          //load given class into application
          class c = Class.forName(args[0]);
          // "c" object refers to the loaded class
          // create object for loaded class
          Object obj = c.newInstance();
                          ↳ creates object of a java class
                            that is referred by object "c"
          // print object data
          System.out.println("object data is:" +obj.toString());
        }
      Catch (InstantiationException iae) // to handle known exception
        {
          iae.printStackTrace();
        }
```

```
    catch (ClassNotFoundException cnf) // to handle known exception
      {
        cnf.printStackTrace();
      }
    catch (Exception e)   // to handle all unknown exceptions.
      {
        e.printStackTrace();
      }
  } // main
} // class
```

> javac objTest.java (Compile time)

> java objTest Test (runtime)

> java objTest java.awt.Rectangle (runtime)

O/P Test: 0-arg Constructor
object data is : a=0 b=0.0

* when the above newInstance() method is used to create object of abstract class (or) interface then that method raises "java.lang.InstantiationException"
* It is always recommended to catch and handle the exceptions by keeping multiple catch blocks after try block. This helps the programmer to handle both known and unknown exceptions and to differentiate the logics based on the exception that are raised (refer above).

<u>what is the difference between <exception obj>.toString() and <exception obj>.printStackTrace() ?</u>

* toString() method prints name of the exception class along with description when exception is raised. whereas printStackTrace() prints the route cause and hierarchy in the application execution when exception is raised.
* Every java method is executed in stack memory. This stack memory is filled up with messages when exception is raised during method execution.

* The printStackTrace() call displays these messages on console monitor.

(*) When java does not support pointers why does it throws NullPointerException?

(A) When java method is called on a variable that holds null value this NullPointerException will be raised. To solve this problem make that variable referring to object before calling the method.

Ex: Problem:

```
Date d;
d.toString(); // raise java.lang.NullPointerException
```

solution:

```
Date d = new Date();
d.toString(); // success
```

* In java.lang.NullPointerException class name the word pointer is no way related with C, C++ pointers. Look at that word with dictionary meaning which says method is called on a variable that points to null value.

— • —

How many ways are there to create object of a java class?

There are multiple ways. They are

(1) By using new keyword
(2) By using static factory method
(3) By using instance factory method
(4) By using factory pattern
(5) By using newInstance()
(6) By using cloning process
(7) By using deserialization process....

1) By using new keyword

```
String s1 = new String("Hai");
Date d = new Date();
```

2) By using static factory method

* A method in a java class that is capable of constructing and returning its own java class object is called as static factory method.

```
Thread t = Thread.currentThread();
String s1 = String.valueOf(10);
class c = class.forName("Test");
```
} static factory methods

NOTE: when java class contains only private constructors to create object of that class outside the class we take the support of static factory methods.

Ex: java.lang.class is having only private constructors so we always use class.forName() method as shown above to create the object.

3) By using instance factory method:

```
String s = new String("hello");
String s1 = s.concat(" how are u");   // s1 holds "hello how are u"
```

↓ instance factory method

* static factory methods will be called without object. Instance factory methods will be called by using object.
* In order to create a new object for java class by using existing object and its data take the support of instance factory methods.

NOTE: In the above example concat() method is using object "s" and its data while constructing "s1" object.

4) By using factory pattern

* If method of one java class is returning object of other java class then it is called as factory pattern.

Ex: StringBuffer sb= new StringBuffer("how are you");

String s = sb.subString(6, 3);

The method of StringBuffer class is returning object of String class. (factory pattern).

* If method of one java class wants to return another java class object then these both classes need not to be there in inheritance relationship. If they are visible to each other that will be enough.

5) By using newInstance() method

class c = Class.forName("java.awt.Button");

Button b = (Button)c.newInstance();

* Here "c" represents the loaded class java.awt.Button but it is the object of java.lang.Class.

→ creates and returns the object of java.awt.Button class representing with 'c' object

6) By using cloning process:

* The process of creating new object for a java class having existing object data is called as cloning process.
* programmer can perform cloning on only clonable objects.
* The object whose class implements marker interface called java.lang.Clonable is called as cloneable object.
* The interface whose implementation makes JVM to provide special run-time behavior for the objects of class is called as marker interface (or) tag interface.
* A marker interface can be there with methods (or) without methods.

Ex: java.lang.Clonable
java.lang.Serializable
java.io.Externalizable
java.rmi.Remote
javax.servlet.SingleThreadModel
java.lang.Runnable

* when object is created through cloning process constructor will not be executed.
* The clone() method of ~~lan~~ java.lang.object class can perform cloning operation.
* when new object is created through cloning process the new object and existing object allocates memory seperately and acts as two different objects having two different hash codes. So modification done in one object will not reflected the other object, and vice versa.
* Every object will be identified through its unique hash code number.

Example application on cloning

```
// Test.java
class Test implements java.lang.cloneable
  {
    int a;
    float b;
  public Test()
    {
      sop("Test: 0-param Constructor");
    }
  public Test (int x, int y)
    {
        a = x;
        b = y;
        sop("Test: 2-param constructor");
    }
  public String toString()
    {
      return "a =" + a + "b =" + b;
    }
  public object myclone() throws Exception
    {
      return super.clone();
                 ↳ calls clone method of java.lang.object class
    }
} // Test

Public class MainApp
  {
      p s v m (String args[])
        {
```

```
//original obj
Test t1 = new Test(10,20);
sop("t1 obj data is:"+t1.toString());
//create obj through cloning
Test t2 = (Test) t1.myClone();
sop("t2 obj data is:"+t2.toString());
sop("t1 obj hashcode is:"+t1.hashCode());
sop("t2 obj hashcode is:"+t2.hashcode());
//modify original obj data
t1.a=30;
sop("t1 obj data is:"+t1.toString());
sop("t2 obj data is:"+t2.toString());
t2.b=56.4f;
sop("t1 obj data is:"+t1.toString());
sop("t2 obj data is:"+t2.toString());
}// main
}// class
```

```
>javac mainApp.java   (compilation)
>java Main App        (execution)
```

7) By using deserialization process :

* The process of writing data of an object to a file is called as serialization.
* The process of reading data from a file and constructing new object having that data is called as deserialization.
* When object is created to deserialization constructor will not execute.

* During Java application execution the RAM memory will be utilised to maintain application data. (input values and results generated by application), due to this we can not use application data after the execution of the application (or) across the multiple executions of the same application.
* To overcome the above problem store application data in permanent place like files and DB softwares (persistance stores) through persistance operations.

Java app —uses (I/O streams)→ Files
(serialization and deserialization)

Java app —jdbc concepts→ DB software
(oracle, sybase, mysql)

* The place where we can store and use data for long time is called as persistance store. Ex: file, database software.
* A logic that is used to interact with persistance stores and to manipulate the data is called as persistance logic.
* Performing insert, update, delete and read operations on persistance stores is called as performing persistance ~~stor~~ operations.
* Persistance operations can be called as CURD operations (or) CRUD operations (or) SCUD operations.

C → create (insert), U → update (modify), R → Read (select)
D → Delete (Remove).

S → select
C → create
U → update
D → Delete

Limitations with files as persistance stores:

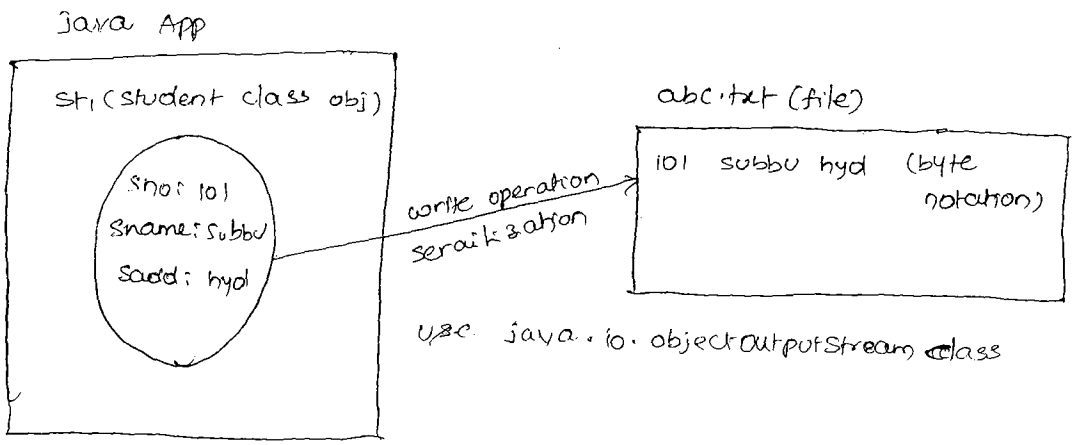
(1) Can not manage huge amount of data
(2) No security
(3) No query language support to manipulate the data.
(4) Delete and update operations are complex operations to perform.
(5) Relationships are not possible.
(6) Merging and comparision of data is complex operation.

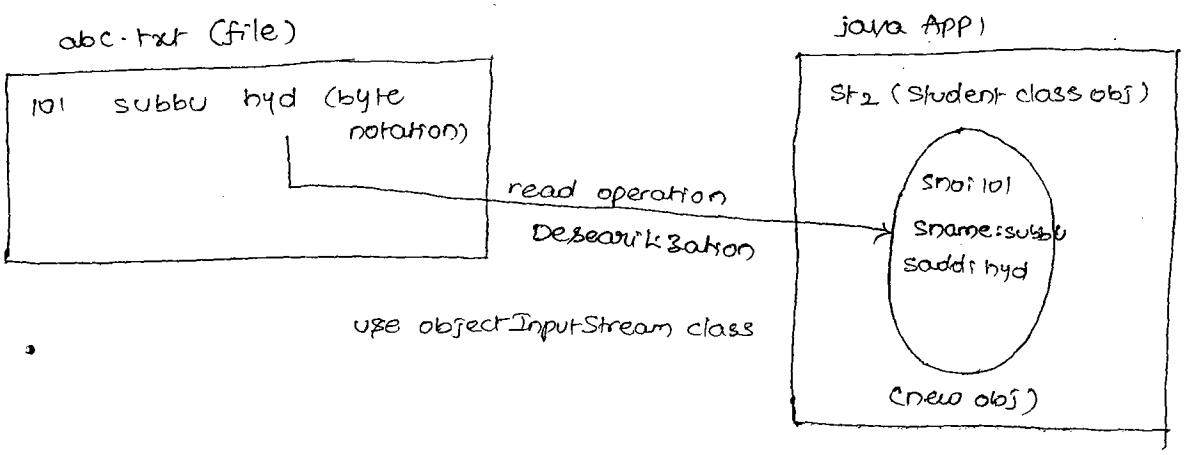
NOTE: To over come above problems use database software as persistance store.

NOTE: In small scale applications like desktop, mobile games use files as persistance store. In medium, large scale applications like bank apps, websites use database software as persistance store.

Serialization and deserialization:

* In small applications the industry generally prefers serialization and deserialization persistance operations on the files.
* The process of capturing object data and writing that data into a file is called as serialization.

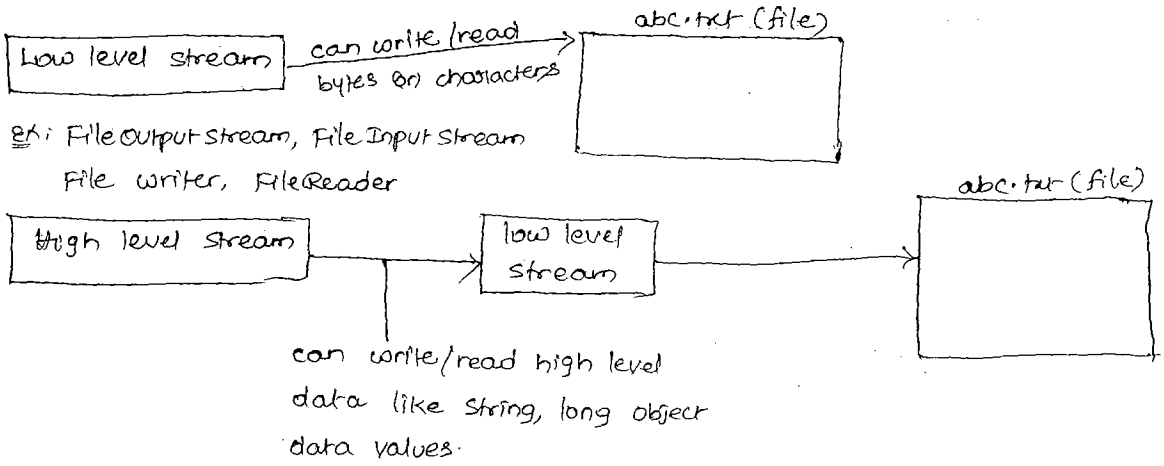




* In serialization process object will not be written to a file but the data of the object will be written to a file.
* The process of reading data from a file and constructing new java object having that data is called as deserialization.
* A stream is a continuous flow of data that represents communication channel between application and destination resource (file).
* Byte streams can write and read bytes.
* character streams can write and read characters.

<u>what is the difference between highlevel stream and lowlevel stream?</u>

(A)



Ex: DataOutputStream
DataInputStream
ObjectOutputStream
ObjectInputStream

* High level streams interacts with files through low level streams and can work with high level data like string values, object values and etc.
* Low level streams interacts with files directly and can deal with low level data like characters and bytes.

Sample Application

(1) Read sno, name, m_1, m_2, m_3 values from key board. → presentation logic

(2) calculate total and avg

total = $m_1+m_2+m_3$ /3.0f;
avg = total/3.0f;
} Business logic

(3) generate rank for the student — business logic

(4) store student details in db table as record — persistance logic

(5) display student details and results on monitor — presentation logic.

* presentation logic provides user interface for end user to supply input values and to display results
* Business logic generates results based on given input values (It is main logic of the application)
* persistance logic manipulate application data in persistance stores.
* If entire data of an object is written to a file that is called complete serialization.
* If partial data of an object is written to a file then that is called selective serialization.
* If class is having static or transient variables then objects of that class supports only selective serialization.
* static member variables allocates memory outside the object so they doesn't participate in serialization.
* Transient member variables allocates memory inside the object but they will be marked for not participating in serialization process.

* Programmer can perform persistance operations only on serializable objects.
* Java object becomes serializable only when class of that object implements java.io.serializable interface (marker, empty interface).

```
public class student implements java.io.serializable
{
    int sno;
    String name;
    float avg;
    - - - - - -
    - - - - - -
    - - - - - -
}
```

objects of this class supports only complete serialization.

```
public class student implements java.io.serializable
{
    int sno;
    String name;
    static / transient float avg;
    - - - - - - - -
    - - - - - - - -
}
```

objects of this class supports only selective serialization.

<u>Sample code for serialization</u>

```
FileOutputStream fos = new FileOutputStream ("abc.txt"); // low level stream
ObjectOutputStream oos = new ObjectOutputStream (fos); // high level stream
//create object for student class
student st1 = new student (101, "raja", 47.65f);
// Perform serialization
oos.writeObject (st1);
```

↳ captures "st1" object data and writes that data to a file.

```
// close streams
fos.close();
oos.close();
```

Prototype of writeobject( ) method

public final void writeobject (object obj) throws IOException.

Three important statements.

(1) If java method parameter type is java class name then you can call that method by having that class (or) one of its subclass object as argument value.

(2) If java method parameter type is abstract class name then you can call that method having one subclass object of that abstract class as argument value.

(3) If java method parameter type is interface name then you can call that method with one implementation class object of that interface.

Sample code for deserialization

```
// create highlevel stream
ObjectInputStream ois = new ObjectInputStream (new FileInputStream
                                                ("abc.txt"));
//perform deserialization
    object obj = ois.readobject();
// type casting
    student st2 = (student) obj;
// close streams
    ois.close();
```

→ reads data from the abc.txt file and creates and returns java class object having that data.

Prototype of readObject() method

public final object readobject () throws IOException, ClassNotFound Exception.

(1) If java method return type is concrete class name then that method can return either that concrete class object or one of its subclass object.

(2) If java method return type is interface name then that method returns one implementation class object of that interface.

(3) If java method return type is abstract class name then that method returns one subclass object of that abstract class.

Application on serialization and deserialization

Resources of the application

student.java

writeTest.java (perform serialization)

readTest.java (perform deserialization)

```
// student.java
import java.io.*;
public class student implements serializable
                                ↳ mandatory
{
    int sno;
    string sname;
    float avg;
    public student()
    {
        system.out.println("student : 0-param Constructor");
    }
    Student (int x, String y, float z)
    {
        sno = x;
        sname = y;
        avg = z;
        sop ("student : 3-param constructor");
    }
    public void disp()
    {
        sop(" sno="+sno+ "sname="+sname+ "avg="+avg);
    }
}
```

```
// writeTest.java
import java.io.*;
public class writeTest
{
  public static void main(String args[])
  {
    try
    {
      // create stream objects
      FileOutputStream fos = new FileOutputStream("abc.txt");
      ObjectOutputStream oos = new ObjectOutputStream(fos);
      // perform serialization
      student st1 = new student(101, "anil", 45.78f);
      oos.writeObject(st1);
      // close streams
        fos.close();
        oos.close(); sop("serialization is completed");
    } // try
    catch (IOException ioe) // to handle known exception
    {
      ioe.printStackTrace();
    }
    catch(Exception e)  // to handle unknown exception
    {
      e.printstackTrace();
    }
  } // main
}// class
```

```
> javac student.java
> javac writeTest.java
> java writeTest
```

NOTE: when serialization is done on the object it not only writes object data to a file but it also writes structure of the class to that file.

* If application tries to perform serialization on non-serializable object then ~~that~~ the application throws java.io.NotSerializable

Exception.

```
//readTest.java
import java.io.*;
public class readTest
{
    public static void main(String args[])
    {
        try
        {
            //create stream
            objectInputStream ois = new objectInputStream (new File
                                                InputStream ("abc.txt");
            //perform deserialization
             object obj = ois.readObject();
             //type casting
             student st2 = (student) obj;
             //display object (st2) data
              st2.disp();
              //close stream
              ois.close();
            }
        catch (IoException ioe)
            {
                ioe.PrintStackTrace();
            }
        catch (ClassNotFoundException cnf)
            {
                cnf.printStackTrace();
            }
    }//main
}//class
```

```
> javac readTest.java
> java readTest
```

* There is no terminology called selective deserialization. The object data that is return through serialization process must be retrieved through de serialization process.
* The structure of the class must be same while performing serialization and deserialization operations. otherwise the deserialization

,, Invalid class Exception.

a object over the network it must be

able object.

its and collection framework data structures

default.

list are non-serializable java objects. Can

can send these java objects over the

NO, because arraylist object and its elements must be there as serializable objects in order to send array list object over the network.

* A specification is a document that contains set of rules and guidelines to develop the software or software applications.
* There are two types of specifications.

(1) open specification (Free to every one)

Ex: JDBC specification, servlet specification and etc...

(2) Propraitory specification (specific to one organization)

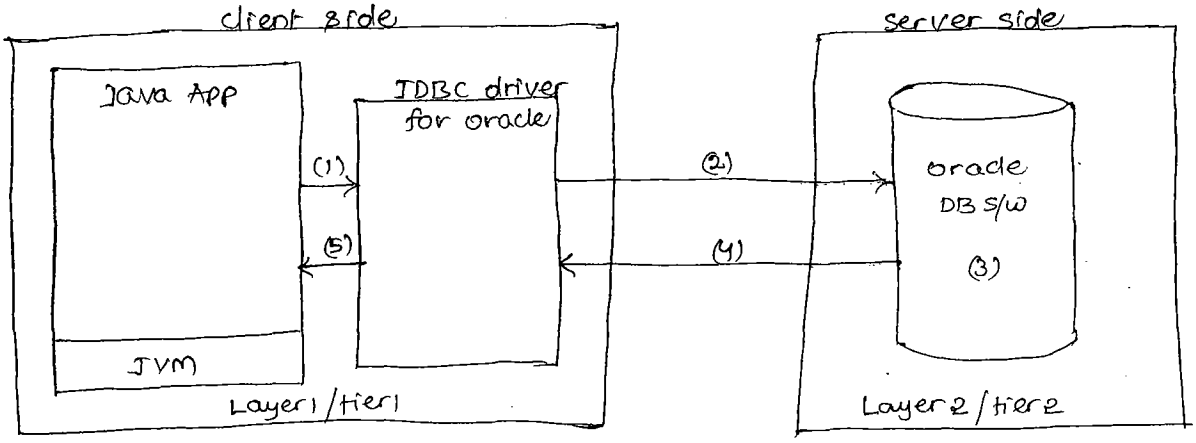
Ex: .net specification.

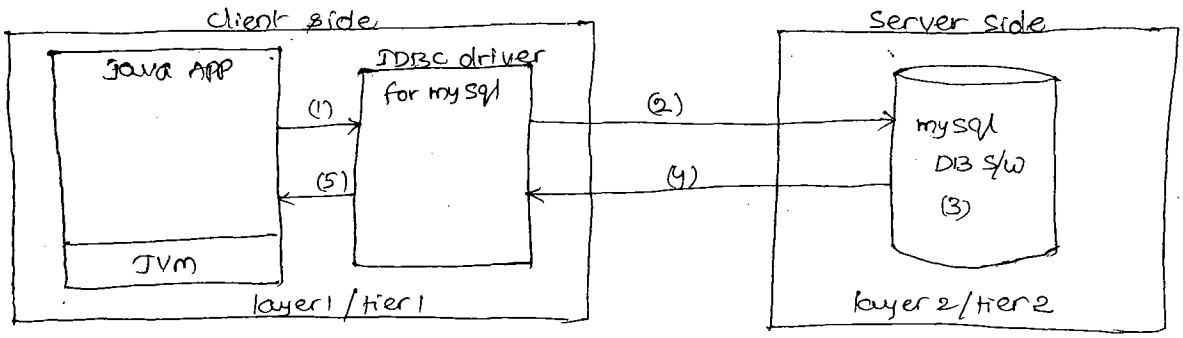
NOTE: Any company can create softwares based on open specification

* Only the creator of specification is allowed to develop software based on propraitory specification.

JDBC

* JDBC is an open specification given by SunMicrosystems having rules and guide lines to develop JDBC drivers.
* JDBC driver is a bridge software between java application and database software. It converts java calls to database calls and viceversa.





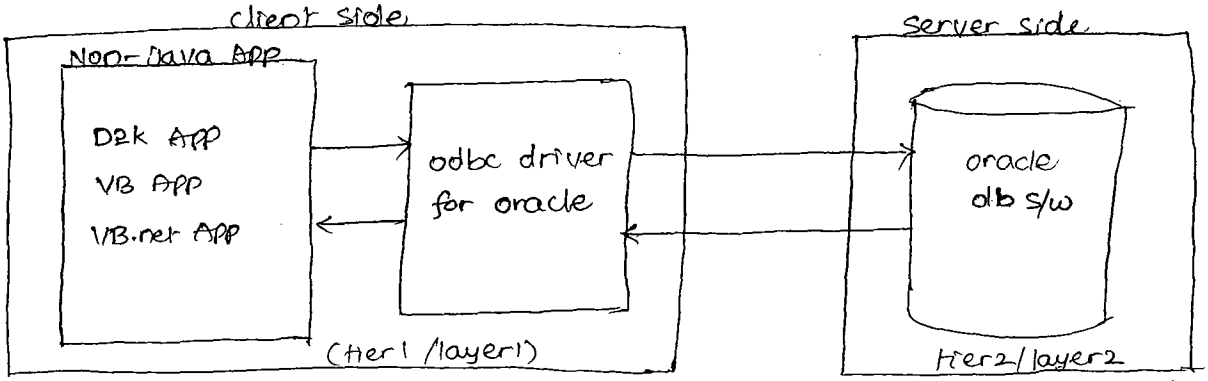
NOTE: The above JDBC applications are two tier applications. The two layers of this two tier application can be there either in same computer or in two different computers.

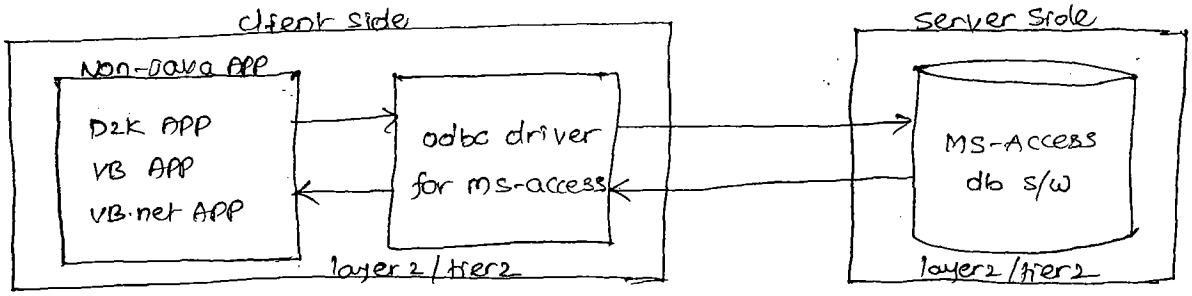
* Each JDBC driver is specific to one database software.
* We can get JDBC drivers from three parties.
  (1) Sun micro systems
  (2) Database vendor
  (3) Third party vendor.
* It is recommended to use Database vendor supplied JDBC drivers.
* Non-java applications use ODBC drivers to interact with database softwares.
* Every ODBC drivers is specific to one database software.

What is an ODBC (open database connectivity)?

* ODBC is an open specification ~~that contains~~ given by X-open company that contains set of rules and guide lines to develop

ODBC drivers for different database softwares.



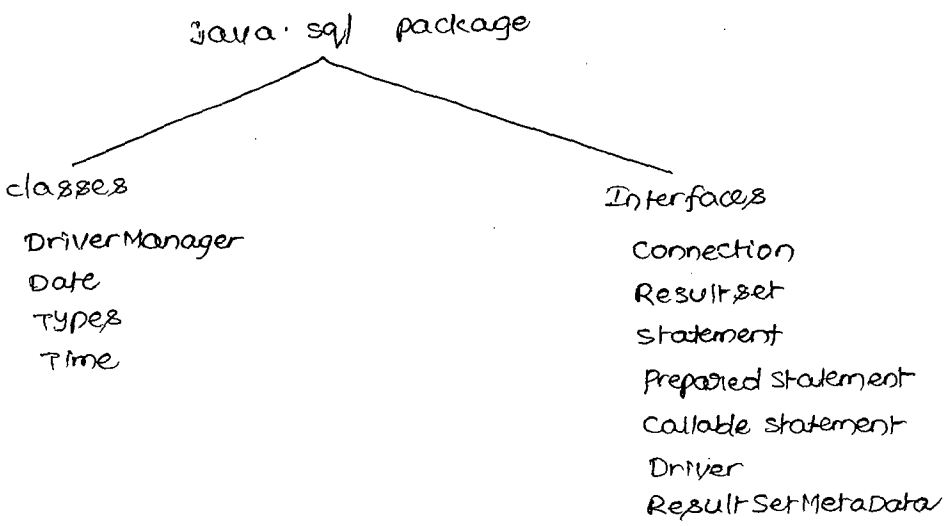


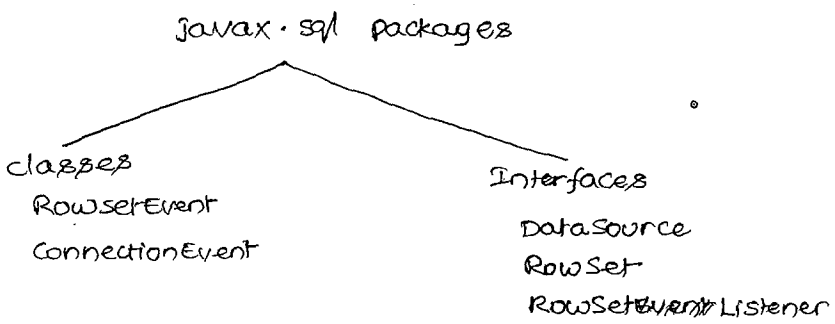
when all other software technologies are using ODBC drives. why java applications need an exclusive JDBC driver to interact with database softwares?

→ ODBC drivers are given based on ODBC specification. This specification is designed in 'c' language using pointers support. Since java doesn't support pointers the java applications need exclusive java based JDBC drivers to interact with database softwares.

---

* Every specification contains an api representing rules and guidelines.
* In java the abstract methods of API interfaces and abstract classes represent rules whereas the concrete methods of APIs abstract classes, concrete classes represent guidelines.
* The JDBC specification related API comes in the form of java.sql, javax.sql packages.





* In the above given JDBC API packages concrete methods of abstract, concrete classes represent guidelines to develop JDBC driver. Whereas the abstract methods of interfaces and abstract classes represent rules to develop JDBC driver. The vendor company implements these rules and follows these guidelines while developing JDBC driver for each database software.
* Implementation of all the interfaces that are given in the packages of JDBC API is not the responsibility of programmer, it is the responsibility of vendor company who gives JDBC driver to the market.
* Each JDBC driver is set of classes implementing various interfaces of JDBC API packages. while defining methods of these interfaces the classes contains logic to interact with specific database software.
* Since all JDBC drivers ~~are~~ given for different database softwares coming from different vendors are given based on common JDBC specification rules and guidelines given by sunmicrosystems the way we work

with all these JDBC drivers in our java applications to interact with database softwares is going to be much similar

sample code to understand JDBC specification, JDBC driver implementation and utilizing JDBC driver from java applications

Assume this is the JDBC specification

MyConnect.java (Interface)

```
public interface MyConnect
  {
    public void getConnection();  //rule
  }
```

Assume this is JDBC driver for oracle (vendor1)

```
public class OracleConnect implements MyConnect
  {
      public void getConnection()
        {
          - - - - -
          - - - - - logic to interact with oracle db s/w
        }
  }
```

Assume this is JDBC driver for Mysql (vendor2)

```
public class MysqlConnect implements MyConnect
  {
      public void getConnection()
        {
          - - - - -
          - - - - - logic to interact with mysql db s/w
        }
  }
```

Application Development

App1 (Java application)

activates JDBC driver for oracle

call getConnection(); // makes App1 taking with oracle db s/w

App2 (Java application)

activates JDBC driver for mysql

call getConnection(); // makes App2 taking with mysql db s/w

NOTE: In the above sample code App1, App2 applications are using two different JDBC drivers in much similar manner to interact with two different database softwares.

* Generally vendor companies supply their JDBC drivers in the form of jar files.
* To get the source code from compiled code use java decompiler.
  Ex: DJ Decompiler (seperate installation)
* Even though every JDBC driver software contains lots of classes but the JDBC driver will be identified through its driver class name.
* The java class that implements java.sql.Driver interface directly or indirectly is called as JDBC driver class.
* To activate JDBC driver from our java applications we need to work with the driver class name of that JDBC driver.
* JDK software gives one built-in JDBC driver. Its driver class name is sun.jdbc.odbc.JdbcOdbcDriver
  ↳ package ↳ Driver class implementing java.sql.Driver interface
* Oracle database software gives one built-in JDBC driver. Its driver class name is oracle.jdbc.driver.OracleDriver
  ↳ package ↳ Driver class
* we can expect odbc drivers based on odbc specification from three parties.
  (1) X-open Company
  (2) DB Vendor
  (3) Third party vendor

NOTE: The windows XP OS installation gives us multiple ddbc drivers.

* Every odbc driver is identify through its DSN. There is a possibility of creating three types of dsns.
  (1) User DSN → specific to currently logged in windows user
  (2) System DSN → visible all the windows users of a machine.

(3) File DSN → sharable in the network machines.

Procedure to create Microsoft ODBC driver for oracle:

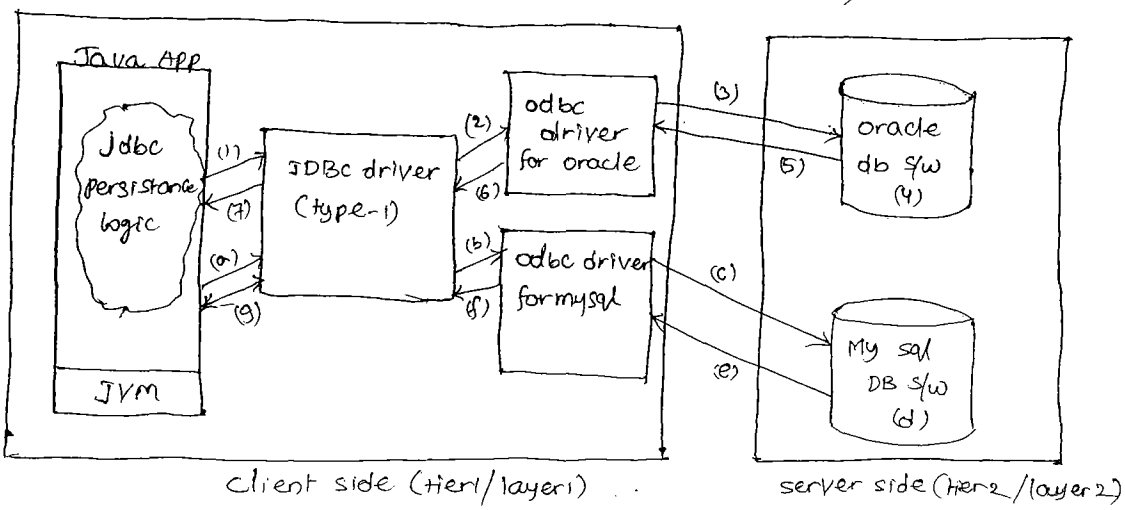
start → settings → control panel → performance and maintanance → Adminstrative tools → datasources → user dsn → add → microsoft ODBC for oracle → finish → data source name [xyz] username [scott] server [optional for local or personal oracle software installation, manditory for the centralized installation of oracle software (gather dns string)] → ok → ok

* There are four different methodologies or mechanisms to develop Jdbc drivers based on the rules and guidelines of Jdbc specification. They are.

(1) Type 1
(2) Type 2
(3) Type 3
(4) Type 4

* Every mechanism talks about architecture or plan to develop Jdbc drivers.
* The type1 mechanism based drivers are given to take the support of odbc drivers while interacting with database softwares. In this process the type1 driver takes the support of native code to communicate with odbc drivers.

Type1 driver architecture (partial)



* JDBC driver type-1 is not specific to any database software because it won't interact with database software directly. It interacts with database softwares by using the database software specific odbc driver.
* Only sun micro system is supplying type-1 mechanism based Jdbc driver as built-in Jdbc driver of Jdk software.
* Jdk software supplies a basic and built-in service called driver manager service to manage set of Jdbc drivers and to establish the connections with database softwares by using Jdbc drivers.
* Every JDBC driver must be registered with drivermanager service. For this create JDBC driver class object and keep that object in driver manager service.
* our java applications can activate and work with drivermanager service by dealing with java.sql.DriverManager class.

Jdk's built-in driver is Type 1 driver

this driver's class name is: sun.jdbc.odbc.JdbcOdbcDriver

To register the above type1 driver with DriverManager Service

```
//create object for Jdbc drivers class
sun.jdbc.odbc.JdbcOdbcDriver obj = new sun.jdbc.odbc.JdbcOdbcDriver();
// register jdbc driver with Drivermanager service
Drivermanager.registerDriver(obj);
```

* The JDBC API of Jdbc specification given by sunmicro systems will be used in two different angles.
  (1) As rules and guide lines by vendor companies to develop JDBC drivers.
  (2) As base for the programmer to activate JDBC driver and to develop JDBC persistance logic from Java applications. This persistance logic is responsible to manipulate data base data (CURD operations)
* one JDBC driver type1 can interact with multiple odbc drivers of different db s/ws.

Standard steps to develope jdbc code in our java Application

(1) register jdbc driver with Drivermanager service
(2) Make Drivermanager service to establish Connection with DB s/w.
(3) create statement object.
(4) use statement object to send sql queries to database software and to execute them in database software.
(5) use statement object to gather results from database software.
(6) Process the results
(7) close connection with database software

* statement object acts as courier service or vehicle to send and receive inputs and outputs between java application and database software. This is base for the programmer to manipulate the database data from java applications.
* connection between java application and database software represents the communication channel between java application and database software.
* To establish this connection the driver manager service uses an appropriate one of the registered JDBC driver.
* The code that is not written in java but invokable from java applications is called as native code.
* JDBC driver type1 internally uses this native code to interact with ODBC drivers.
* Exp Except type1 mechanism drivers no other mechanism drivers (type-2, type-3, type-4) use ODBC drivers support to interact with database softwares.

write a java application to establish the connection with oracle database software by using JDBC driver type-1 (Built-in JDBC driver of Jdk software)

The software setup that is required

jdk any version (jdk 1.6)
oracle db s/w (any version)

dsn created for microsoft odbc driver for oracle

```
// ConnTest.java
import java.sql.*;

public class ConnTest
{
    public static void main (String args[]) throws Exception
    {
        //create JDBC driver class object (Type-1 driver related)
          sun.jdbc.odbc.JdbcOdbcDriver obj= new sun.jdbc.Odbc.JdbcOdbcDriver();
        // register jdbc driver with drivermanagerservice
          Drivermanager.registerDriver(obj);
        //Establish connection with database software.
          Connection con = Drivermanager.getConnection("jdbc:odbc:oradsn", "scott", "tiger");
                                                        Protocol
                                                        subname(dsn)
                                                        db url
                                                        db user
                                                        db password
        if (con == null)
            system.out.println("Connection is not established);
        else
            system.out.println("Connection is established");
        } // main
    } //class
```

> javac ConnTest.java
> java ConnTest

* Based on the protocol specified in the url of Drivermanger.getConnection() method the Drivermanager service picks up an appropriate JDBC driver even though more JDBC drivers are registered with Drivermanagerservice. The Drivermanager service uses that selected/picked up driver to establish the connection with database software.

* jdbc:odbc protocol makes DriverManager service to pickup the registered type-1 jdbc driver.
* jdbc:odbc is application level protocol
  jdbc → main protocol, odbc → sub protocol
* Protocol is a set of rules followed by two parties who participates in communication.
* Network level protocol contains rules to get communication between two physical computers. Ex: TCP/IP
* Application level protocol contains rules to get communication between two softwares or software applications. Ex: HTTP, jdbc:odbc
* JDBC:ODBC is the application level protocol containing set of rules to get communication between DriverManager service and database software.
* Most of the application level protocols can run on the top of network level protocols.

```
Connection con = DriverManager.getConnection("jdbc:odbc:oradsn", "scott", "tiger");
```

con → jdbc connection object representing connectivity with database software

* In the above statement how can you say "con" is an object when java.sql.Connection is an interface?

→ In the above statement con is object of underlying JDBC driver supplied java class that implements java.sql.Connection Interface.

To know that class name

```
system.out.println("Con obj class name is:" + con.getClass());
```

In type-1 driver con object class name is "sun.jdbc.odbc.JdbcOdbcConnection" and this class implements java.sql.Connection Interface indirectly as shown below.

java.sql.Connection (Interface)

↑ Extends

sun.jdbc.odbc.JdbcOdbcConnectionInterface (Interface)

↑ Implements

sun.jdbc.odbc.JdbcOdbcConnection (class)

* class "JdbcOdbcConnection" is implementing java.sql.Connection interface.

NOTE: DriverManager.getConnection() method call returns object of a java class that implements java.sql.ConnectionInterface.

* when interface reference variable points to its implementation class object then that reference variable can also be called as indirect object of implementation class.

```
Connection Con = DriverManager.getConnection("jdbc:odbc;oradsn",);
```

* In the above statement Con is the reference variable of java.sql.ConnectionInterface pointing to object of a class that implements java.sql.ConnectionInterface, so "Con" also becomes object of that class.

In our JDBC code why we are not specifying class name of JdbcConnection object (Con)?

(A) The class name of connection object will change based on the JdbC driver we use so if that class name is specified our JdbC code becomes specific to one jdbc driver so we never specify that class name to write jdbc code in a common way for all jdbc drivers.

Rewriting ConnTest.java

```
// ConnTest.java
import java.sql.*;
public class ConnTest
{ public static void main(String args[])
  {
```

```
//load jdbc driver class
class.forName ("sun.jdbc.odbc.JdbcOdbcDriver");
//here type1 driver will be registered with drivermanager
                                        service internally

//establish connection with db s/w
Connection con = DriverManager.getConnection ("jdbc:odbc: oradsn",
                                              "scott", "tiger");
if (con == null)
    system.out.println("Connection is not established");
else
    System.out.println("Connection is established");
} //main
} //class
```

<u>In the above code class.forName (-) method call just makes JVM to load given jdbc driver class. How can you say jdbc driver is registered with DriverManager service in that process?</u>

1) class.forName (-) makes JVM to load sun.jdbc.odbc.JdbcOdbc_Driver class → static block of this driver class executes automatically → This static block contains logic to create object of jdbc driver class and to register that object with DriverManager service by calling Drivermanager.registerDriver (-) method. In the static block of sun.jdbc.odbc.JdbcOdbcDriver class we can see the following code registering type-1 driver with Drivermanager service.

```
static
{
    - - - - -
    - - - - -
    //creating type-1 jdbc driver class object
    JdbcOdbcDriver jdbcodbcdriver = new JdbcOdbcDriver();
    try
    {
        //registering type1 driver with DriverManager service
        Drivermanager.registerDriver (Jdbcodbcdriver)
    }
```

```
catch (SQLException sqlexception)
    {
      - - - - -
      - - - -
    }
}
```

NOTE: Almost every vendor supplied jdbc driver related jdbc driver class contains the above given static blocks. Registering ~~them~~ its respective jdbc driver with drivermanager service.

(Q) How many ways are there to register jdbc driver with Driver-Manager service?

(A) (1)

```
sun.jdbc.odbc.JdbcOdbcDriver obj = new sun.jdbc.odbc.JdbcOdbcDriver();
Drivermanager.registerDriver(obj);
```

The above code registers jdbc driver (same) twice with Drivermanager service.

(i) to static block of driver class when driver class object is created.

(ii) Explicitly when programmer calls Drivermanager.registerDriver().

It is not recommanded process.

(2)

```
sun.jdbc.odbc.JdbcOdbcDriver obj = new Sun.jdbc.odbc.JdbcOdbcDriver();
```

Here jdbc driver will be register Drivermanager service only once but programmer creates one explicit unused object for driver class. It is not recommanded approach.

(3)

```
DriverManager.registerDriver (new sun.jdbc.odbc.JdbcOdbcDriver());
```

→ Anonomous object

Limitations are same as approach no.1

(4)

```
class.forName ("sun.jdbc.odbc.JdbcOdbcDriver");
```

Static block of driver class registers jdbc driver with Driver-Manager service only once.

It is recommanded approach.

5) `Class.forName(args[0]);`

```
// ConnTest.java
import java.sql.*;
public class ConnTest
{
    public static void main(String args[])
    {
        //load jdbc driver class
        Class.forName(args[0]);
        //establish connection with d/b s/w
        Connection con = DriverManger.getConnection("jdbc:odbc:oradsn",
                                                     "scott","tiger");
    } //main
}//class
>javac ConnTest.java
>java ConnTest sun.jdbc.odbc.JdbcOdbcDriver
                 ↳ args[0] value
```

same as approach no. 4 but allows to pass Jdbc driver class name from outside the application dynamically at runtime. It is recommanded approach.

6) we can pass input value to java application as system property value by using java -D option. To read this value our application can use System.getProperty(-) method.

```
// ConnTest.java
import java.sql.*;
public class ConnTest.
{
    public static void main(String args[])
    {
        //load jdbc driver class Read system property value
        String cname = System.getProperty("my.driver");
        //load jdbc driver class
        Class.forName(cname);
        //establish connection with db s/w
        Connection con = DriverManager.getConnection("jdbc:odbc:oradsn",
                                                      "scott", "tiger");
```

```
    } //main
  } //class

>javac ConnTest.java
>java -Dmy.Driver = sun.jdbc.odbc.JdbcOdbcDriver ConnTest
```

↳ system property name and value

Recommanded to use same as approach no. 5.

(7) The DriverManager class attempts to load the driver classes automatically which are referenced in the system property called "jdbc.drivers"

```
//ConnTest.java
import java.sql.*;
public class ConnTest
  {
    public static void main(String args[]) throws Exception
      {
        //establish connection with db s/w
        Connection con = DriverManager.getConnection("jdbc:odbc:oradsn",
                                                      "scott","tiger");
      } //main
  } //class
>javac ConnTest.java
>~~java -Djdbc.odbc.JdbcOdbc~~
>java -Djdbc.drivers = sun.jdbc.odbc.JdbcOdbcDriver ConnTest
```

↳ fixed property name

The above process makes current machine remembering registered jdbc driver for long time.

(8) In the process of loading subclass the JVM automatically loads super class. In this process the static blocks of both subclass and superclass will be executed automatically. The below process is not recommended it makes our class not extending from other useful classes like Thread, Frame, Applet and etc.

```
// ConnTest.java
import java.sql.*;
public class ConnTest extends sun.jdbc.odbc.JdbcOdbcDriver
{
    public static void main(String args[]) throws Exception
    {
        //establish connection with DS s/w
        Connection con = DriverManager.getConnection(
```

: According to DBMS the sql queries are

(1) DML queries (insert, update, delete queries)

(2) DRL queries (select queries)

(3) DDL queries (create table, alter table and etc...)

(4) TCL queries (Commit, rollback, savepoint and etc...)

: According to jdbc persistance logic there are two types of sql queries

(1) select sql queries (DRL queries)

(2) Non-select sql queries (DDL, DML, TCL queries)

: select sql query execution returns bunch of records that are selected from database table, whereas non-select queries (like DML queries) return a numeric value representing no. of records that are effected because of query execution.

· our jdbc persistance logic uses executeUpdate() to send and execute non-select queries in database software. Similarly it uses executeQuery() method to send and ~~ex~~ execute select queries in db s/w

Prototype of executeQuery (-) method

public ResultSet executeQuery (String qry) throws SQLException.

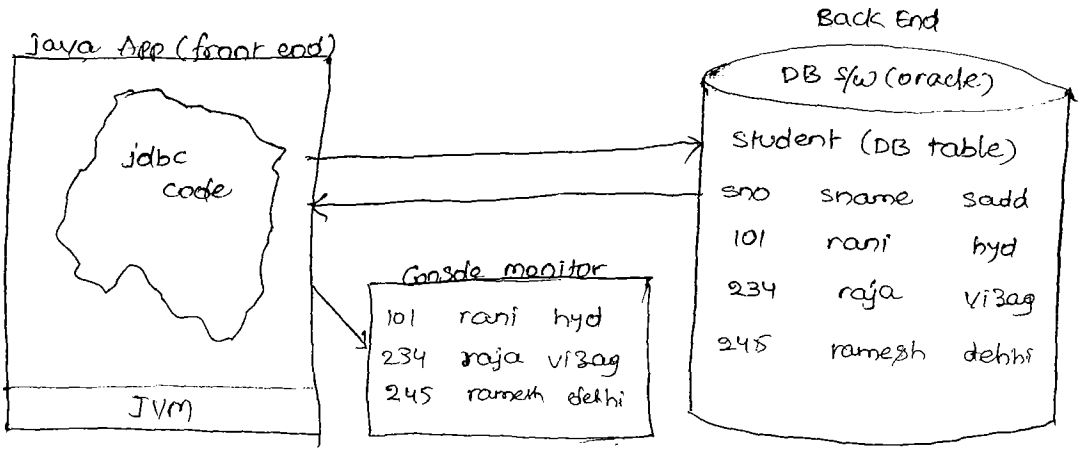
* executeQuery (String qry) method returns jdbc resultset object having set of records collected from database software after executing select sql query.

Prototype of executeUpdate (-) method

public int executeUpdate (String qry) throws SQLException

* This method returns a numeric value representing no. of records that are effected after executing non-select sql query in database software.
* The above given two methods are declared in java.sql.Statement Interface.
* since the end users like civil engineers, bank employees and etc.. can not work with database software directly by using sql queries. The expect GUI or CUI applications as front end to perform various operations on back end called d/b s/ws.
* while developing these front end applications in java environment we use jdbc support to interact with database software and to manipulate database data.

write a java application to read the records of database table from java application



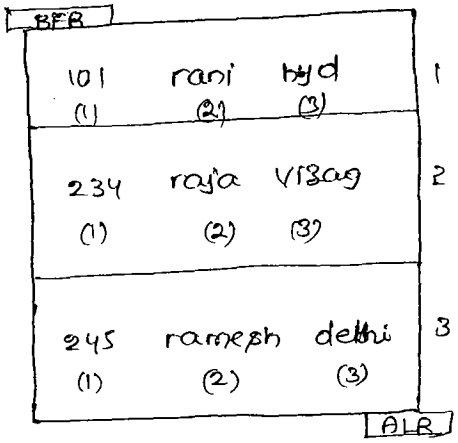
Step(1): Make sure that oradsn is available as dsn for odbc driver for oracle.

Step(2): keep student database table ready in database software (oracle) with records.

Step(3): Develop jdbc code based application using type1 driver as shown below.

```
// selectTest.java
import java.sql.*;
public class selectTest
{
    public static void main (String args[]) throws Exception
    {
        // load jdbc driver class
        Class.forName ("sun.jdbc.odbc.JdbcOdbcDriver");
        // establish connection with database software
        Connection con = DriverManager.getConnection ("jdbc:odbc:oradsn",
                                                      "scott","tiger");
        // create jdbc statement object
        Statement st = con.createStatement ();
        // send and execute sql query in database software and
          gather the results.
        ResultSet rs = st.executeQuery ("select * from student");
        // Process the result
        while (rs.next () == true)
        {
            system.out.println ( rs.getInt (1) + " " + rs.getString (2) +
                                 " " + rs.getString(3));
        } //while
        // close jdbc objects
        rs.close() ;
        st.close() ;
        con.close();
    }
}
```

rs (Resultset object)



* Every result set object contains two positions
  (1) BFR (Before First record)
  (2) ALR (After Last record)
* By default record pointer / cursor resides in BFR position.
* Records in result set and values in each record of result set will be stored having one based index.
* rs.next() method moves the record pointer from current position to next position. If the next position is valid position (record) then this method returns true. If the next position is ALR (or) BFR position then this method returns false.
* To read each record values from jdbc result set object we can use getXxx (-) methods, with column name (or) column index.

  getXxx (-)

  getString (-), getFloat(-), getInt(-), getDouble(-), getLong(-), getDate(-) and etc.

Prototype of getXxx (-) methods :

Public Yyyy getXxx (String columnname / int column index) throws SQLException.

* It is always recommanded to close the jdbc objects in the reverse order of their creation.
* If you don't close jdbc objects, at the end of their utilization then they will be closed automatically and abnormally at the end of the application's execution.
* The jdbc applications can not read uncommitted data from the database tables.
* Jdbc statement object means it is the object of a jdbc driver supplied java class implementing java.sql.StatementInterface directly or indirectly.
* Jdbc resultset object means (or) it is the object of a jdbc driver supplied java class implementing java.sql.resultset interface directly or indirectly.
* All jdbc objects of jdbc programming are objects of jdbc driver supplied java classes implementing various interfaces of jdbc API but programmer never specifies these class names in his jdbc programming because their names will be change based on the jdbc driver.
* When interface reference variable points to its implementation class object then reference variable ~~is~~ ~~obj~~ becomes object of implementation class with limitations.

```
interface ABC
  {
      Public void x();
  }
public class Demo implements ABC
  {
      Public void x()
        {
            sop("Demo: x()");
        }
      public void y()
        {   sop("Demo: y()");
        }
```

```
public static void main (String args[])
  {
    ABC ab = new Demo();
     ab.x(); // valid
     ab.y(); // invalid
    sop("class name of ab is" + ab.getClass());
  }
}
```

↳ gives Demo class

```
Statement st = con.createStatement();
```

* In the above statement a) create statement() is the method declared in java.sql.Connection interface (As the rule of jdbc specification).

b) The jdbc driver supplied java class that implements java.sql.Connection interface contains implementation for this createStatement() method (in type-1 driver that class name is sun.jdbc.odbc.JdbcOdbcConnection).

c) The definition of createStatement() method contains logic to create and return object of a java class (jdbc driver supplied) that implements java.sql.Statement interface (in type-1 driver that class name is sun.jdbc.odbc.JdbcOdbcStatement).

```
Connection con = DriverManager.getConnection(—, —, —);
```

* getConnection is the static method of DriverManager class. This method definition returns object of jdbc driver supplied java class implementing java.sql.Connection interface as jdbc connection object. (jdbc driver supplied class). (In type-1 driver class is sun.jdbc.odbc.JdbcOdbcConnection).

```
// SelectTest1.java
import java.sql.*;
import java.util.*;
```

```
public class Test1
{
    public static void main(String args[]) throws Exception
    {
        - - - - - -
        - - - - - -   } Logic to create jdbc connection
        - - - - - -                                object

        //read input values from key board.
        scanner sc = new Scanner(System.in);
        System.out.println("enter student no:");
        int no = sc.nextInt();
        // create jdbc statement object
        statement st = Con.create Statement();
        //send and execute sql query in db s/w and gather results
        Resultset rs = st.execute Query("select * from student
                                        where sno >= "+no);
        //Process the result (Resultset obj)
        while (rs.next != false)
        {
            s.o.P(rs.getInt(sno) + " " + rs.get String("sname") + " "
                                + rs.getString("sadd"));
        }
        // close jdbc objects
        rs.close();
        st.close();
        Con.close();
    } // main
} // class
```

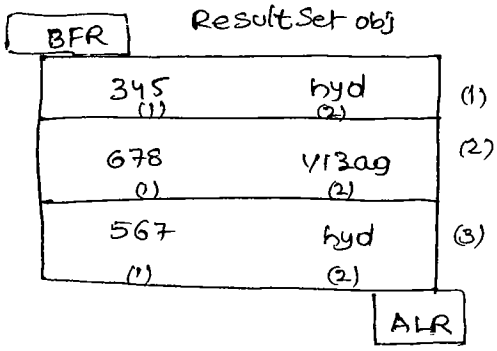
* while retrieving column values from the records of jdbc Resultset object use those column indexes in the order column values are stored in the records of jdbc Resultset object. Don't use those column indexes in the order columns are there in the database table.

Example code:

```
statement st = Con.createStatement();
ResultSet rs = st.executeQuery("select sno, sadd from student");
                                       ①    ②
// Process the resultset object
while (rs.next() != false)
{
    sop(rs.getInt(1) + " " + rs.getString(2));
}
```

↳ gives sno column values

↳ gives sadd column values



selecting specific column values from database table with multiple conditions

```
//read input values from keyboard
scanner sc = new Scanner(System.in);
sop("enter starting range of student no:");
int stno = sc.nextInt();
sop("enter ending ranage of student no:");
int endno = sc.nextInt();
//create jdbc statement obj
Statement st = Con.createStatement();
ResultSet rs = st.executeQuery("select sadd, sname from
                student where sno >=" + stno + " and sno <="
                + endno);
```

```
//process the result set object
while (rs.next()!=false)
{
    sop(rs.getstring(1) + " " + rs.getstring(2));
}
```

gives sadd values gives sname values

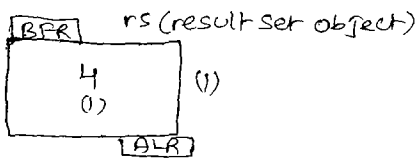
* while framing SQL queries with string input values the programmer needs to be little bit careful

Example Code:

```
//read input values from keyboard
scanner sc= new scanner (System.in);
sop(" Enter begining char or chars of student name to perform
      search operation");
String cond = sc.next();  // gives 'char:r
//Converting input value as required for sql condition
cond = "'"+ cond + "%'";  // r becomes 'r%'
//create jdbc statement object
Statement st = con.createstatement();
Resultset rs = st.executeQuery ("Select * from student where
                                  sname like "+cond);
//process the resultset object
while (rs.next()!=false)
{
    sop(rs.getstring(1) + " " + rs.getstring(2) + " " rs.getString(3));
}
```

* Aggrigate functions retrieve data from database table as not specific to any record of the table.

Example to send and execute sql aggrigate function based query



```
//create jdbc statement object
 Statement st = Con.create statement();
Resultset rs = new Resultset();
ResultSet rs = st.executeQuery("select count(*) from
                                        student");
//process the jdbc Resultset object
 if (rs.next())
   {  int cnt = rs.getInt(1);
      // int cnt = rs.getInt("Count(*)");
      sop("Count of records:" + cnt);
   }
```

* once ResultSet object is closed we can not retrieve data from ResultSet object.
* when jdbc objects are closed by calling close() method all the database resources that are associated with these objects will be released immediately.
* To solve unsupported ClassVersion Exception problem that raises after installing oracle software make sure that java_home\bin directory is kept as first value of path environment variable.

my computer → properties → advanced tab → env variables

variable name : path

value : D:\java\jdk 1.6.0_11\bin ; D:\oracle\product\10.2.0\db_1\bin ; .

↳ java_home

↳ oracle_home

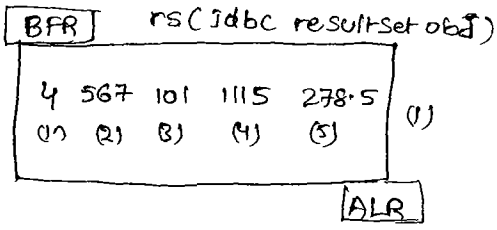
. → current directory

ok → ok → ok

<u>Example code to send and execute multiple aggrigate functions based select sql query.</u>

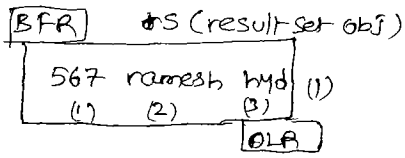
```
//create jdbc statement object
 Statement st = Con.create statement();
```

```
ResultSet rs = st.executeQuery ("select Count(*), max(sno),
                                 min(sno), avg(sno), sum(sno) from student");
//process the jdbc resultset object
if (rs.next())
{
  sop("count of records:" + rs.getInt(1));
  sop("max val of sno" + rs.getInt(2));
  sop("min val of sno" + rs.getInt(3));
  sop("sum of so column"+ rs.getInt(4));
  sop("avg of sno col values:" + rs.getFloat(5));
}
```



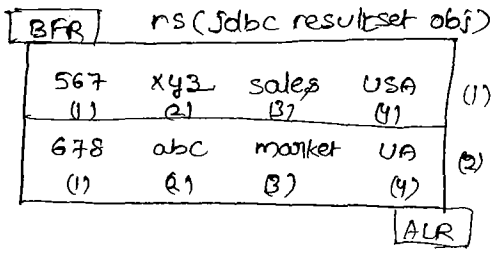
<u>Example code to execute sub query</u> (Gives that student record who is having max sno value)

```
//create jdbc statement object
Statement st = con.createStatement();
ResultSet rs = st.executeQuery ("select * from student where
                                 sno = (select max(sno) from student)");
//process the jdbc resultset object
if (rs.next())
  {
    sop( rs.nextInt(1) + " " + rs.getString(2)+" " + rs.getString(3));
  }
else
  sop ("no record found");
```



Example code that selects the data from multiple columns of multiple database tables.

```
//create jdbc statement object
statement st = con.create statement();
ResultSet rs = st.executeQuery ("select empno, ename, dname,
                                 loc from emp, dept");
//process the resultset object
while (rs.next())
  {
    sop(rs.getInt("empno") + " " + rs.getString("ename") + " "
        + rs.getString("dname") + " " + rs.getString("loc"));
  }
```



write a java application by using jdbc driver type-1 to delete the records of student table based the given student no.

```
//DeleteTest.java (use executeUpdate() method)
import java.sql.*;
import java.util.*;
public class DeleteTest
  {
    public static void main (String args[]) throws Exception
      {
        //read input values from keyboard
        Scanner sc = new Scanner(System.in);
        int no = sc.nextInt();
        //register jdbc driver with DriverManager service
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

```
//establish connection with DB software
  Connection con = DriverManager.getConnection("jdbc:odbc:oradsn",
                                                "scott", "tiger");
//create jdbc statement object
  Statement st = con.createStatement();
//send and execute query in db software
  int res = st.executeUpdate("delete from student where
                                          sno = "+sno);
//process the result
  if (res==0)
      system.out.println("no record found to delete");
  else
      system.out.println(res+" no. of records are deleted");
//close jdbc objects
  st.close();
  con.close();
}
}
```

* Java application sends and execute non-select queries on database software by enabling autoCommit mode. we can disable this autoCommit mode from our Java application by using con.setAutoCommit(false) method.
* After disableing autoCommit mode if Java application sends and executes non select sql queries in database software then use con.commit() method to commit the query execution results (or) use con.rollback() method to rollback the query execution results.

<u>Example Code</u>

```
// disable auto Commit mode
    con.setAutoCommit(false);
//create jdbc statement object
```

```
statement st = con.create statement();
//send and execute sql query in db s/w
int res = st.executeUpdate("delete from student where sno="
                                                    +no);
//rollback or Commit
con.rollback(); //or con.Commit();
```

* The above code demonistrates TCL operations.

(Q) When all other SQL queries are going to database software by using jdbc statement object why commit and rollback instructions should be given to database software by using jdbc connection object?

(A) statement object deals with single query execution at a time. But commit and rollback operations should be performed with respect to multiple sql queries execution so we use jdbc connection object which represents whole database to pass the commit and rollback instructions.

The SQL queries whose execution can generate results should be given to database software by using jdbc statement object. The instructions whose execution do not return results like commit, rollback should be given to database software using jdbc connection object.

Write a java application to insert the record into database table by gathering values from keyboard.

```
// insertTest.java
import java.sql.*;
import java.util.*;
public class insertTest
  {
    public static void main(String args[])
      {
        //read input values from keyboard
          scanner sc = new scanner(system.in);
```

```
System.out.println("Enter student no.");
int no = sc.nextInt();
System.out.println("Enter student name");
String name = sc.next();  // raja
System.out.println("Enter student address");
String addr = sc.next();  // hyd
// Convert input values as required for sql query
name = "'" + name + "'";  // raja becomes 'raja'
addr = "'" + addr + "'";  // hyd becomes 'hyd'
// create jdbc connection object
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection con = DriverManager.getConnection("jdbc:odbc:oradsn",
                                              "scott", "tiger");
// create jdbc statement object
Statement st = con.createStatement();
// frame example sql query with direct values
// insert into student values (101, 'raja', 'hyd');
// frame query with variables
String query = "insert into student values("+no+","+name+
                                              ","+addr+")";
System.out.println(query);
// send and execute sql query in db s/w
int result = st.executeUpdate(query);  // non-select
// process results
if(result==0)
    System.out.println("Record insertion is failed");
else
    System.out.println("Record insertion is done");
```

```
try
{
  // read input values from keyboard
   sc = new Scanner(System.in);
   int no=0;
   String name =null;
   String address=null;
   if (sc! =null)
     {
       System.out.println("Enter existing student number");
       no = sc.nextInt();
       System.out.println ("Enter new name");
       name = sc.next();
       system.out.Println (" Enter new address ");
       address = sc.next();
     } // if
  //create Jdbc con obj
     class.forName ("sun.jdbc.odbc.JdbcOdbcDriver");
     Con= DriverManager.getConnection ("jdbc:odbc:oradsn", "scott", "tiger");
     // create statement object
     if (con ! =null)
         st = Con.create Statement();
  // frame example query with direct values
  // String query ="update student set sname='maharaja', sadd='vizag'
                                        where sno=101;
   String query="update student set sname='"+name+"', sadd='"+address
                                        +"' where sno="+no;
   system.out.Println(query);
   //send and execute the query in database s/w. int res=0;
   if (res ! =null)
   res = st.executeUpdate(query);
   // process results
    if ( res==0)
        system.out.Println("no record found");
```

```
// close jdbc objects
    st.close();
    Con.close();
  } //main
} // class
```

Industry coding standards

* Don't declare the exception to be thrown using throws statement. Always catch and handle the exceptions with try and multiple catch blocks.
* write comment for every line.
* To avoid nullPointerException call method on java object after ensuring that object is not holding null value. Those jdbc objects (or) any other stream objects in finally block because that block executes irrespective of the exception that is raised and gives the gaurentee of closing stream objects in any situation.

Write a java application to update student details based on student no.

```
// updateTest.java (with coding standards)
  /* App to update student details
   * version : 1.0
   * author : Team-S */
  import java.sql.*;
  import java.util.*;
  public class UpdateTest
  {
    public static void main(String args[])
    {
      Connection Con = null;
      Statement st = null;
      Scanner sc = null;
```

```
    else
        system.out.println(res + "no. of records are updated");
  } // try
  catch (ClassNotFoundException cnf) // handles known exception
    {
      cnf.printStackTrace();
    }
  catch (SQLException se)      //handles known exception
    {
      se.printStackTrace();
    }
  catch (Exception e)      //handles unknown exception
    {
      e.printStackTrace();
    }
  finally
    {
      //close jdbc objects
      try
        {
          if (st != null)
              st.close();
        }
      catch (SQLException se)
        {
          se.printStackTrace();
        }
      try
        {
          if (Con != null)
              Con.close()
        }
      catch (SQLException se)
        {
          se.printStackTrace();
        }
    } // finally
  } // main
} // class
```

* In java local variables do not have initial values/default values, so when they are declared they must be assigned with initial values based on their data types.
* create table, drop table, alter table etc queries are called as DDL queries. In real time programers are not responsible to do these DDL operations through queries execution.
* Every company contains DBT (Database Team) and that team is responsible to design the tables by satisfying all the six forms of normalization rules.

Net Beans :

type : IDE for java environment

Version : 6.7.1 (Compatable with jdk 1.6)

vendor : sun microsystems (oracle corporation)

open source software (free software)

To download software : www.netbeans.org

for help : www.netbeans.org

Procedure to develop java application by using netbeans IDE to drop database table

Step(1) : Launch netbeans IDE

Step(2) : create java project from netbeans IDE

File menu → new project → java → java application → next → project name → deselect create main class → select main project → finish.

Step(3) : Add java class to the project

Right click on the project → new → java class → class name (DropTest) → finish

psvm + tab space

For various short cuts of netbeans IDE refer help menu observe keyboard shortcuts.

Ctrl + shift + i → package import

```
//Droptest.java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
import java.sql.Scanner;
public class Droptest
{
  public static void main(String args[]) throws Exception
  {
    //read input values from keyboard
     scanner sc=new scanner(system.in);
     system.out.println("Enter table name to drop");
     string tabname = sc.next();
    //create jdbc connection object
     class.forName("jdbc.sun.jdbc.odbc.JdbcOdbcDriver");
     Connection con= DriverManager.getConnection("jdbc:odbc:oradsn",
                                                  "scott", "tiger");
    //create statement object
     statement st = con.create statement();
    //send and execute DDL query to db s/w
     int res = st.executeupdate("drop table "+tabname);
    //process the results
     if (res ==0)
          system.out.println(tabname + " is not found");
     else
          system.out.println("table dropped");
    // close jdbc objects
       st.close();
       con.close();
    } //main
  } //class
```

Step(4): Run the application

Right click in the source code of droptest.java → run file → click on it

Procedure to set command line argument values to java application of netbeans IDE project

Rightclick on project → run → main class droptest → arguments

val1 val2 ---
↓ ↘
args[0] args[1]

How can we use single method call to execute both select and non-select querres?

1) use execute() method call on jdbc statement object for this operation. This method call returns true when select query is executed and this method call returns false when non-select query is executed.

Prototype: public boolean execute (String query) throws SQLException.

This method call makes the programmer to gather the query execution results seperately through other method calls.

if select query is executed (return value is true)

Resultset rs = st.getResultSet(); → gathers the result

if non-select query is executed (return value is false)

int cnt = st.getUpdateCount();
↓
numeric value representing the no. of records that are effected due to non-select sql query execution.

* working with execute() method is not recommanded process so use executeQuery() method for executing select query and

Use executeUpdate() method for executing non-select query.

Example on execute method

```
// create statement object
    Statement st = con.createStatement();
//send and execute query
//boolean flag = st.execute("select * from student");
boolean flag = st.execute("delete from student where sno=100");
if (flag==true)
  {
    System.out.println("select qry executed");
    Resultset rs= st.getResultSet();
    st.executeQuery("select * from student");
    while (rs.next())
      {
        system.out.println (rs.getInt(1)+" "+rs.getString(2)+" "
                                          +rs.getString(3));
      }
    rs.close();
  }
else
  {
    system.out.println("non-select query executed");
    int cnt = st.getUpdateCount();
    System.out.println("no. of records that are effected"+ cnt);
  }
```

* Any vendor can develop JDBC driver for any database software by using one of the following four methodologies (or) mechanisms. They are

(1) Type-1 (Jdbc-odbc bridge)

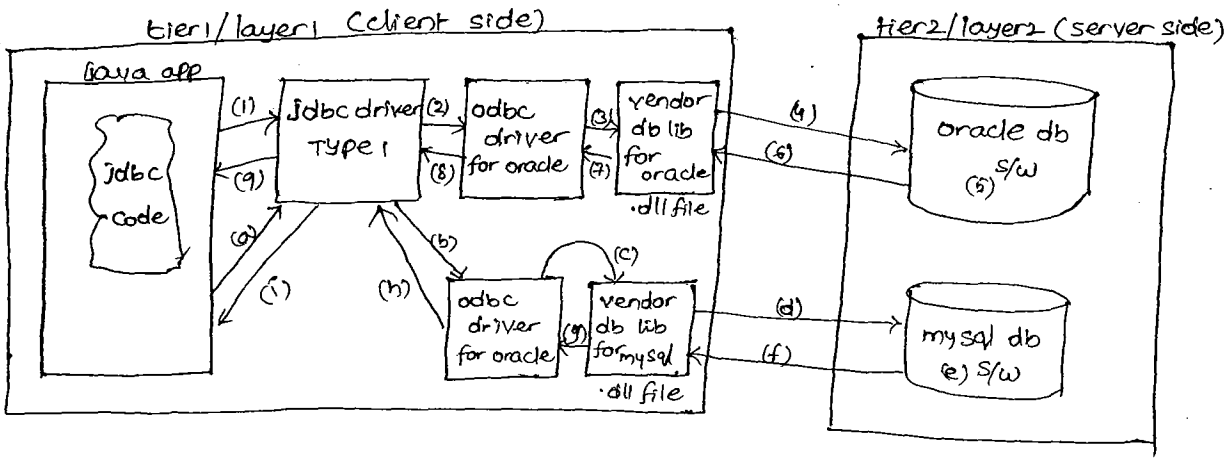
(2) Type-2 (Native-API/ partly java Driver)

(3) Type-3 (Net protocol /All Java Driver)

(4) Type-4 (Native-protocol / all-java Driver)

* Every database software gives one library called vendor db library having code and logic to locate and interact with db software. Jdbc drivers and odbc drivers take the support of these vendor db libraries to locate and communicate with their respective database softwares.
* Generally this vendor database library comes as .dll file. In case of oracle this file name is oci.dll (comes with oracle s/w installation).
* oci - oracle connection interface.

Type1 mechanism/ methodology/Architecture



* In client server environment in order to work with centralized oracle database software the server oracle software will be installed on the server machine of the network and client oracle software will be installed in the remaining machines of the network. Both client and server oracle software installations give vendor db library for oracle.

Advantages:

* Type-1 driver is not specific to any database software. It uses an appropriate database software specific odbc driver to interact with database softwares. only sun micro systems is supplying jdbc driver based on type-1 mechanism.
* can be used to interact with any db software.

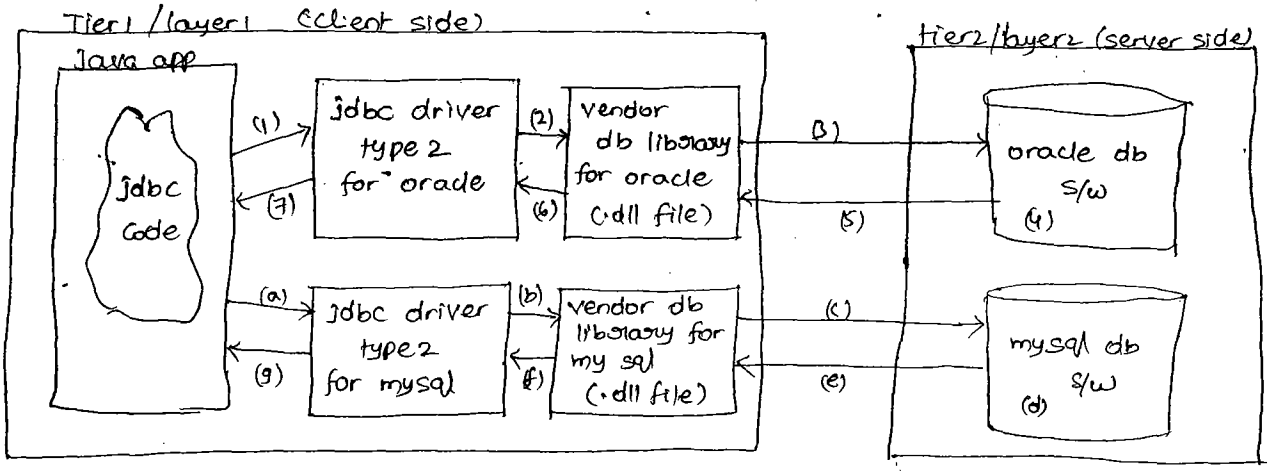
* Comes as built-in driver of jdk software so no need of arranging this driver seperately.
* Since odbc drivers are available almost for all database softwares we can use type-1 driver to interact with all the database softwares.

Disadvantages

* Because of multiple conversions the performance is poor.
* Since vendor db library and odbc driver are required at client side
  a) Type-1 driver is not suitable for internet based applications.
  b) Type-1 driver is not suitable for untrusted applet to database software communication.
* Type-1 driver is not suitable in large scale projects.
* Trusted applet can interact with files on filesystem to perform read and write operations whereas untrusted applets can not do this work.

Type-2 mechanism :

Type2 mechanism/ methodology/ Architecture



* Type-2 jdbc driver uses native code to communicate with vendor db library. Except type1 mechanism no other mechanism takes the support of odbc drivers.
* Type-2 jdbc driver takes the support of db vendor db library directly to interact with database software.

Advantages

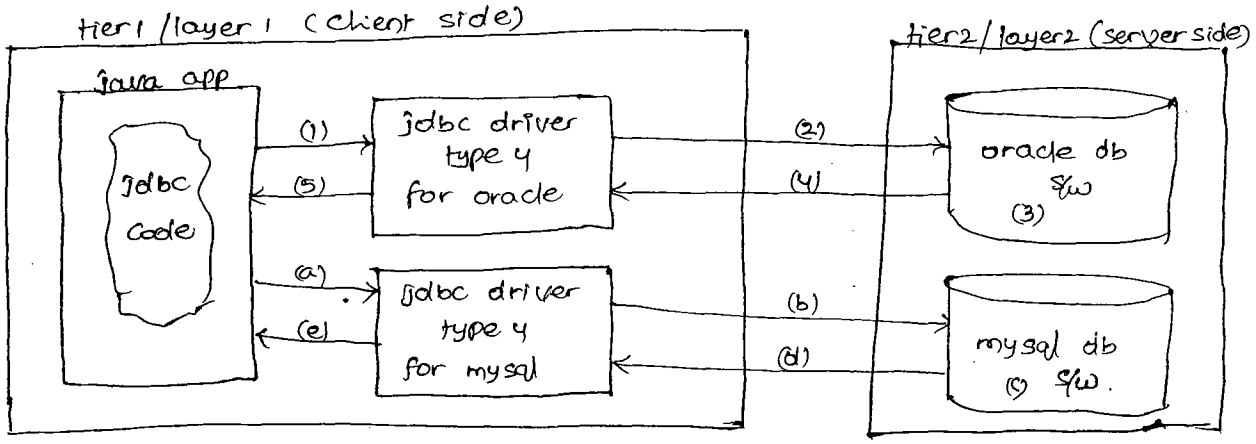
* No need of working with odbc drivers.
* use significantly better performance compare to type1 driver.

Disadvantages

* For every database software we need one seperate jdbc driver type-2.
* since ~~odbc driver is not~~ vendor db library is required at client side we can not use type-2 driver in internet based applications and for untrusted applet to database software communication.
* Not suitable to large scale applications.
* programmer needs to arrange type-2 jdbc drivers seperately

Type-4 mechanism:

Type 4 mechanism/ methodology/ Architecture



* Type-4 driver is designed to interact with database softwares directly without taking support of vendor db library and odbc driver.

Advantages

* gives better performance compare to type-1 and type-2 drivers.
* No need of working with odbc drivers and vendor db libraries.
* This driver will be developed purely in java environment so it exibits platform independency and architecture neutrality.

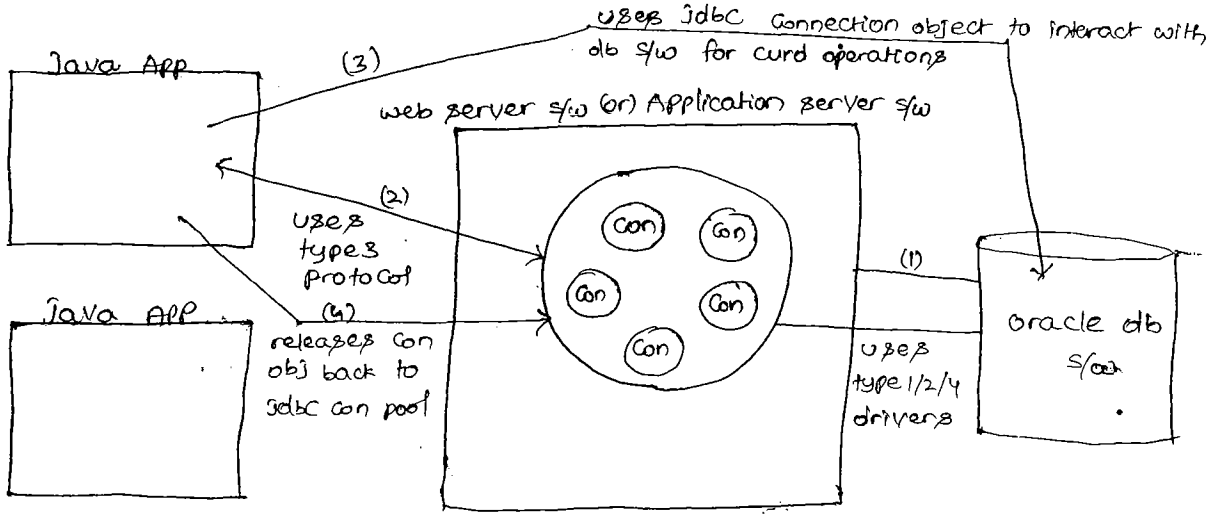
* This driver can be downloaded to client machine dynamically from internet network.
* suitable for internet programming based applications and for untrusted applet to database software communication.
* suitable for large scale projects.

Disadvantages

* For every database software we need one seperate jdbc driver type-4.
* Programmer must arrange jdbc drivers seperately because they don't come along with jdk software.

Type 3 mechanism

* Type 3 is not real jdbc driver it is given as protocol having set of rules to get communication between java application and jdbc connection pool.
* Jdbc connection pool is a factory that contains set of readyly available jdbc connection objects before actually being used.
* The advantage of jdbc connection pooling is we can use less no. of jdbc connection objects to make more java applications interacting with database softwares.
* Jdbc connection pools will be created and managed by webserver, application server softwares like tomcat, weblogic and etc.



* All jdbc connection objects of jdbc connection pool represents connectivity with same and single database software.

Ex: Jdbc connection pool for oracle means all jdbc connection objects in that pool represents connectivity with same database software (oracle).

with respect to diagram

(1) webserver (or) application server uses type 1 (or) type 2 (or) type 4 driver to interact with database software and to create connection objects in connection pool.

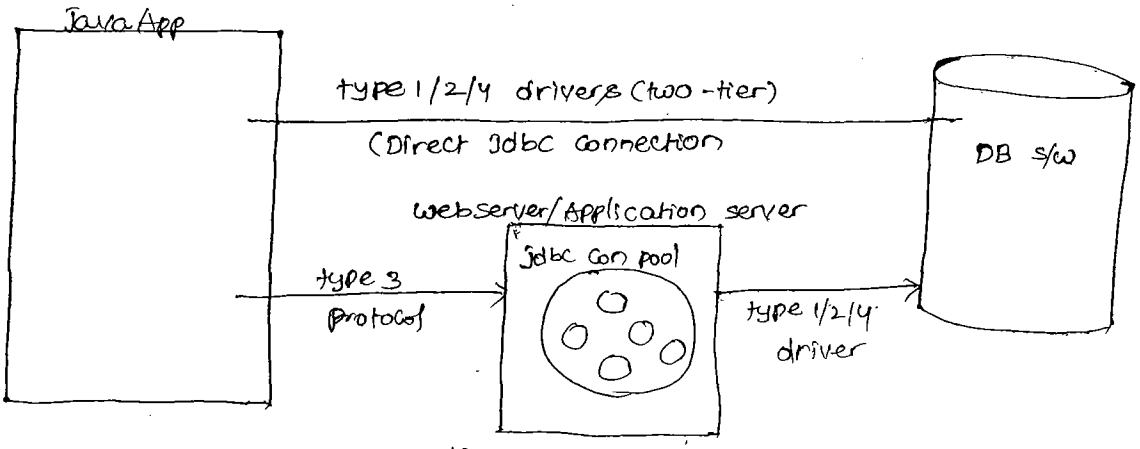
(2) Java application uses type 3 protocol to get one connection object from connection pool.

(3) Java application uses that connection object to create other jdbc objects and to perform CURD operations on database.

(4) Java application releases the connection object back to connection pool once task is completed. This connection object now becomes free to give service to other clients.

: while working with jdbc connection pool the programmer is not responsible to create, manage and destroy jdbc connection objects from Java application. All these operations will be taken care by jdbc connection pool.

summary diagram of jdbc drivers (all)



3 tier environment (pooled jdbc connection)

* The jdbc connection that is created by the programmer directly and manually is called as direct jdbc connection object.

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection Con = DriverManager.getConnection(--,--,--);
```

↓ Direct jdbc connection object.

* The jdbc connection object that is collected from jdbc connection pool is called as pooled connection object.

↔ For more information about jdbc drivers and architecture refer page nos 10-17 of booklet.

* Type 3 is given to work with jdbc connection pool environment.

<u>what is the jdbc driver that you have utilized in your project?</u>

(A) If project is small scale application like two-tier application (or) office automation application then use jdbc driver type-4. If project is medium scale (or) large scale project like three-tier application like web application, bank application and etc.. then use type 3 with type-4 jdbc drivers. Here type-4 driver will be used to create connection objects in connection and type-3 protocol will be used to get connection objects from connection pool.

<u>what is the difference between path and class path?</u>

(A) <u>Path</u>: In order to make .cmd, .bat and .exe files of one directory executable from any location of computer then add the address location of these files to path environment variable.

* .bat file combines set of DOS commands or windows commands into single unit to execute them in sequence.

<u>Problem</u>:

```
e:\abc\run.bat
date
time
dir
```

e:\abc> run (success)

e:\abc> run.bat (success)

e:\> run (fails)

run is not an external or internal command

d:\demo> run (fails)

run is not an internal or external command.

<u>Solution</u>: work with path environment variables

Take the address location of run.bat file i.e, e:\abc folder and add that to path environment variable.

mycomputer → properties → Advanced tab → env variables →

variable name : PATH

value : e:\abc; <other existing values>;.

→ok → ok → ok

e:\abc> run (success)

e:\> run (success)

d:\demo> run (success)

c:\> run (success)

* All our java tools like javac, java, javap are given as .exe files in java_Home\bin directory so to make them executing from any folder of your computer then add java_Home\bin directory to path environment variable.
* Don't set environment variable values (like path and classpath) from command prompt. Always use mycomputer properties related environment variables.
* multiple values added to environment variable must be seperated with ";" symbol. we should not use this ";" symbol as beginner of the first value as the terminator of last value.

* When new values are added to my Computer environment variable they will be reflected only in the new Command prompts that are opened after adding new values.

Class path:

* When Java application uses new APIs ~~and~~ (or) third party apis (other than Jdk APIs) then the new APIs, third parties APIs related directories are Jar files must be added to class path environment variable to make Java applications, Java tools recognising that new API, third party API.
* classpath is purely related to Java environment.
* API in Java language is nothing but set of classes and interfaces, which come in form of packages.

Understanding Problem:

E:\Demo\ABC

Test.Java

```
Package P1;
Public class Test
{
    Public void m1()
    {
        System.out.println("Test:m1()");
    }
}
```

e:\Demo\ABC > Javac -d . Test.Java

gives Test.class in P1 Package.

```
E:\Demo
    |---> ABC
            |---> Test.Java
            |---> P1
                    |-> Test.class
```

D:\work
|—> mainApp.java

MainApp.java

```
import P1.Test;
public class mainApp
{
  public static void main (String args[])
  {
    Test t = new Test();
    t.m1();
  } // main
} // class
```

D:\work> javac MainApp.java (x -> failure)

error: can not find symbols P1, Test, m1()

Applying solution : work with class path

Add the directory location where "P1" package is available (e:\demo\ABC) to CLASSPATH environment variables variable.

my computer -> properties -> advanced tab -> env variables -> new ->

variable name: CLASSPATH

value : e:\demo\ABC;.

-> ok -> ok -> ok

(or)

my computer -> properties -> advanced tab -> env variables -> edit ->

variable name: CLASSPATH

value : e:\demo\ABC; <other existing values>;.

-> ok -> ok -> ok

D:\work> javac MainApp.java (success)

gives mainApp.class

D:\work> java MainAPP

-> gives output.

* To make java tools running any where in your computer then work with path environment variable. To make these java tools to recognising and utilizing third party APIs and new APIs then work with CLASSPATH environment variable.
* Generally programmers and third party vendors keep their APIs in jar files and they will supply them to others. These other programmers uses these new APIs and third party APIs by adding their jar files to classpath.

E:\Demo\ABC> jar cf sathya.jar.

c → create jar file
f → use given name as jar file name.

* To add jar file to classpath

myComputer → properties → Advanced tab → env variables → edit →

variable name: CLASSPATH

value : e:\Demo\ABC\sathya.jar; <other existing values>;.

→ ok → ok → ok

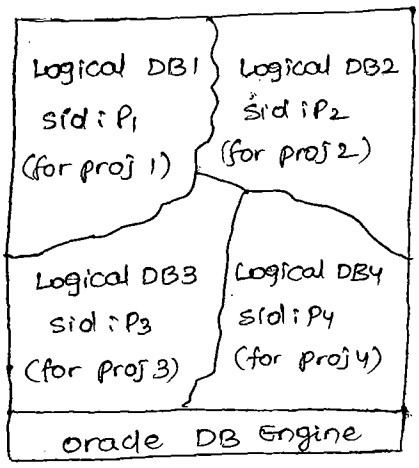
* Oracle corporation supplies its type 4 mechanism, type 2 mechanism related jdbc drivers in the form of "ojdbc14.jar" file along with oracle software installation.
* The location of ojdbc14.jar file is

oracle_home\product\10.2.0\db-1\jdbc\lib (oracle 10g)

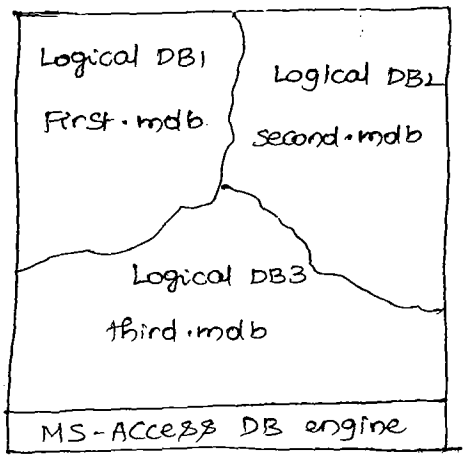
* Jar file can represent new APIs (or) third party APIs.
* Jar file can represent jdbc drivers.
* Jar file can represent a package webapplication (web site).
* Jar file can represent an EJB component.
* Jar file can combine multiple resources into single file like zip file.
* In every physically installed database software we can create multiple logical partitions as logical databases.
* Every logical database will be identified with its database name.

* In oracle database software database name of logical database is called as sid.

Ex: If multiple projects of a Company are utilizing same oracle database software then the oracle database software should be installed only once in Common machine of a Company and multiple logical databases will be created in that oracle database software for multiple projects on one per project basis.



(Physical oracle log DB s/w)



(physical MS-Access DB s/w)

* In MS-Access each .mdb file is one logical database and it will be identified with its file name.
* creating new logical databases in physical database software is the responsibility of DBA.
* In each logical database multiple DB tables, DB users and etc... can be there.
* when oracle software is installed it will give one default logical database, to know sid of this logical database:

approach (1):

```
SQL> select * from GLOBAL_NAME;

ORCL.REGRESS.---.--
```

↳ sid of current logical database (default).

approach (2): open oracle_home\product\10.2.0\db_1\network\admin\tnsnames.ora

file and look for service_name = orcl entry.
↳ sid of logical database.

approach (3):

Control panel → performance and maintance → adminstrative tools → services → Look for oracle Service ORCL
↳ sid of logical database.

* If logical database of physical database software is allowing multiple users or client applications concurrently to manipulate the data then it is called as multiuser database software. If only one user (or) client application is allowed to perform the above said operation then it is called single user database software.

Examples of multiuser database softwares

oracle, sybase, mysql, Postgresql and etc...

Examples of singleuser database softwares

MS-Access, foxpro, foxbase and etc...

Working with jar Command:

```
E:\temp
    |--> unit1
            |--> a.txt
            |--> b.txt
```

to create jar file

e:\temp\unit1> jar cf sathya.jar .

c → create jar file
f → with specified name as jar file name.

to see the table of content of jar file

e:\temp\unit1> jar tf sathya.jar

t → table of content in jar file.

: Every jar file contains one manifest.mf file in META-INF folder. This manifest file contain description about various resources of jar file.

to add new file to existing jar file

e:\temp
|→ unit
|→ c.txt (create new file)

e:\temp\unit> jar uf sathya.jar c.txt

Adds c.txt file to sathya.jar file.

u → updates the existing jar file

to extract files from sathya.jar file

to extract specific file

e:\temp\unit1> jar xf sathya.jar a.txt

to extract all files from jar file

e:\temp\unit1> jar xf sathya.jar

x → Extraction of jar file

: we can view and extract the content of jar file either by using winzip tool (or) winrar tool.

: The oracle corporation supplied type 2 mechanism based Jdbc driver for oracle is called as "oracle oci" driver.

: The oracle corporation supplied type 4 mechanism based Jdbc driver for oracle is called as "oracle thin" driver.

: Every software installed in the computer resides in a logical position called software port and every software port will be identified with its port number.

· Total 65535 software ports can be there in a single computer. In that 1 to 1024 are reserved for operating system services.

oracle software → default port no: 1521

Tomcat software → default port no: 8080

weblogic software → default port no: 7001

Internet Explored → default port no: 1078

* socket no is: (IP address / host name of machine) + (port no of software service)
* Port no. of oracle software is: 1521
* socket no. of oracle db software when installed on a machine having ip address 180.45.67.4

$\boxed{180.45.67.4 + 1521}$

socket no. of oracle db s/w.

* To communicate with any software installed on any computer we must know its socket number.

<u>Details of oracle corporation supplied type 2 mechanism based jdbc driver for oracle</u>

<u>Vendor</u> : oracle corporation

Commercial jdbc driver

<u>mechanism</u> : type 2 mechanism

<u>technical name</u>: oracle <u>oci</u> driver
↓
oracle Connection Interface

<u>jdbc driver class</u> : <u>oracle.jdbc.driver</u>. <u>OracleDriver</u>
↓ package name ↓ class implementing java.sql.Driver Interface

<u>Database url</u> : <u>jdbc:oracle</u> : <u>oci8</u> : @ <u><sid></u>
↓ Protocol ↓ subname ↓ logical database name

<u>jar file representing this oracle oci driver</u>

oracle 8i → classes111.jar or classes12.jar

oracle 9i → classes1111.jar (or) classes12.jar (or) ojdbc14.jar

oracle 10g → classes12.jar (or) ojdbc14.jar

oracle 11g → ojdbc6.jar

Procedure to develop java application by using oracle oci driver

step(1): Gather details about oci driver as shown above

step(2): Develop java application by using oracle oci driver as shown below

```
//selectTest.java
import java.sql.*;
import java.util.*;
public class Test
{
    public static void main(String args[]) throws Exception
    {
        //load jdbc driver class
        Class.forName("oracle.jdbc.driver.OracleDriver");
        //establish connection with DB s/w
        Connection con = DriverManager.getConnection("jdbc:oracle:oci8:
                                                      @ordl", "scott", "tiger");
                                                        ↓
                                                       sid
        //create jdbc statement object
        Statement st = Con.createStatement();
        //send and execute sql query in db s/w and gather the values
        ResultSet rs = st.executeQuery("select * from student");
        //process the result-set object
        while (rs.next())
        {
            System.out.println(rs.getInt(1) + " " + rs.getString(2) + " "
                                + rs.getString(3));
        }
        //close jdbc objects
        rs.close();
        st.close();
        con.close();
    }
}
// javac selectTest.java
// java selectTest
```

Step(3): Add ojdbc14.jar file or other related equivalent jar files to class path environment variable.

my Computer → properties → advanced tab → env variables → edit

variable name: CLASSPATH
value: oracle\product\10.2.0\db_1\jdbc\lib\ojdbc14.jar;
<other existing values>;.

→ok →ok →ok

Step(4): Compile and execute the application.

> javac selectTest.java
> java selectTest

* we can write jdbc code and we can create jdbc objects like statement object, resultset object and etc in a common way irrespective of the jdbc driver that we are using.
* As of now sun micro system is not supplying type2 and type4 mechanism based drivers so the programmer must gather them from database vendors (or) third party vendors.

Oracle corporation supplied type4 mechanism based jdbc driver for oracle

Technical name: oracle thin driver
mechanism: type 4
Vendor: oracle corp
Commercial jdbc driver
Jdbc driver class: oracle.jdbc.driver.OracleDriver
(or)
oracle.jdbc.OracleDriver.
Jdbc url /db url: jdbc:oracle:thin:@<hostname/ip address>:<port no>
jdbc → protocol; thin → subname; <hostname/ip address> → of computer where oracle db s/w is installed; <sid> → logical db; <port no> → of oracle db s/w (1521)
jar files that represents this driver: same as oracle oci driver

why the db url of type 4 jdbc drivers is quite lengthy when compare to the db url of type 1, type 2 drivers?

) Type1 driver is designed to talk with database software through odbc driver so its db url just need to locate odbc driver (jdbc: odbc: <dsn>)

once type2 driver locates vendor db library of client side that db library takes care of communicating with database software so its db url just need to locate vendor db library (jdbc:oracle: oci8:@ <sid>).

Type 4 drivers needs to locate and communicate with database softwares directly without taking support of other resources of client side, so its url contains multiple details. (jdbc:oracle:thin:@ <ipaddress /hostname>: <portno>: <sid>).

Procedure to work with oracle thin driver in our java applications.

NOTE: To refer a computer being from current machine we use the word called local host.

step(1): Gather the details of oracle thin driver as shown above.

step(2): Develop java application by using oracle thin driver.

//select Test.java

same as previous class application but the following db url to create jdbc connection object.

```
//establish connection with DB s/w
Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:
                                                1521:orcl","scott","tiger");
```

step(3): Add ojdbc14.jar file or other equivalent jar files to class path.

same as previous class.

step(4): Compile and execute the java application.

```
>javac selectTest.java
>java selectTest
```

* The values, jar files added to my Computer environment variable class path are not visible to the Projects created in IDE softwares. That means we need to add jar files to the libraries of IDE project seperately.

* If java application of the netbeans IDE project uses oracle thin driver then the oracle thin driver related ojdbc14.jar must be added to the library folder of the project (Right click on libraries folder → add jar → Browse and select ojdbc14.jar).

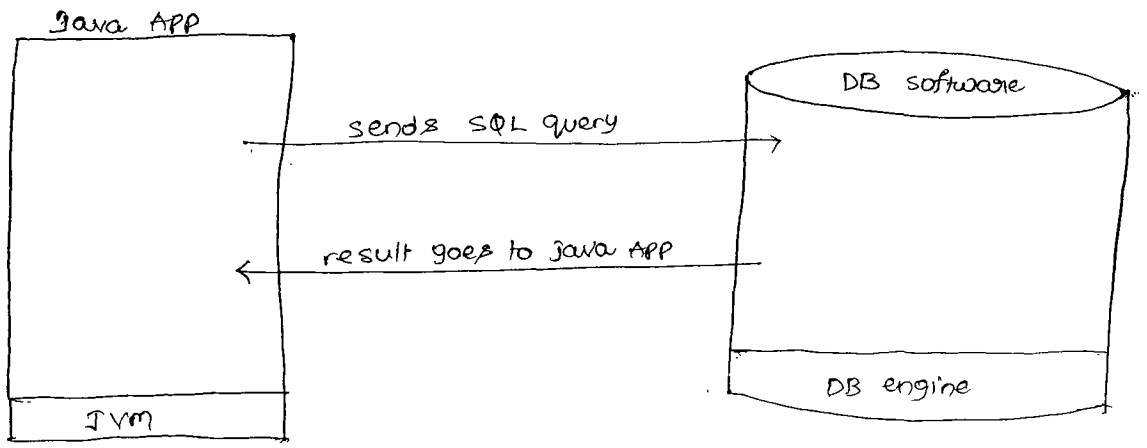
```
java.sql.Statement
        ↑ extends
java.sql.PreparedStatement
        ↑ extends
java.sql.CallableStatement
```

* There are 3 types of Jdbc statement objects which can be used in our java application to send and execute sql queries in database software

(1) Simple statement object (object of a java class that implements java.sql.Statement interface)

(2) PreparedStatement object (object of a java class that implements java.sql.PreparedStatement interface)

(3) Callable statement object (object of a java class that implements java.sql.CallableStatement interface)



* The database engine of database software performs following operations on SQL query that comes from client application.

(1) parse (2) Execute (3) fetch.

* In parse operation the syntax of the query will be verified by splitting the query into multiple tokens.
* In execute operation the parsed query will be executed and its output will be stored in the buffer.
* In fetch operation results from the buffer will be gathered and results will be delivered to client application.

<u>Limitations with simple statement object</u> :

* Framing sql query with variable for simple statement object is complex process.
* If simple statement is used to execute same query for multiple time with same or different values then
  a) Same query goes to database software for multiple times.
  b) Same query will be parsed in database software for multiple times.
  c) Same query will be executed in database software for multiple times with same (or) different values.
  d) The same query output will be fetched out for multiple times.
* In that above operations (a) and (b) are unnecessary operations for multiple times on single query. But they can not be avoided while working with simple statement object. This makes database software to perform unneccessary operations which may degrade the performance of the application.
* To overcome above limitations use precompiled sql query with the support of prepared statement object.

<u>NOTE</u>: The limitation (2) talks about the simple statement can't execute same query multiple times in db s/w by just sending and parsing that query only for one time in database software.

* Simple statement object sends raw sql query from java application to database software every time so the query goes through parsing, execution and fetch operations on every equery each time.
* The jdbc Prepared statement object deals with pre Compiled sql query.
* The query that comes to database software from java application without values only once, becomes parsed query in db s/w only once and allows client application (java application) to set the values for the query for multiple times, to execute the query for multiple times, ~~the~~ with same (or) different values to gather the results for multiple times is called as precompiled sql query.
* Jdbc Prepared Statement object resides in java application and represent pre Compiled sql query of database software so the programmer can provide further instructions to precompiled query like assigning values, query execution and etc... by using this prepared statement object from java application.
* To execute sql query only for one time in database software use simple statement object. similarly to execute same sql query for multiple times in database software with different or same values use jdbc prepared statement object.

<u>Procedure to work with jdbc Prepared statement object</u>

(1) prepare sql query with parameters / place holders (?)

```
String qry = "insert into student values (?, ?, ?)";
```

place/holders (1, 2, 3) — place holders/parameters

(2) send sql query to database software as pre-Compiled query and represent that query with Prepared statement object.

```
PreparedStatement PS = Con.PrepareStatement(qry);
```

This object represents ~~Prep~~ precompiled query of db s/w being from java application.

This method sends given sql query to db s/w and makes that query as Pre Compiled query in db s/w.

3) set values to parameters of precompiled query by using setXXX() methods.

```
ps.setInt(1, 567);
ps.setString(2, "Raja");
ps.setString(3, "hyd");
```

↳ parameter index

NOTE: The parameters of sql query will have one based index.

Here "hyd" is parameter value.

(4) Execute the query in database software.

```
int result = ps.executeUpdate();
```

(5) Process the results

```
if (res==0)
    System.out.println("record insertion is failed");
else
    System.out.println("record insertion is successful");
```

6) to execute same sql query for multiple times with same or different values repeat step 3 to steps for multiple times.

7) close jdbc Prepared Statement object.

```
ps.close();
```

write a java application to read multiple student details from key board and insert them to database table as records.

NOTE: Here we need to execute same sql query for multiple times for that works with precompiled sql query with support of PreparedStatement object.

```
// PstTest.java (with coding standards)
// Insert multiple student details to DB table
// version : 1.0
// author : team-s
import java.sql.*; import java.util.*;
public class PstTest
{
  public static void main(String args[])
  {
```

```
Connection con = null;
PreparedStatement ps = null;
Scanner sc = null;
sc = new Scanner(System.in);
int n = 0;
if (sc != null)
 {
   //read student count from end user
   System.out.println("Enter no. of student");
   n = sc.nextInt();
 } // if
try {
 // write jdbc code
   // create jdbc connection object
   Class.forName("oracle.jdbc.driver.OracleDriver");
   Con = DriverManager.getConnection("jdbc:oracle:thin:@localhost
                                      :1521:", "scott", "tiger");
                                        ↓
                                       ord
   // prepare sql query with parameters
   String qry = "insert into student values (?,?,?)";
   // create prepared statement object representing precompiled
                                                  sql query
   if (con != null)
     {
       ps = Con.prepareStatement(qry);
     }
   // read each student details from keyboard and insert them
                                  to database table as record.
   if (sc != null && ps != null)
     {
       for (int i=1; i<=n; i++)
         {
           System.out.println("Enter " + i + " student details");
           System.out.println("Enter no: ");
           int no = sc.nextInt();
           System.out.println("Enter name");
           ~~int~~ new String name = sc.next();
```

```
        system.out.println ("Enter address:");
        String address = sc.next();
        //set the above values to parameters of query
        ps.setInt (1, no);
        ps.setString (2, name);
        ps.setString (3, address);
        //execute the query
        int res = ps.executeUpdate();
        //process the result
        if (res==0)
          {
            system.out.println(i +" student details are not inserted");
          }
        else
          {
            system.out.println (i +"student details are inserted");
          }
    }// for
  }// if
}// try
catch ( classNotFoundException cnf)
   {
      cnf.printStackTrace();
   }
catch ( SQLException se)
   {
     se.printStackTrace();
   }
catch (Exception e)
   {
     e.printstackTrace();
   }
finally
 {
    try
     {
       if (ps != null)
            ps.close
     }
     catch ( SQLException se)
       {  se.printStackTrace()
       }
```

```
        try
        {
            if (Con != null)
                Con.close();
        }
        catch (SQLException sel)
        {
            sel.printStackTrace()
        }
    }// finally
}// main
}// class
```

```
//> javac PstTest.java
//> java PstTest
```

* Compare to simple statement object the jdbc Prepared Statement object reduces network traffic (amount of data) between java application and database software in each round trip.
* The setXXX(-) methods allow the programmer to set values to sql query or query parameters in java notation so working with string type input values for sql query becomes easy while working with jdbc prepared statement object.
* By using prepared statement object we can execute both select and non select sql queries with zero or more parameters.
* Provide one based index sql query parameters from left to right order.

<u>Example to execute select sql query by using jdbc PreparedStatement object</u>

```
//create PreparedStatement object
PreparedStatement ps= Con.PrepareStatement("select * from student
                                where sno>=? and sno<=?");
                                          1        2
//set values to the parameters of query
ps.setInt(1,100);
ps.setInt(2,300);
```

```
//execute query
ResultSet rs = ps.executeQuery();
//process results
while (rs.next())
  {
    sop( rs.getInt(1) + " " rs.getString (2) + " "+rs.getString (3));
  }
```

* The SQL query of jdbc PreparedStatement object can be there with or without parameters.

```
//create jdbc PreparedStatement object
PreparedStatement ps = con.PrepareStatement ("select * from student");
//execute query
ResultSet rs = ps.executeQuery();
//process results
while(rs.next())
  {
    sop(rs.getInt(1)+" " + rs.getString (2) + " " + rs.getString(3));
  }
```

query without parameters.

* The parameters of Prepared Statement object related sql query can represent only condition values and input values but they can not represent table names, column name, of the SQL keywords of the query

select * from studen where sname like ? (valid query)

select * from ? where ? like ? (invalid query)

select * ? student where sname like ? (invalid query)

select * from student where sno >= ? and sno<=200 (valid query)

* one java application can have multiple jdbc connection objects, statement objects and ResultSet objects.

* Generally java jdbc applications are two tier applications containing three layers.

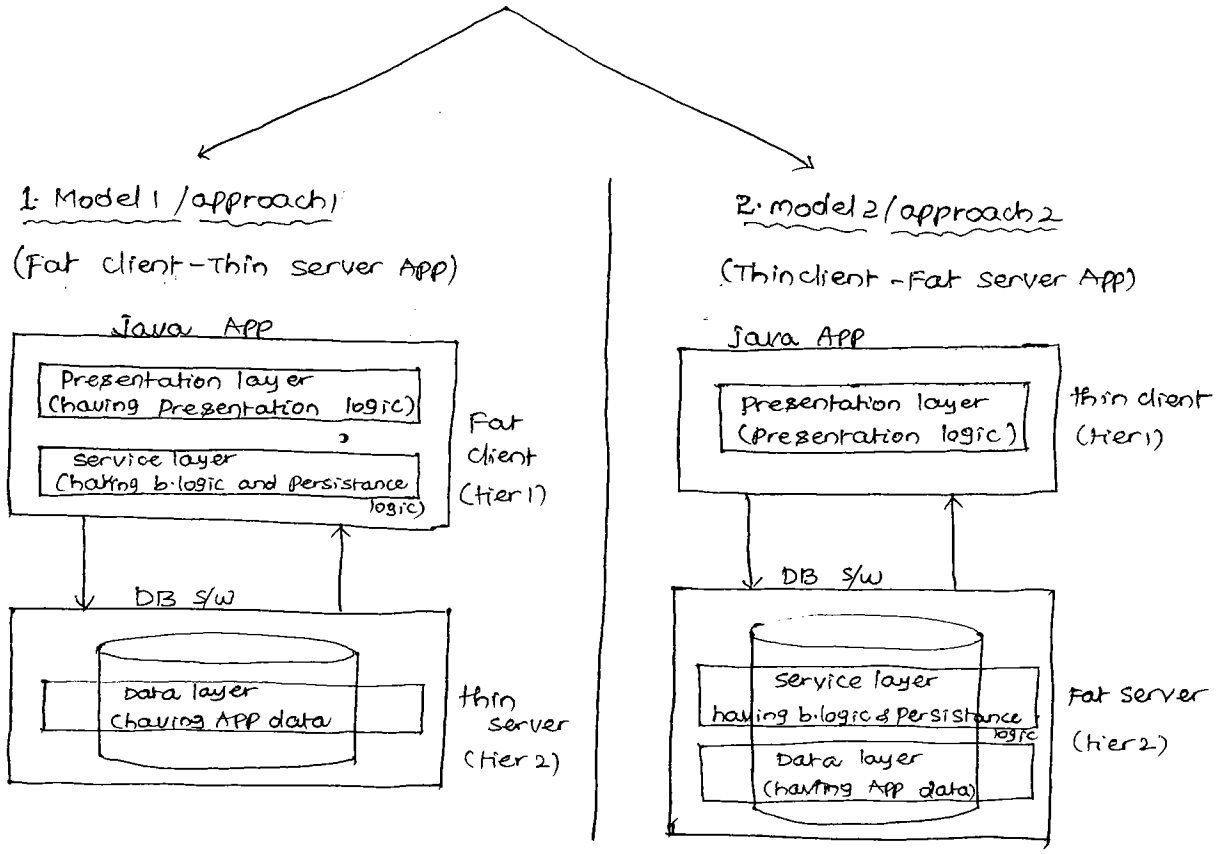
(1) Presentation layer having presentation logic

(2) service layer having business logic and persistance logic.

(3) Data layer having application data.

* Data layer is responsible to maintain, manage application data with the support of persistance stores like business logic, persistance logic and etc.

The two models of Developing Jdbc two-tier applications



* The jdbc applications which deals with simple statement object, Prepared statement object are called as "Fat client - thin server" applications.
* In thin client and Fat server applications the business and persistance logic will be placed in DB S/w as PL/SQL Procedures and functions and to call them from java application the jdbc CallableStatement object will be utilized.
* Since PL/SQL Procedures / functions resides and execute permenantly in the database software they are also called as stored procedures and functions.
* PL/SQL procedure doesn't return a value whereas PL/SQL function returns a value.

* The PL/SQL function, procedure related parameters contains mode along with type.

<u>modes</u> <u>are</u>

In (default)

out

Inout

* If logic is

```
y := x * x;
```

then

x → In parameter

y → out parameter

```
x := x * x;
```

x → Inout parameter.

* we can gather results from PL/SQL procedure from out parameters or Inout parameters.
* we can gather results from PL/SQL function either as return value or outparameters / Inout parameters.
* PL/SQL programming, SQL queries are database software dependent. so Jdbc persistance logic is database software dependent persistance logic. To develop database software independent persistance logic use ORM software like Hibernate. (Object Relational mapping - ORM)

<u>steps</u> <u>to develop</u> <u>PL/SQL</u> <u>Procedure</u> <u>in oracle</u>

<u>step(1)</u>: Launch sql*plus tool.

<u>step(2)</u>: Execute any dummy query

<u>step(3)</u>: Launch ed editor

```
SQL> ed
```

<u>step(4)</u>: Type and save PL/SQL procedure.

```
create or replace first_Pro (x in number, y out number)
as
begin
y := x * x;
end;
/
```

procedure

parameter name

parameter mode

parameter type

step(5): Execute the PL/SQL procedure

File menu → save → exit → SQL> /

Procedure to work with jdbc callable statement object to call PL/SQL procedure of Database software

step(1): Prepare qry calling PL/SQL procedure.

```
String qry = "{ call first_pro (?, ?)}";
```

(1) (2)

Represents parameters

step(2): create CallableStatement object having the above query.

```
CallableStatement cs = Con.prepareCall(qry);
```

cs → Represents the PL/SQL procedure on the db s/w (first_pro)

step(3): set input values to in parameters of PL/SQL procedure call. by using setXxx(-) methods.

```
cs.setInt(1, 20);
```

1 → parameter index

20 → parameter value

step(4): Register out parameters with jdbc data types.

```
cs.registerOutParameter(2, Types.INTEGER);
```

2 → parameter index

Types.INTEGER → jdbc type

* Every out parameter must be registered with jdbc type. All jdbc data types are bridge datatypes between underlaying database software datatypes and java data types. They help jdbc driver to convert database software notation result into java notation result while working with PL/SQL procedure or functions.

Java data type	jdbc data type	oracle data type
int	Types.INTEGER	number
long	Types.LONG	number
String	Types.VARCHAR	varchar2
byte[]	Types.BINARY	BLOB
java.sql.Date	Types.DATE	date
⋮	⋮	⋮

* All jdbc datatypes are constants of java.sql.Types class.
* public static final member variables of a java class/ java interface are called as constants.

NOTE: IN parameters of PL/SQL procedures and functions need not to be register with jdbc types because the jdbc driver can detect their jdbc datatypes automatically based on the values that are set and based on the setXXX(-) methods that are used to set the values to IN parameters.

step (5): Make database software executing PL/SQL procedure.

```
cs.execute();
```

step (6): Gather results from OUT parameters of PL/SQL procedure.

```
int result = cs.getInt(2);
```

↓ Decide this method based on the jdbc type that is used to register out parameter

↳ out parameter index.

step(7): To call PL/SQL procedure for multiple times with same or different values then repeat step(3) to step(6) for multiple times (except step(4)).

step (8): close jdbc callable statement object.

```
cs.close();
```

: If multiple applications of a module (or) multiple modules of a project (or) multiple projects of a company are looking to use same business logic (or) persistance logic then instead of writing those

logics in every application of a module (or) in every module of a project (or) in every project of a Company it is recommanded to write only once as centralized logics in DB s/w in the form of PL/SQL procedure or function and use them in multiple applications of a module (or) in multiple modules of a project (or) in multiple projects of a Company having reusability.

Example application based on above steps

```
// CSTest.java
import java.sql.*;
import java.~~java~~.util.*;
public class CSTest
{
    public static void main(String args[]) throws Exception
    {
        // read input values from key board.
        Scanner sc= new Scanner(System.in);
        System.out.println("Enter a value");
        int no= sc.nextInt();
        // create jdbc connection object
        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection con= DriverManager.getConnection("jdbc:oracle:thin:
                                                    @localhost:1521:orcl",
                                                    "scott", "tiger");
        // prepare sql query calling PL/SQL Procedure
        String qry = "{call first_pro(?, ?)}";
        // create callable statement object
        CallableStatement cs = con.prepareCall(qry);
        // register OUT parameters with jdbc types.
        cs.registerOutParameter(2, Types.INTEGER);
        // set IN parameters
        cs.setInt(1, no);
```

```
//make db s/w to execute PL/SQL procedure
 cs.execute();
//gather results from OUT parameters
 int result = cs.getInt(2);
System.out.println("The Result is"+result);
//close jdbc objects
 cs.close();
 con.close();
}//main
}//class
```

//make sure that first_pro(-,-) PL/SQL procedure is available in oracle DB s/w.

//> javac CSTest.java

//> java CSTest

* Any mechanism based jdbc driver supplied by any vendor can work with all the 3 types of jdbc statement objects.

<u>PL/SQL Procedure in oracle dealing with SQL query</u>

```
create or replace procedure emp_pro(eno in number, name out varchar,
                                    bsal out number)
as
begin
select ename into name from emp where empno = eno;
select sal into bsal from emp where empno = eno;
       ↓          ↓
     column     outparameter
     name
end;
```

```
//CSTest2.java
import java.sql.*;
import java.util.*;
public class CSTest2
{
   public static void main(String args[])throws Exception
   {
```

```
//read input values from keyboard
Scanner sc=new Scanner(System.in);
System.out.println("Enter emp no:");
int no=sc.nextInt();
//create jdbc con obj
Class.forName("oracle.jdbc.driver.OracleDriver");
Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost
                                        :1521:ord", "scott", "tiger");

//Prepare query calling the PL/SQL procedure
String qry="{call emp_pro(?,?,?)}";
//Create callable statement object
CallableStatement cs=con.prepareCall(qry);
//register out parameters with jdbc types
cs.registerOutParameter(2, Types.VARCHAR);
cs.registerOutParameter(3, Types.~~VARCHAR~~ INTEGER);
//set values to IN parameters
cs.setInt(1, no);
//execute the PL/SQL procedure
cs.execute();
//Gather results from OUT parameters
system.out.println("emp name is:"+cs.getString(2));
system.out.println("emp sal is:"+cs.getInt(3));
//close jdbc objects
cs.close();
con.close();
}//main
}//class
```

* In PL/SQL programming of oracle we can use cursor datatype variables to store bunch of records given by SQL select query. For this we generally use sys_ref cursor data type of oracle PL/SQL programming.
* cursor of PL/SQL programming is like jdbc resultset object of java programming.

Example application on cursor

```
create or replace procedure find_emps( fchars in varchar2, myrecords
                                        out sys_refcursor) as
                                            ↓
                                        a cursor type datatype
begin
  open myrecords for
      select empno, ename, sal, job from emp where ename like fchars;
end;
```

* sunmicrosystem is not supplying jdbc type in Types class to register cursor type out parameters so use oracle corporation supplied OracleTypes.CURSOR constant as jdbc data type for the above said situation.
* To receive results from cursor type out parameters use getObject(-) method to receive jdbc resultset object.

```
// CursorCsTest.java
import java.sql.*;
import java.util.*;
import oracle.jdbc.*; // for oracle Types class (this pkg is available in ojdbc14.jar).
class CursorCsTest
{
  public static void main (String args[]) throws Exception
  {
    //read cond character from keyboard
     scanner sc = new Scanner (System.in);
     sop("Enter chars (first letters of emp name)");
     string cond = sc.next();
     cond = cond + "%";   // r becomes r%
    // write jdbc code
     class.forName("oracle.jdbc.driver.OracleDriver");
     Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost
                                                  :1521:ord", "scott", "tiger");
    // create callableStatement object
     CallableStatement cs = con.prepareCall("{call find_emps(?,?)}");
```

```
//register OUT parameters with jdbc types (oracle Corporation supplied
                                                jdbc types)

cs.registerOutParameter (2, oracleTypes.CURSOR);
//set values to IN parameter
cs.setString(1, cond);
//execute PL/SQL procedure
cs.execute();
//gather results from out parameter
ResultSet rs = (ResultSet) cs.getObject(2);
//display result
while(rs.next())
{
  sop(rs.getInt(1) + " " + rs.getString(2) + " " + rs.getString(3) + " "
                                        + rs.getString(4));
}
// close jdbc objects
rs.close();
cs.close();
con.close();
}//main()
}//class
```

* The oracle PL/SQL programming provides a ~~built~~ built-in cursor variable called SQL%ROWCOUNT. This variable always holds no. of records that are effected when non select SQL queries are executed in PL/SQL Procedure or function.

<u>Example application:</u>

<u>PL/SQL Procedure in oracle (Based on non-select query)</u>

```
create or replace Procedure fire_emps (salrange in number, cnt out
                                                number) as

begin
    delete from emp where sal > salrange;
    cnt := SQL%ROWCOUNT;
end;
```

↳ returns a numeric value indicating no. of records that are effected because of above delete query execution

```
// CsTest3.java
public class CsTest3 //ba
{
  public static void main(String argS[]) throws Exception
  {
    //read input values from keyboard
    scanner sc = new scanner (system.in)
    system.out.println("Enter salary range");
    int salrange = sc.nextInt();
    // create jdbc connection object
    class.forName("oracle.jdbc.driver.OracleDriver");
    Connection con = DriverManager.getConnection("jdbc:odbc:oci8:@ord",
                                                 "scott", "tiger");
    // create callableStatement object
    CallableStatement cs = con.prepareCall("{call fire_emps(?,?)*}");
    // register OUT parameters with jdbc types
    cs.registerOutparameter(2, Types.INTEGER);
    // set IN values
    cs.setInt(1, salrange);
    // execute PL/SQL Procedure
    cs.execute();
    // gather results from OUT parameters
    system.out.println("cnt of emps who are fired:"+cs.getInt(2));
    //close jdbc objects
    cs.close();
    con.close();
  } // main()
} // class
```

<u>Working with PL/SQL functions</u>:

* PL/SQL function returns a value.
* Results from PL/SQL function can be gathered ~~fro~~ either from out parameters or from return value. (we can even use both).
* while calling PL/SQL function the query must have a parameter representing return value and that parameter should be treated as out parameter

and should also be registered with jdbc type.

Example PL/SQL function:

```
create or replace function first_fx (x in number, y out number) return
                                                              number
as
z number;
begin
  y:=x*x;
  z:=x*x*x;
  return z;
end;
```

Procedure to call PL/SQL function from java application by using jdbc

Callable Statement object

1) Prepare query calling PL/SQL function

```
String query = "{ ? = call first_fx (?, ?) }";
```

(1: parameter to hold return value of PL/SQL function; 2, 3: parameters)

2) create CallableStatement object representing PL/SQL function.

```
CallableStatement cs = Con.prepareCall (query);
```

3) register return parameter, out parameter with jdbc types.

```
cs.registerOutparameter (1, Types.INTEGER); //return parameter
cs.registerOutparameter (3, Types.INTEGER) ;
```

4) set values IN parameters

```
cs.setInt (2, 100);
```

(2: parameter index; 100: value)

5) Make database software executing PL/SQL function

```
cs.execute();
```

6) Gather results from return parameter, out parameters

```
int res1 = cs.getInt(1); // gives cube value (z);
int res2 = cs.getInt(3); // gives square value (y);
s.o.p(res1+" "+res2);
```

7) To call and execute PL/SQL function for multiple times with same or different input values repeat step(4) to step(6) for multiple times.

8) close CallableStatement object

```
cs.close();
```

```
// CsFxTest.java
import java.sql.*;
import java.util.*;
public class CsFxTest
{
  public static void main(String args[]) throws Exception
  {
    //read input values from key board
    Scanner sc=new Scanner(System.in);
    System.out.println("Enter value");
    int no = sc.nextInt();
    //create jdbc connection object
    class.forName("oracle.jdbc.driver.OracleDriver");
    Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost
                                                  :1521:orcl","scott","tiger");
    //prepare query calling PL/SQL function
    String query = "{ ? = call first_fx(?,?)}";
    //create CallableStatement object
    CallableStatement cs = con.prepareCall(query);
    //register return, OUT parameters with jdbc types.
    cs.registerOutParameter(1, Types.INTEGER); //return parameter
    cs.registerOutParameter(3, Types.INTEGER);
    //set values to IN parameters
    cs.setInt(2, no);
    //execute PL/SQL function
    cs.execute();
    //gather results from return, OUT parameters
    System.out.println("cube value:"+cs.getInt(1));
    System.out.println("square value:"+cs.getInt(3));
```

```
        //close jdbc objects
        cs.close();
        con.close();
    }//main
}// class
```

PL/SQL function in oracle (dealing with sql queries)

```
create or replace function emp_fx(stno number, endno number, name out
                                  varchar2, bsal out number) return varchar2
as
  desg varchar2(20);
begin
   select ename into name from emp where empno>=stno and empno<=endno;
   select sal into bsal from emp where empno>=stno and empno<=endno;
   select job into desg from emp where empno>=stno and empno<=endno;
   return desg;
end;
/
```

```
select ename, sal, job into name, bsal, desg from emp where
empno >= stno and empno<=endno
```

```
// CsFxTest2.java
public class CsFxTest2
  {
    public static void main(String args[]) throws Exception
      {
        //read input values from cmd line args
        int stno = Integer.parseInt(args[0]);
        int endno = new Integer(args[1]).intValue();
        // create jdbc con obj
        class.forName("oracle.jdbc.driver.OracleDriver");
        Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost
                                                   :1521:orcl", "scott","tiger");
        //create callableStatement object
        callableStatement cs=con.prepareCall("{ ?= call emp_fx(?,?,?,?)}");
        cs.registerOutParameter(1, Types.VARCHAR);
        cs.registerOutParameter(4, Types.VARCHAR);
        cs.registerOutparameter(5, Types.INTEGER);
```

```
//set values in parameters
 cs.setInt(2, stno);
 cs.setInt(3, endno);
//execute PL/SQL function
 cs.execute();
//gather results from out parameters, return parameters
 System.out.println("emp name is:" + cs.getString(4));
 System.out.println("emp salary is:" + cs.getInt(5));
 System.out.println("emp Desg is:" + cs.getString(1));
//close jdbc objects
 cs.close();
 con.close();
}//main
}//class
//> javac CsFxTest2.java
//> java CsFxTest2 7900 8000
```

Example application

PL/SQL function having both select and non select SQL queries

```
create or replace function test_fx(no in number, cnt out number) return
                                                        sys_refcursor
                                                          ↑
                                                   cursor as a return
                                                          type
as
 mycur sys_refcursor;
begin
 delete from emp where empno=no; → non select query
 cnt:= SQLY.ROWCOUNT; → holds a numeric value representing the no. of records
                                                  that are deleted.
 open mycur for
    select empno, ename, sal, job from emp; → Here cursor holds bunch of
                                              selected records
 return mycur;
end;
/
```

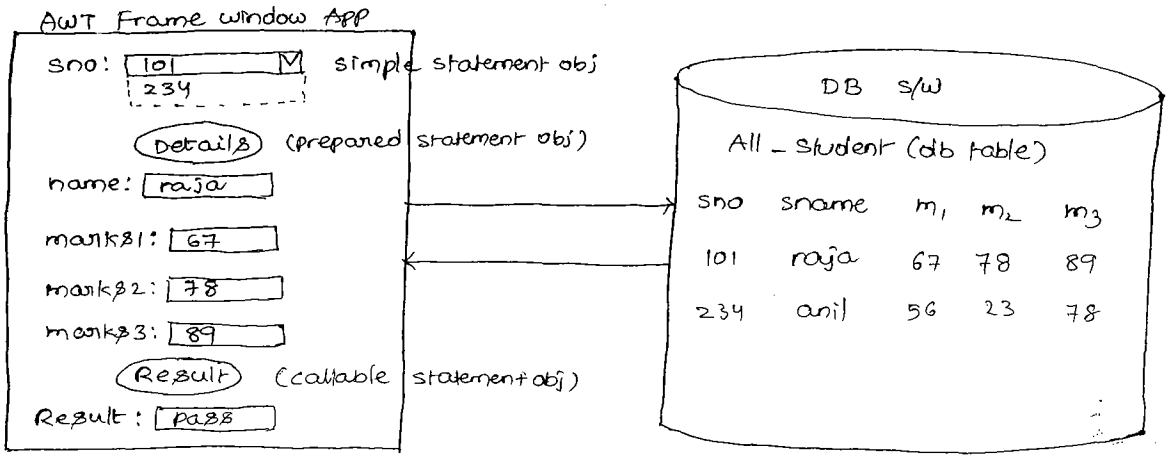
```
//CsTest3.java
import java.sql.*;
import java.util.*;
import oracle.jdbc.*;
public class CsTest3
  {
    public static void main(String args[]) throws Exception
    {
      // read input values from keyboard
      Scanner sc=new Scanner(System.in);
      System.out.println("Enter employee no.");
      int no=sc.nextInt();
      // create jdbc connection object
      Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
      Connection Con=DriverManager.getConnection("jdbc:odbc:oradsn",
      //use type4 driver instead of type1                "scott","tiger");
      //create CallableStatement object
      CallableStatement cs=Con.prepareCall("{?= call test_fx(?,?)}");
      //register out, return parameters with jdbc types
      cs.registerOutParameter(1,OracleTypes.CURSOR);
      cs.registerOutParameter(3, Types.INTEGER);
      //set values to IN parameter
      cs.setInt(2,no);
      //call PL/SQL function
      cs.execute();
      //gather results from OUT, return parameters
      ResultSet rs=(ResultSet) cs.getObject(1); //from return parameter
      int count=cs.getInt(3); //from OUT parameter
      //process the result
      if(count==0)
        System.out.println("Records not deleted");
      else
        System.out.println(count+" records deleted");
```

```
while (rs.next())
  {
    system.out.println(rs.getInt(1)+" "+rs.getString(2)+" "
                       +rs.getInt(3)+" "+rs.getString(4));
  }
//close jdbc objects
cs.close();
con.close();
}//main
}//class
```

* All jdbc drivers give support to work with all the three types of jdbc statement objects, but type 1 driver can't use CallableStatement object to call cursor based PL/SQL functions and procedures.

<u>Conclusion on three jdbc statement objects</u>

* use simple statement object to execute SQL query only for one time in the entire application.
* use PreparedStatement object to execute same SQL query for multiple times with same or different values by making that query as pre-compiled query.
* To make persistance logic as centralized and reusable persistance logic take the support of PL/SQL functions and procedures of database software and use jdbc CallableStatement object to call them from java applications.
* Using AWT, swing concepts we can develop GUI front end based applications to interact with database software.



* By adding jdbc code in AWT, swing applications we can make them interacting with database software.

* Example application to develop AWT based GUI front end application *

Resources required in the above AWT App

Container
- Frame window

Components
- Label → 6
- TextField → 5
- ChoiceBox → 1
- Button → 2

LayoutManager
- FlowLayout

Events
- ActionEvent

EventListener
- ActionListener

Event handling method
- actionPerformed(-)

For the source code of above App refer App 9 in booklet page no: 22-24

* In the above application all the three jdbc statement objects are used.

* Student number to choice box should come only once in the entire execution of the application so its relevant query (select sno from student) should ~~come~~ be executed only once so simple statement object is recommanded.

* Based on the student number that is choosen from the choice box the remaining details of the student should come when details button is clicked. In this process we need to execute same sql select query (i.e. select * from student where sno = ?) for multiple times with different values so PreparedStatement object is used.

* To centralize ~~re~~ the student result generalization logic PL/SQL Procedure is taken in the db s/w so this PL/SQL procedure should be called when result button is clicked. For this CallableStatement object should be used.

* It is always recommanded to create jdbc objects in one time execution block like constructors and utilize them in repeatedly executing blocks like event handling methods for better performance.
* If java application wants to talk with database software for 10 times instead of creating 10 no. of jdbc connection objects create one jdbc connection and use it for ten no. of times.
* Method called from constructor executes as one time execution block.
* self class methods and super class public/protected methods in sub class can be called without object.
* The logic that you want to execute automatically during application startup is called as load on startup logic. we need to place these logics either in static block (or) constructor.
* In the above diagram based application values in the choice box should be populated as load on startup values.
* Jdk supplies built-in tool called javaw to make the GUI java Apps as windows applications/desktop applications.
* Basic jdk setup is enough to work with this tool.

Example:

Right click on desktop → new → Short cut → [javaw AllStmtsTest] → next → name for short cut [collegeapp] → finish → Right click on collegeapp icon → Properties → Short cut tab → Start in [Application path]

Short cut key [Ctrl+Alt+Y]

→ change icon → apply → ok.

NOTE: The above javaw tool can not be used to make CUI applications as windows application.

How to execute CUI java applications as windows based desktop application?

1) Prepare batch file as shown below in any place of windows operating system.

myapp.bat (create on the desktop)

```
cd E:\NApps\Apps\advjava\jdbc
e:
javac selectTest.java
java SelectTest
pause
```

what is the difference between AWT and Swing?

AWT	Swing
(1) Components look and feel is OS dependent	(1) OS independent
(2) Every Component maintains one peer Component at OS level, so these Components are heavy weight	(2) No peer Component at OS level so these Components are light weight
(3) gives minimum Components support (16+)	(3) gives more Components support (32+)
(4) Gives minimum Layout Manager support (4)	(4) Gives more Layout Manager support (6+)

Procedure to develop AllStmts application as Swing application by using the NetBeans IDE 6.7

Step(1): create java project in netbeans IDE.

File menu → New Project → java → java application → next →

Project Name: myproject → finish

Step(2): Add Jframe to the project source packages

Right click on source packages folder → new → Jframe Form

class Name: AllStmtsTest → finish

Step(3): Design the frame window as shown in the previous diagram. by using drag and drop operations.

Step(4): Register actionListener on button Components

Right click on Details button → Events → action → actionPerformed.

Perform the same operation on Result button.

Step(5) : Declare the following instance variables in the source code of AllStmtsTest.Java

```
Connection Con = null;
Statement St = null;
PreparedStatement ps = null;
CallableStatement cs = null;
ResultSet rs1 = null, rs2 = null;
```

Step(6) : write following user defined method in AllStmtsTest.Java having jdbc code and call that method from Constructor.

```
public void myInit()
{
  try
  {
    Class.forName("oracle.jdbc.driver.OracleDriver");
    Con = DriverManager.getConnection("jdbc:oracle:thin:@localhost
                                        :1521:ord", "scott", "tiger");
    // load-on-startup logic
    Statement st = Con.createStatement();
    rs1 = st.executeQuery("select sno from All_Student");
    while(rs1.next())
    {
      jComboBox1.addItem((String) rs1.getString("sno"));
    }
    // create other jdbc statement objects
    ps = Con.prepareStatement("select * from All_Student where
                                        sno = ?");
    cs = con.prepareCall("{call FIND_PASS_FAIL (?,?,?,?)}");
    cs.registerOutParameter(4, Types.VARCHAR);
  }
  catch(Exception e)
  {
    e.printStackTrace();
  }
}// myInit()
```

```
public AllStmtsTest()
{
    initComponents();
    myInit();
}
```

→ Constructor

step(7): write following code jButtonActionPerformed(-) method as details button related Eventhandling code.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
{
    //Code related Details button

    try
    {
        int no = Integer.parseInt((String)jComboBox1.getSelectedItem());
        ps.setInt(1,no);
        //execute the query
        rs2 = ps.executeQuery();
        //Set the record of ResultSet object to text boxes
        if(rs2.next()){
        jTextField1.setText(rs2.getString(2));
        jTextField2.setText(rs2.getString(3));
        jTextField3.setText(rs2.getString(4));
        jTextField4.setText(rs2.getString(5));
        }
        rs2.close();
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
}
```

gives the selected item of ComboBox

step(8): write the following code in jButton2actionPerformed(-) method for result button.

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt)
{
    //for result button
    try
    {
        //Set IN parameters values
        cs.setInt(1, Integer.parseInt(jTextField2.getText());
```

```
cs.setInt(2, Integer.parseInt(jTextField3.getText());
cs.setInt(3, Integer.parseInt(jTextField4.getText());
//execute PL/SQL Procedure
cs.execute();
//gather result from OUT parameter and set result to textbox
jTextField5.setText(cs.getString(4));
}
catch (Exception e)
{
e.printStackTrace();
}
```

Step(9): Add ojdbc14.jar file to the libraries of the project.

Right click on libraries folder → Add JAR → Browse and select ojdbc14.jar

Step(10): run the application

Right click on source code of AllStrutsTest.java → Run File

How to run a compiled java application from any folder of the project?

) Add the directory location of the application related .class file to classpath environment variable, to run that java application from any folder of the computer.

```
e:\apps\temp
        |—— TestApp.java
```

TestApp.java

```
public class TestApp
{
  public static void main(String args[])
  {
    sop("TestApp: main(-)");
  }
}
//>javac TestApp.java
```

* Add e:\apps\temp folder to CLASSPATH environment variable

my Computer → properties → advanced tab → env variables →

variable name : CLASSPATH

value : e:\apps\temp; <other values>;.

→ ok → ok → ok

e:\> java TestApp (success)

d:\> java TestApp (success)

c:\xyz> java TestApp (success)

* Large scale companies perprefer working with the commercial oracle db s/w whereas small scale companies prefer working with the open source db s/ws like mysql, postgre sql etc...

<u>MySQL</u>

type : multiuser db s/w

vendor: Devx or Sun micro systems (Java soft)

open source or free db s/w

default portno : 3306

default username and password : root, root

(can be changed during installation/activation of s/w)

version : 4.x

Allows to create multiple logical databases

Default logical db names : mysql, test

To download software : www.devx.com

* After installing MySQL db s/w we must activate it seperately. For this go to mysql_home\bin and use winmysqladmin tool and look for traffic signal symbol in the system tray area.

<u>Procedure to modify username and password of MySQL db s/w after installation</u>

Right click on traffic signal symbol in system tray area → show me → my.ini setup tab → modify user and password → save modification

Procedure to create logical db, db table in mysql db s/w

Step(1): Launch the SQL prompt of MySQL database software

c:\mysql\bin ⟶ click on mysql.exe file

Step(2): create logical database

mysql> create database mydbl;

↓ Name of the logical database

Step(3): Connect to this logical database to mydbl

mysql>Connect mydbl;

Step(4): create db table in mydbl logical db.

mysql> create table Employee (EID int(5) Primary key, FIRSTNAME varchar(20), LASTNAME varchar(20), EMAIL varchar(20));

Step(5): Insert records into this database table.

msql> insert into employee values(345, 'ramesh', 'rao', 'rao@gmail.com');

* If you are not aware of the sql queries of certain database software then you can use some third party GUI based db tools to perform various operations on the db s/w.

Ex: mysql front for mysql db s/w

sqlyog for mysql db s/w

TOAD for mysql db s/w

TOAD for oracle db s/w

Pl/sql developer for oracle db s/w.

NOTE: All the above tools must be installed ~~seperately~~ seperately.

Procedure to perform above given operation (like logical db creation and etc) by using sqlyog tool.

(1) ~~Kam~~ Launch sqlyog tool

Launch → Continue

MySQL Home address : localhost
user name : root
password : root1

port : 3306

→ connect

(2) create new logical database.

right click on root@localhost → create database → mydb2 → create

(3) create database table in mydb2 logical database.

right click on mydb2 → create table → Enter values for table →

create table → table name : Employee

(4) Add records to the table.

right click on employee table → insert/ upgrade data → values

Procedure to develop java application to interact with mysql db s/w by using type-1 jdbc driver:

step(1): Install odbc driver for mysql seperately in your Computer (It will not come as default odbc driver).

step(2): create dsn for odbc driver for mysql.

Control panel → performance and maintainance → administrative tools → data sources → user dsn → add → MySQL ODBC 5.1 Driver → finish →

Data source Name : mysqldsn

server : localhost

port : 3306

user : root

password : root1

Database : mydb1

→ Test → ok

step(3): Develop java application as shown below by using type1 jdbc driver.

```
// selectTest.java
import java.sql.*;
public class SelectTest
{
```

```
public static void main (String args[]) throws Exception
{
    //load jdbc driver class
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    //establish connection with db s/w
    Connection Con= DriverManager.getConnection ("jdbc:odbc:mysqldsn",
                                                 "root", "root@");
    //create Prepared statement object
    PreparedStatement ps= Con.prepareStatement ("select * from
                                                 employee");
    //execute query
    ResultSet rs = ps.executeQuery();
    //process results
    while (rs.next())
      { sop(rs.getInt(1)+" "+rs.getString (2)+" "+rs.getString(3)+
                                  " "+rs.getString (4));
      }
    //close jdbc objects
    rs.close();
    ps.close();
    Con.close();
  }
}
```

* The Devx Company supplied type-4 mechanism based jdbc driver for mySQL is technically called as Connector/J jdbc driver.

Connector/J jdbc driver

target db s/w : mysql db s/w
version: 3.0.8
vendor : Devx
seperate installation is required
driver mechanism ; type 4
open source jdbc driver
jdbc driver class : org.gjt.mm.mysql.Driver (or)
com.mysql.jdbc.Driver

Jdbc URL : jdbc : mysql : /// <logical db name>

(or)

jdbc : mysql : / <hostname> : <portno> / <logical db name>

ZIP file that represents this jdbc driver : "mysql - Connector - java - 3.0.8 - stable . ZIP."

(download from "deux.com).

Jar file that represents this jdbc driver : mysql - Connector - java - 3.0.8 - stable - bin . jar file.

(collect from the above ZIP file extraction).

Procedure to work with Connector/J jdbc driver in java application to interact with mysql database software

Step(1): Gather the above said Jar file mysql - Connector - java - 3.0.8 - stable - bin . jar file and add in the CLASSPATH.

Step(2): Gather Connector/J jdbc driver details as shown above.

Step(3): Develop java application by using Connector/J jdbc driver as shown below.

```
//SelectTest.java
import java.sql.*;
public class SelectTest
{
    public static void main(String args[]) throws Exception
    {
        //load jdbc driver class
        class.forName("org.gjt.mm.mysql.Driver");
        //or
        //class.forName("com.mysql.jdbc.Driver");
        //establish connection with DB s/w
        Connection con=DriverManager.getConnection("jdbc:mysql:///mydbl",
                                                    "root","root1");
                                                    protocol
        //or
        Connection con=DriverManager.getConnection("jdbc:mysql://localhost:
                                                    3306/mydbl","root","root1");
}
}
```

* To work with jdbc driver of any database software supplied by any vendor we must gather the following details.

(i) jdbc driver class name that implements java.sql.Driver interface

(ii) Jdbc URL (or) db URL to locate database software.

(iii) Jar file that represents Jdbc driver.

* while developing small scale projects having the installation of MSoffice take the support of MSAccess as database software.

· MS-Access

Type: single user db s/w

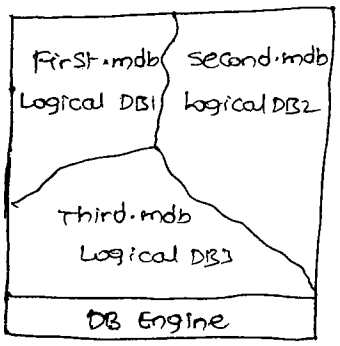
Version: 11.6 (MS-Office 2003)

vendor: microsoft

Commercial software

Allows to create logical databases

Each .mdb file acts as one logical database.



(Physical MS-Access DB s/w

Procedure to create logical database and database table in MS-Access db s/w

start → programs → MS-office → MS-Access → create new → blank database → Filename: mydb → create → create table design view

Field Name	Data type
pid	Number
Pname	text
pqty	Number

→ Ctrl+w → yes → ok → NO → Launch product table → ter data

as records → click on save button.

* Industry generally prefers working with type-1 jdbc driver to interact with MS-Access database software because the type-2 and type-4 jdbc drivers of MS-Access db s/w are commercial jbc drivers.

Procedure to develop java application to interact with MS-Access db s/w

Step(1): create dsn for odbc driver for MS-Access.

Control panel → performance and maintainance → adminstrative tools → datsources → add → Microsoft Access Driver → Finish →

Data Source Name : accdsn

→ select → Browse and select the mydb.mdb file → ok

Step(2): Develop java application by using type-1 driver as shown below.

```
// AccessTest.java
import java.sql.*;
public class AccessTest
  {
    public static void main(String args[]) throws Exception
     {
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
        Connection con=DriverManager.getConnection("jdbc:odbc:accdsn");
                                                      dsn created above
        Statement st=con.createStatement();
        ResultSet rs=st.executeQuery("select * from product");
        while(rs.next())
         {
           sop(rs.getInt(1)+" "+rs.getstring(2)+" "+rs.getInt(3));
         }
        rs.close();
        st.close();
        con.close();
     } // main
  } // class.
```

* The single user of MS-Access is nothing but the windows login user.

3 overloaded forms of DriverManager.getConnection() method

1) public static Connection getConnection (String url, String username, String password) throws Exception.

2) public static Connection getConnection (String url);

3) public static Connection getConnection (String url, Properties pro)

↓
Map datastructure of java.util package

use (1) and (3) form to interact with multiuser db s/ws (like oracle...).
use (2) form to interact with single db softwares (like MS-Access).

understanding java.util.Properties class

* It is a map data structure.
* It is subclass to java.util.Hashtable.
* It allows only String objects as keys and values of elements.
(other map data structures allows any objects as keys and values in elements).
* It allows the programmer to gather the element values from outside the application as text properties file content.

abc.txt (text properties file)

```
name = ramesh
age = 30
address = hyd
# it is demo file
```

× The text file that maintain the entries in the form of key = value pairs is called as text properties file.

Example application to create java.util.Properties class object by utilizing the data of text properties file.

```
e:\apps
  |-> advjava
      |-> jdbc
          |-> myfile.properties
          |-> PropsTest.java
```

myfile.properties

```
# Demo file
name = raja
age = 30
address = hyd
```

P(Properties class object)

keys	values	
name	raja	(element-0)
age	30	(element-1)
address	hyd	(element-2)

PropsTest.java

```
//PropsTest.java
import java.io.*;
import java.util.*;
public class PropsTest
{
    public static void main(String args[]) throws Exception
    {
        //locate properties file
        FileInputStream fis = new FileInputStream("myfile.properties");
        //create properties class object and load myfile.properties data
        //into that object
        Properties p=new Properties();
        p.load(fis);

        system.out.println("data of Properties class obj is:"+p.toString());
        System.out.println("data value of name key is:" +p.getProperty("name"));
    }//main
}//class.
```

* In real time projects the dba changes the username and password of database software at regular intervals for security reasons. (like once in a month).
* To make jdbc code as flexible code it is recommanded to pass database username and password values to application from outside the application by taking the support of properties file.
* The standard principle in software industry is don't hard code any values in your java application that are changeble in the future. It is recommanded to collect them from outside the application by using properties file.

Example on (3) form of DriverManager.getConnection (-) method

```
e:\apps
    |-> advjava
        |-> jdbc
            |-> DbDetails.properties
            |-> SelectTest.java
```

DbDetails.properties

```
# DB details
user = scott
password = tiger
```

fixed keys
as given in documentation

SelectTest.java

```
//SelectTest.java
import java.io.*;
import java.sql.*;
import java.util.*;
public class SelectTest
{
    public static void main (String args[]) throws Exception
    {
        //locate properties file
        FileInputStream fis = new FileInputStream ("dbdetails.properties");
        //store dbdetails.properties file data into java.util.Properties class object
        Properties p = new Properties();
        p.load(fis);
        //register jdbc driver with DriverManager service
        Class.forName ("oracle.jdbc.driver.OracleDriver");
        Connection con = DriverManager.getConnection ("jdbc:oracle:thin:@localhost:1521:orcl", p);
        ==========
    }
}
```

```
//> javac SelectTest.java
//> java SelectTest
```

Postgresql

allows to create logical dbs the default logical db is postgres

Type: multi-user db s/w
Version: 8.3
Vendor: university of california & Mr. Scott.
open source db s/w
default portno: 5432
to download s/w: download as zip file from www.postgresql.org website (postgresql-8.3.13-1.zip)

* To install postgresql software extract postgresql-8.3.13-1.zip file to a folder and use the setup file that comes after extraction.

Procedure to create ~~logical database and~~ database table in postgresql in database software (in the default logical database postgres)

Step(1): start postgresql database software.

start → programs → postgresql 8.3 → start service

Step(2): open sql prompt of postgresql database software.

start → programs → postgresql → psql to postgres → password for user postgres: sathya
↳ choosen during installation

NOTE: Here we are connected to default logical database called postgres

Step(3): create database table

```
create table product (pid int, pname varchar(20), price real);
```

Step(4): Insert records into database table

```
insert into product values (101,'table', 678.45);
insert into product values (102,'pen', 348.45);
commit;
select * from product;
```

Pid	Pname	Price
101	table	678.45
102	pen	348.45

* The type4 mechanism based third party vendors like jdbc drivers for postgresql db s/w is technically called as postgresql thin driver.
* The type4 mechanism jdbc drivers are called as thin drivers whereas the type1 and type2 mechanism based drivers are called as thick drivers.

<u>postgres thin driver (type4)</u>

target db s/w : Postgres db s/w
mechanism : type4
vendor : open community (university of California)
open source jdbc driver
jdbc driver class : org.postgresql.Driver
URL : jdbc:postgresql: <logical db name>
jar file that represents jdbc driver : postgresql-8.4-701.jdbc3.jar
(for jdbc3 spec based jdbc driver)
postgresql-8.4-701.jdbc4.jar
(for jdbc4 spec based jdbc driver)

<u>Procedure to develop java application interacting with postgresql db s/w by using the above thin driver</u>

<u>step(1):</u> Arrange the postgresql database software related thin driver based jar file in a folder.

E:\postgresql jarfiles
|→ postgresql-8.4-701.jdbc4.jar
(for jdk 1.6 environment)
|→ postgresql-8.4-701.jdbc3.jar
(for jdk 1.5 environment)

<u>step(2):</u> Add one of the above given two jar files to classpath.

<u>step(3):</u> Develop java application as shown below by using the postgresql thin driver.

```
// PostgreSQLApp.java
import java.sql.*;
```

```
public class postgreSQLApp
{
  public static void main (String args[]) throws Exception
  {
    //create jdbc con
    Class.forName("org.postgresql.Driver");
                  ↓package   ↳ jdbc driver class name
    //Connection Con= DriverManager.getConnection("jdbc:postgresql:
                                                   Protocol
                                  postgres", "postgres", "satya");
                                  logical db name  db user name  password
    // or
    Connection Con = DriverManager.getConnection("jdbc:postgresql://
                                                  localhost:5432/postgres",
                                                  "postgres", "satya");
    //create jdbc statement object
    Statement st= Con.createStatement();
    //send and execute SQL query
    ResultSet rs = st.executeQuery("Select * from product");
    //process the result
    while(rs.next())
    {
      sop(rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3));
    }
    rs.close();
    st.close();
    Con.close();
  }
}
```

Step(4): compile and execute the java application.

* Latest specification of jdbc is : jdbc 4.0
* Each that is developed based on jdbc specification contains version.

  type1 version: 2.x

  oracle thin driver version: 9.2.x

  postgres thin driver version: 8.4.x
* JEE module is not a installable software, it is a big s/w specification having lots of sub specifications like servlet specification JSP specification, EJB specification and etc.

* Latest JEE specification is JEE5

In that

servlet 2.5 spec

JSP 2.1 spec

EJB 3.0 spec

JMS 1.2 spec are there as sub specifications.

Meta-Data

* Data about data is called as meta data that means gathering much information about given information is nothing but meta-data.
* If data is one website then search engine website looks like meta data.
* In jdbc meta data programming is useful to know limitations, capabilities and facilities of underlaying database software and its resources.
* Jdbc supports 3 meta data programming concepts.

1) DatabaseMetaData
2) ResultSetMetaData
3) ParameterMetaData

DatabaseMetaData :

* DatabaseMetaData object means it is the object of a class that implements "java.sql.DataBaseMetaData" interface.
* By invoking various methods on DatabaseMetaData object we can gather the following details.

```
DataBaseMetaData dbmd = Con.getMetaData();
```

Method declared in java.sql.Connection interface

→ Database name and version

→ support for PL/SQL procedures and functions

→ Max characters in table name and column name etc...

ResultSetMetaData

* It gives various details about the database table that is represented by Jdbc ResultSet object like column names, column datatypes, column count and

etc...

```
ResultSet rs = st.executeQuery("Select *from student");

ResultSetMetaData rsmd = rs.getMetaData(); // method declared in
                                              java.sql.ResultSet
```

* ResultSetMetaData object means it is the object of a class that implements "java.sql.ResultSetMetaData" interface

Parameter MetaData:

* Gives details about parameters available in sql queries of jdbc PreparedStatement object and CallableStatement object.

```
PreparedStatement ps = Con.prepareStatement("insert into student values
                                              (?,?,?)");
                                              ↓
                                              parameters of the query

ParameterMetaData pmd = ps.getParameterMetaData();
```

* ParameterMetaData object means it is the object of a class that implements "java.sql.ParameterMetaData" interface.
* Using pmd object we can gather the details of parameter like parameter name, type and other details.
* MetaData programming is no way related with performing CURD operations on database tables. It is given to gather advanced details about database software and other resources.

For example application on DataBaseMetaData refer application 16 of the booklet (page no-29)

* If any method calls of MetaData programming returns 0 or null then we need to understand the underlaying jdbc driver is not capable of gathering that information.
* The metadata programming related output will change based on the database software and the jdbc driver we use.
* DatabaseMetaData is useful to develop certain database tables like sql DB tools like sqlyog, mysqlfront, toad etc....

```
class.forName("oracle.jdbc.driver.OracleDriver");
Connection con=DriverManager.getConnection(jdbc:oracle:thin:@localhost
                                        :1521:ord", "scott", "tiger");
Statement st = Con.createStatement();
ResultSet rs = st.executeQuery("select * from student");
  while (rs.next())
    {
      - - -
    }
```

* The above code gives db table (student records) but inorder to display them with column names and values take the support of "ResultSetMetaData" object.

<u>Example code to print Resultset object data along with column names and column datatypes</u>

```
class.forName("oracle.jdbc.driver.OracleDriver");
Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:
                                        1521:ord", "scott", "tiger");
Statement st=Con.createStatement();
ResultSet rs = st.executeQuery("select * from student");
ResultSetMetaData rsmd = rs.getMetaData();
int colcnt = rsmd.getColumnCount();
//to print column names
for(int i=1; i<colcnt; i++)
  {
    system.out.println(rsmd.getColumnLabel(i) + "\t\t\t");
  }
sop("\n");
// to print datatypes
for (int i=1; i<=colcnt; i++)
  {
    sop(rsmd.getColumnTypeName(i) + "\t\t\t");
  }
sop("\n");
```

```
        while (rs.next())
         {
          sop( rs.getInt(1) + "\t\t\t" + rs.getString(2) + "\t\t\t" +
                                          rs.getString(3));
         }
        rs.close();
        st.close();
        con.close();
      }
    }
```

* ResultSet MetaData programming is quite useful towards reports generation on database table data.

<u>How can you retrieve database table records by passing table name to application dynamically (without knowing columns)?</u>

use ResultSetMetadata object as shown below

```
import java.sql.*;
public class xyz
 {
   Public static void main (String args[]) throws Exception
    {
      class.forName("oracle.jdbc.driver.OracleDriver");
      Connection Con = DriverManager.getConnection ("jdbc:oracle:thin:@localhost:
                                                    1521:ord", "scott", "tiger");
      Statement st = con.create Statement();
      ResultSet rs = st.executeQuery ("select * from" + args[0]);
      ResultSetMetaData rsmd = rs.getMetaData();
      int Colcnt = rsmd.getColumnCount();
      //to print Column names
      for (int i = 1; i <= Colcnt; i++)
        {
         sop( rsmd.getColumnTable (i) + "\t\t");
        }
      sop();
```

```
//to print col datatypes
 for (int i=1 ; i<=colcnt ; i++)
   {
    sop(rsmd.getColumnTypeName(i) + "\t\t");
   }
 sop();
//to display records
 while(rs.next())
  {
    for (int i=1 ; i<=colcnt ; i++)
      {
        sop(rs.getString(i)+ "\t\t");
      }
    sop();
  }
  rs.close();
  st.close();
  con.close();
 }
}
```

```
o/p   Eid      Ename    Add
      Number   Varchar  varchar2   getColumnLable
      101      Raja     hyd
                                   getColumnTypeName
      235      Ramesh   hyd
                                   getString
```

NOTE: while executing the above code pass database table name as Command Line argument.

what is the difference between ResultSet object and ResultSetMetaData object?

* ResultSet object points to table data whereas ResultSetMetaData object points to database table structure like column names, column datatypes and etc...
* Most of the jdbc drivers are not giving support for parameterMetaData programming.
* The maximum vendors of jdbc drivers are not providing implementation for "java.sql.parameterMetaData" interface.

DB s/w

1. Conventional DB s/w
   - oracle
   - sybase
   - MS-Access
   - Mysql
   - postgresql

2. Non-Conventional DB s/w
   - MS-Excel
   - csv-files
   - tab-files

* Non-Conventional DB s/w will be used as helping/ supporting DB s/w when Conventional DB s/ws are used as main DB s/ws.
* In ICICI banking project oracle is taken as Main DB s/w and MS-Excel is taken as helper DB s/w to get the printouts of accounts statements after every 3 months for this they move last 3 months data from oracle to MS-Excel.
* MS-Excel gives good performance towards print operations when compare to oracle.
* Since odbc drivers are already available to communicate with all non-Conventional DB s/ws. So use jdbc driver type-1 for java application to non-Conventional DB s/w communication.

<u>MS-Excel</u> :

MS-Excel s/w (physical DB s/w)

- → college.xls (Logical DB1)
  - → sheet1 (db table 1)
  - → sheet2 (db table 2)
- → Company.xls (Logical DB2)
  - → sheet1 (db table 1)
  - → sheet2 (db table 2)

sheet1 of college.xls

sno	sname	add
101	Ravi	hyd
345	Anil	hyd
123	Rajesh	vizag

← (db table column names)

db table col values

db tables records

* Here College.xls and Company.xls are workbooks.

Procedure to interact with MS-Excel from java application

step(1): create work book in MS-Excel s/w (College.xls as shown above)

E:\Apps\temp\College.xls file

step(2): create DSN for the Microsoft Excel driver

Control panel → performance and maintanence → Adminstrative tools →

Data Sources → User DSN tab → Add → Microsoft Excel (.xls) →

Finish →

Data Source Name : xlsdsn (anyname)

→ select work book → browse and select → E:\Apps\temp\College.xls → ok

Step(3): Develop java application by using type1 jdbc driver as shown below

```
// ExcelTest.java
import java.sql.*;
public class ExcelTest
{
  public static void main(String args[]) throws Exception
  {
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    Connection con = DriverManager.getConnection("jdbc:odbc:xlsdsn");
                                   (Excel is single user db so no
                                    need to username and password)
    Statement st = con.createStatement();
    ResultSet rs = st.executeQuery("select * from [sheet1$]");
                                   The MS-Excel sheet name as db table name
```

```
while(rs.next())
  {
   sop(rs.getInt(1) + " " + rs.getString(2) + " " + rs.getString(3));
  }
rs.close();
st.close();
con.close();
}//main
}//class
```

CSV - files (cama seperator values) :

E:\apps
- ⟶ temp (Logical DB)
  - ⟶ abc.csv (db table 1)
  - ⟶ Myfile.tab (db table 2)

abc.csv

sno, sname, add ⟶ db table column names			
101	Raja	hyd	db table records
345	Rajesh	hyd	

Myfile.tab

sno	sname	add	⟶ Column names
101	Raja	hyd	db table records
345	Ramesh	hyd	

* The total file system of OS (windows) acts as physical DB s/w, in that each directory acts as one logical DB, in that directory each file that contains formatted data acts as one DB table.

Procedure to communicate with text file (csv file) from java application

step(1): prepare the csv file ready as shown above

E:\NAPPS\Apps\temp\abc.csv

step(2): create DSN for Microsoft text Driver

Control panel → performance and maintainence → Administrative tools → Data Sources → User DSN tab → Add → Microsoft Text driver → Finish

Data source Name : txtdsn (any name)

→ select directory → E:\NAPPS\Apps\temp\ → ok

Step(3): Develop java application by using type-1 driver as shown below.

```
// TextTest.java
import java.sql.*;
class TextTest
{
  public static void main (String args[]) throws Exception
  {
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    Connection Con = DriverManager.getConnection("jdbc:odbc:txtdsn");
    Statement st = Con.CreateStatement();
    ResultSet rs = st.executeQuery("select * from abc.csv");
                                                  filename as db table name
    while(rs.next())
    {
      sop(rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3));
    }
    rs.close();
    st.close();
    Con.close();
  } // main
} // class
```

* In one Jdbc application there can be multiple Connection objects, multiple statement objects and multiple ResultSet objects.
* one JDBC application can interact with multiple DB softwares at a time.

Write a program to transfer the records of MS Excel to oracle DB S/w

Refer Application 13 of Page no 27, 28

Batch process:

* The process of combining related queries into single unit, sending that unit to DB S/w, executing queries of that unit in DB S/w and gathering results of those queries as a single unit is called as "batch processing".

* Batch processing reduces network round trips between java application and database software especially when java application wants to send and execute multiple SQL queries in database software.
* Instead of sending multiple queries to database software individually one by one it is recommanded to take the support of batch process.
* The query is placed in batch will not be executed by enabling do everything or nothing principle (that means if one query execution fails the remaining queries executions should also be failed/rollback).

<u>Procedure to perform JDBC batch processing</u>

(1) create jdbc statement object

```
Statement st = con.createStatement();
```

(2) Add non select queries to batch.

select queries can't be added to the batch of batch processing.

```
st.addBatch("insert into student values(111, 'xxx', 'yyy')");
st.addBatch("update student set sadd = "hyd" where sno = 101");
st.addBatch("update student set add = "vizag" where sname = "Raju"");
st.addBatch("delete from student where sno = 111");
```

→ keep all these SQL queries in a single batch.

<u>NOTE</u>: Any no. of non-select queries pointing to same database table or different database tables can be added to the above batch for batch processing.

(3) send and execute the above batch of queries in database software.

```
int res[] = st.executeBatch();
```

res[]

1	0
1	1
0	2
3	3

This int res[] contains the batch related queries execution results.

* select query execution gives Jdbc resultSet object, executeBatch() returns int[] and ResultSet object can't be stored in this int[] so we can't add select SQL query to the batch of batch processing.

* For example application on batch processing refer page no 31 and application 21.

* The process of combining related operations into a single unit and executing them by applying do everything or nothing principle is comes under transcation management.

Ex: Transfer money task is the combination of 2 operations.

- withdraw amount from source account
- Deposite amount into destination account.

* we need to execute these two operations by applying do everything or nothing principle performing transaction management.

* To perform transaction management in jdbc applications we use batch processing along with some additional logics.

Transaction Management in JDBC:

* The process of combining set of related operations into single unit and executing those operations by applying "do everything or nothing" principle is called as transaction management.

Example application (performing transfer money operation)

SQL> select * from Account.tab;

Ac no	Holdername	Balance
101	Raja	10,000
102	Ravi	30,000

Keep DB table ready as shown in the table.

Develope the application to transfer the funds between two accounts.

```
// TxMgmt.java
import java.sql.*;
import java.util.*;
public class TxMgmt
{
    public static void main(String args[]) throws Exception
    {
```

```
//read input values from key board
scanner sc = new scanner (System.in);
sop( "Enter source account no:");
int srcno = sc.nextInt();
sop("Enter destination account no:");
int destno = sc.nextInt();
sop("Enter the amount to transfer");
int amt = sc.nextInt();
//create jdbc connection object
Class.forName("oracle.jdbc.driver.OracleDriver");
Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:
                                                1521:orcl", "scott", "tiger");
//Disable autoCommit mode(begining of transaction (Tx))
con.setAutoCommit(false);
//perform transfer money operation(through batch processing)
statement st = con.createStatement();
//for withdraw operation on source account
st.addBatch("update Account_tab set balance=balance-"+amt+"where
                                                acno="+srcno);
//for deposite operation on destination account
st.addBatch("update Account_tab set balance=balance+"+amt+"where
                                                acno="+desno);
//execute the batch
int res[] = st.executeBatch();
//write logic to commit or rollback the transaction
boolean flag=false;
for(int i=0; i<res.length; i++)
  {
     if (res[i]==0)
       {
         flag=true;
         break;
       }
  }
if (flag==true)
   {
     con.rollback();
     Sop(" Transaction rolledback");
   }
else
   {
     con.Commit();
```

```
        so( "Transaction Committed");
      }
      st.cl e();
      con.c se();
    } //mai
  }//class
```

* Batch processing is not possible with preparedstatement object and callable statement obje .

<u>working with differ - types of Resultsets</u> :

Resultsets

Non-scrollab

BFR	Resultset object		
101	Raja	d	1
345	Ramesh	yd	2
678	Ramana	yd	3
		LR	

scrollable

BFR	Resultset object		
101	Raja	hyd	1 ↓
345	Ramesh	hyd	2 ←
678	Ramana	hyd	3 ↑
		ALR	

* The Resultset object that allows to access the records sequentially, uni-directionally (top to Hom) is called as non-scrollable object.
* By default every bc resultset object is non-scrollable Resultset object.
* The Resultset obje that allows the programmer to access records randomly or non-se entially or directly or bidirectionally is called as Scrollable Resultset ect.
* Using scrollable R ultset object we can access the records of Result-set object quickly.

<u>To create non-scrol le Resultset object</u>

```
statement s = con.create statement();
ResultSet rs = st.execute Query ("select * from student");
```

↓

Non-sc ble Resultset object

<u>To create scrollable sultset object</u>

```
statement s = con.create statement (type, mode);
ResultSet rs = st.execute Query("select * from student");
```

↓

Sc ble Resultset object.

Possible values for Type (int):

ResultSet. TYPE_SCROLL_SENSITIVE ; (1005)

ResultSet. TYPE_SCROLL_INSENSITIVE ; (1004)

Possible values for Mode (int):

ResultSet. CONCUR_UPDATABLE ; (1008)

ResultSet. CONCUR_READ_ONLY ; (1007)

NOTE: The possible values of type, mode parameters are int constants internally in java.sql.ResultSet interface.

* Public static final member variables of java class/ java interface are called as constants.

* The jdbc statement object that is created without type and mode values can create only non-scrollable ResultSet object. The jdbc statement object that is created having type and mode values can create scrollable ResultSet object.

The methods invokable on non-scrollable ResultSet object

next ();

getXXX ();

getRow ();

close ();

The methods invokable on scrollable ResultSet object

next ();

getXXX ();

getRow();

close();

previous();

beforeFirst(); → keeps cursor in BFR position

afterLast(); → keeps cursor in ALR position
↓
Record pointer.

first(); → keeps cursor in first record

last(); → keeps cursor in last record.

isFirst(); → checks whether cursor is there in first record or not.

isLast(); → checks whether cursor is there in last record or not.

absolute (-/+n);

relative (-/+n);

<u>what is the difference between absolute (-) and relative (-) methods?</u>

* absolute (-) method moves the record pointer in resultset object in forward and reverse direction. positive no. indicates forward direction with respect to first record. Negative no. indicates with respect to last record.

Ex: absolute (3) → moves the cursor to (from first) 3rd record
absolute (-2) → moves the cursor to 2nd record from last.

* Relative (-) method moves the cursor (record pointer) with respect to current position of cursor. positive no. indicates forward direction, negative no. indicates reverse direction.

<u>what is the difference between absolute path and relative path?</u>

```
e:\apps
    |--> Demo
        |--> abc.txt
```

The absolute path of abc.txt file is

e:\apps\Demo\abc.txt

The relative path of abc.txt file from e:\apps folder

.\Demo\abc.txt

The relative path of abc.txt file from e:\apps\xyz folder

..\Demo\abc.txt

The relative path of abc.txt file from e:\apps\demo\abc folder

..\abc.txt

"." → indicates current directory

".." → indicates parent directory

* In relative path the file/directory location will be given with respect to the current working folder location.
* In absolute path the directory or file location will be given completely with respect to right from root directory.

---

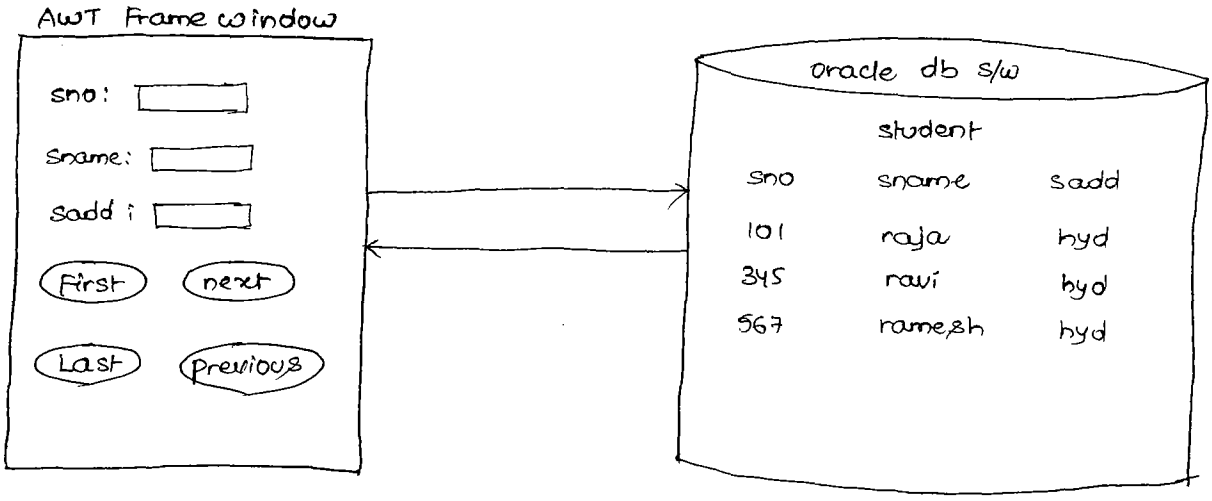
* To display the records of jdbc resultset object bidirectionally (top to bottom and bottom to top) we need to work with scrollable resultset object as shown below.

```
// ScrollTest.java
import java.sql.*;
public class ScrollTest
{
    public static void main(String args[])throws Exception
    {
        // Create jdbc Connection object
        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection Con = DriverManager.getConnection("jdbc:oracle:thin:@localhost
                                                    :1521:orcl", "scott", "tiger");
```

```
// create jdbc statement object with type and mode values
Statement st = Con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
                                   ResultSet.CONCUR_UPDATABLE);
                                   ↳ mode (1008)
                                   ↳ Type (1005
// get scrollable ResultSet object
ResultSet rs = st.executeQuery("select * from student");
// print records in bottom to top order
rs.afterLast();  // keeps cursor in ALR position
while (rs.previous())
  {
    system.out.println(rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3));
  }
// print records in top-bottom order
rs.beforeFirst();  // keeps cursor in BFR position
system.out.println("records... (top to bottom)");
while (rs.next())
  {
    system.out.println(rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3));
  }
// 3rd record is
rs.absolute(3);
system.out.println(rs.getRow()+" record is:"+rs.getInt(1)+" "
                   +rs.getString(2)+" "+rs.getString(3));
// last but 2 record
rs.absolute(-2);
system.out.println(rs.getRow()+" record is:"+rs.getInt(1)+" "
                   +rs.getString(2)+" "+rs.getString(3));
// 4th record from the bottom
rs.relative(-2);
system.out.println(rs.getRow()+" record is:"+rs.getInt(1)+" "
                   +rs.getString(2)+" "+rs.getString(3));
// last ~~but one~~ record
rs.relative(3);
system.out.println(rs.getRow()+" record is:"+rs.getInt(1)+" "+
```

```
        rs.getString(2) + " " + rs.getString(3));
    //The first record is
     rs.first();
     System.out.println(rs.getRow()+"record is:"+rs.getInt(1)+" "
                                +rs.getString(2)+" "+rs.getString(3));
    //the last record is
     rs.last();
     System.out.println(rs.getRow()+"record is:"+rs.getInt(1)+" "
                                +rs.getString(2)+" "+rs.getString(3));
    //close jdbc objects
     rs.close();
     st.close();
     Con.close();
   }//main
 }//class.
```

* Non scrollable Resultset objects are forward only resultset objects by default. That means they allow the programmer to move the cursor in resultset object only in forward direction. (top to bottom)
* All jdbc drivers support scrollable resutsets.



Container → Frame window

Components → label - 3, Button - 4, TextField - 3

LayoutManager → FlowLayout

Event → ActionEvent

EventListener → ActionListener

Event handling method → actionPerformed (-)

* For the above diagram based application refer application 10 of page nos 24-26.

<u>Procedure to develop scrollFrame application (given in previous class) as an awt application by using netbeans IDE.</u>

<u>Step(1):</u> create java project in netbeans IDE

File menu → new → java → java application → next → myproject1 → Finish.

<u>Step(2):</u> Add awt frame to the project

Right click on project → new → other → AWT GUI form → Frame form → next →

Classname: scrollFrame

→ Finish

<u>Step(3):</u> Design frame window as shown in the diagram with awt components.

<u>NOTE:</u> Before performing above step operation change the Layout of frame window from the default BorderLayout to NullLayout

[inspector window → right click on BorderLayout → Free design / NullLayout]

<u>Step(4):</u> Register actionListener on these button components.

Right click on button (any button) → Events → Action → actionPerformed (do this on all 4 buttons)

<u>Step(5):</u> Declare the following instance variables in the source code of ScrollFrame class.

```
Connection Con = null;
ResultSet rs = null;
Statement st = null;
```

Step(6): Develop user defined method in ScrollFrame class having JDBC code to create scrollable ResultSet object.

```
public void makeConnection()
{
  try
   {
     Class.forName("oracle.jdbc.driver.oracleDriver");
     Connection Con = DriverManager.getConnection("jdbc:oracle:thin:
                                    @localhost:1521:orcl", "scott", "tiger");
     st = Con.createStatement (ResultSet.TYPE_SCROLL_INSENSITIVE,
                                    ResultSet.CONCUR_UPDATABLE);
     rs = st.executeQuery ("Select * from student");
   }
  Catch (Exception e)
   {
     e.printStackTrace();
   }
}// makeConnection
```

Step(7): call the above makeConnection method from the Constructor.

```
Public scrollFrame();
      initComponents();
      makeConnection();
```

Step(8): Add following Logic in button1ActionPerformed(-) method (for first button)

```
private void button1ActionPerformed (-)
{
  try
   {
     rs.first();
     textField1.setText (rs.getString(1));
     textField2.setText (rs.getString(2));
     textField3.setText (rs.getString(3));
   }
```

```
catch (Exception e)
  {
    e.printStackTrace();
  }
```

step (9): write following code in button2ActionPerformed (-) method.

(for next button)

```
private void button2ActionPerformed (-)
  {
    try
      {
        if (!rs.isLast())
          {
            rs.next();
            textField1.setText (rs.getString(1));
            textField2.setText (rs.getString (2));
            textField3.setText (rs.getString (3));
          }// if
      }// try
    catch (Exception e)
      {
        e.printStackTrace();
      }
  }
```

step (10): write following code in button3ActionPerformed (-) method

(for previous button)

```
private void button3ActionPerformed (-)
  {
    try
      {
        if (!rs.isFast())
          {
            rs.previous();
            textField1.setText (rs.getString (1));
            textField2.setText (rs.getString (2));
            textField3.setText (rs.getString (3));
          }// if
      }// try
    catch (Exception e)
      {
        e.printStackTrace();
      }
  }
```

step(11): write the following code in buttonyActionPerformed (-) method.
(for Last button)

```
private void buttonyActionPerformed (-)
{
  try
  {
    rs.last();
    textField1.setText (rs.getString (1));
    textField2.setText (rs.getString (2));
    textField3.setText (rs.getString (3));
  }//try
  catch (Exception e)
  {
    e.printStackTrace ();
  }
}
```

step(12): Add ojdbc14.jar file to the libraries of the project.

Right click on libraries folder → Add jar → browse and select ojdbc14.jar file

step(13): Run the application.

Right click in the source code of ScrollFrame.java → Run file.

* A scrollable ResultSet object can also be developed having some additional behaviors like SensitiveResultSet, InsensitiveResultSet, UpdatableResultSet, ReadonlyResultset.

/what is the difference between SensitiveResultSet object and Insensitive_ResultSet object?

* when ResultSet object is representing db table data if modifications done in db table are immediately reflected to ResultSet object then it is called as SensitiveResultSet object. otherwise it is called as InsensitiveResultSet object.

To create SensitiveResultSet object

```
statement st = con.createStatement (ResultSet.TYPE_SCROLL_SENSITIVE,
                                    ResultSet.CONCUR_UPDATABLE);
```

```
ResultSet rs = st.executeQuery("select * from student");
```

To create InSensitiveResultSet object

```
statement st = Con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
                                   ResultSet.CONCUR_UPDATABLE);
ResultSet rs = st.executeQuery("select * from student");
```

* Type1 Jdbc driver supports sensitiveResultSets but not InSensitiveResult sets.
* Oracle thin driver, oci driver, Connector/j driver of mysql, postgresql thin driver supports INSENSITIVE Resultsets but not SensitiveResultSets.
* For example application on sensitive and InSensitive ResultSets refer application 15 of page no.s 28 and 29.
* Don't use sensitive and Insensitive ResultSet object features in your jdbc based projects because no driver is supporting these sensitive and Insensitive behaviours properly.

Creating Scrollable ResultSet objects by using PreparedStatement object

```
PreparedStatement ps = Con.prepareStatement("select * from student",
                                            ResultSet.TYPE_SCROLL_SENSITIVE,
                                            ResultSet.CONCUR_UPDATABLE);
ResultSet rs = ps.executeQuery();
```

* we can use simple statement object or PreparedStatement object to work with scrollable ResultSet object.

/ what is the difference between readonly resultset object and updatable result set object?

* If the modifications done in resultset object are reflecting immediately to database table that result set object is called updatable resultset object. Through updatable result set object we can perform all the four CURD operations on the database table.
* The ResultSet object that allows only read operations on database table is called as readonly resultset object.

what is the difference between sesitive resultset object and updatable resultset object?

* In sensitive resultset object the modifications done in database table will reflect to resultset object whereas in updatable resultset object the modifications done in resultset object will reflect to database table.

To prepare scrollable Resultset object as Readonly Resultset object

```
statement st = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
                                   ResultSet.CONCUR_READ_ONLY);
ResultSet rs = st.executeQuery("select * from student");
```

To prepare scrollable Resultset object as updatable Resultset object

```
Statement st = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
                                   ResultSet.CONCUR_UPDATABLE);
ResultSet rs = st.executeQuery("select * from student");
```

* All non-scrollable resultset objects are Readonly Resultset objects by default.
* we can perform both select and non-select operations on database table by using updatable resultset object as shown below

To perform select operation

```
while (rs.next())
  {
    system.out.println(rs.getInt(1) + " " + rs.getString(2) + " " + rs.getString(3));
  }
```

To insert new record

```
rs.moveToInsertRow();  → creates an empty record
rs.updateInt(1, 567);
rs.updateString(2,"rajesh");   } sets values to the columns of the empty
rs.updateString(3,"hyd");      } record.
rs.insertRow();  ⟶ inserts records in database table
```

To update the existing record

```
rs.absolute(4);  → moves to 4th record in ResultSet object
rs.updateString(2,"rakesh");  → modifying 2nd column value of 4th record.
```

rs.updateRow(); → updates the record in database table.

To delete the existing record

```
rs.absolute (3);
rs.deleteRow ();
```

: Using updatable Resultset object we can perform insert, update and delete operations on database table without using the respective sql queries.

* Limitations with updatable Resultset objects *

(1) Very few jdbc drivers support updatable Resultsets.

(2) we can not perform criteria based non-select operations on database table.

(3) Bulk non-select operations are not possible.

NOTE: only Type-1 jdbc driver is giving support for updatable result set objects. Oracle thin driver, oci driver, mysql Connector/j driver, postgresql thin driver are not supporting updatable Resultsets.

* All jdbc drivers are giving support for Readonly Resultset objects.

Summary table on jdbc Resultset objects

Type-1 driver

Support for Non-Scrollable Resultset : yes
Support for Scrollable Resultset : yes
Support for Readonly Resultset : yes
Support for Sensitive Resultset : yes
Support for Insensitive Resultset : no
Support for updatable Resultset : yes

Oracle thin / oci / Connector-j driver for mysql / postgresql thin Driver

Support for Non-Scrollable Resultset : yes
Support for Scrollable Resultset : yes
Support for Readonly Resultset : yes
Support for Sensitive Resultset : no
Support for Insensitive Resultset : yes
Support for Updatable Resultset : no

* For example application on updatable Resultset object refer application 18 of page no. 30.

Rowsets:

* Jdbc Resultset objects are not serializable objects so we can not send those objects over the networks.
* To solve this problem we can use Rowsets as alternate for Resultsets.
* Jdbc Rowsets are serializable objects by default.
* Rowset object means it is the object of a java class that implements javax.sql.Rowset interface. This interface is sub interface of java.sql.Resultset interface.
* we can develop Rowsets in three modes.

(1) As cached Rowset → It is a disconnected Rowset.
→ It is like insensitive Resultset object.

(2) As Jdbc Rowset → It is a connected Rowset.
→ It is like sensitive Resultset object

(3) As web Rowset → It is a connected Rowset.
→ Allows to read data from xml files.

* Oracle corporation gives support for two types of Rowsets (cached Rowset, Jdbc Rowset). For this it supplies oracle.jdbc.rowset.OracleCachedRowset class and oracle.jdbc.rowset.OracleJdbcRowSet class. These two classes are available in ojdbc14.jar file of oracle 10g. whereas in oracle 9i these two classes are available in ocrs12.jar file and for this jar file ojdbc14.jar file is dependent jar file.
* working with Rowsets is always an easy process because it allows getter and setter methods based programming.
* For example application on Rowsets refer application 19 and 20 of page nos 30 and 31.

Advantages with Rowsets

1) Gives simplified programming (allows single object based setter and getter methods oriented programming).

2) Rowsets are serializable objects so they can be sent over the network.

Disadvantages

(1) very few drivers are supporting Rowsets.

(2) Non-select operations on database table using Rowsets is not possible.

NOTE: All Collection frame work data structures are serializable objects by default. Since very few jdbc drivers are giving support for Rowsets the programmers copy ResultSet object records to collection frame work data structure (like arrayList) and sends that collection frame work data-structure over the network.

* If classes of certain jar file uses the classes and interfaces of certain other jar files then the other jar files are called as dependent jar files to the first main jar file.

Ex: The classes of ocrs12.jar file internally uses the classes of ojdbc14.jar file so we can say ojdbc14.jar file dependent jar file to ocrs12.jar.

Working with Date Values

* While dealing DOB, Date of joining, bill purchase date and etc values we need to see jdbc persistance logic performing insert, update, delete and select operations on date values.

* Don't insert Date values in database table columns as String values because we can't apply the Date related sql functions on those Date values.

* It is always recommanded to insert the date values in "date" data-type columns of database table.

* Different database softwares support different patterns of Date values like oracle supports → dd-mmm-yy pattern (ex: 20-jan-99)

mysql supports ⟶ yyyy-MM-dd pattern (ex: 1999-10-20)

* once java.sql.Date class object is given to jdbc driver representing Date value then the Jdbc driver inserts that date value in the database table column in the pattern that is supported by underlaying database software.

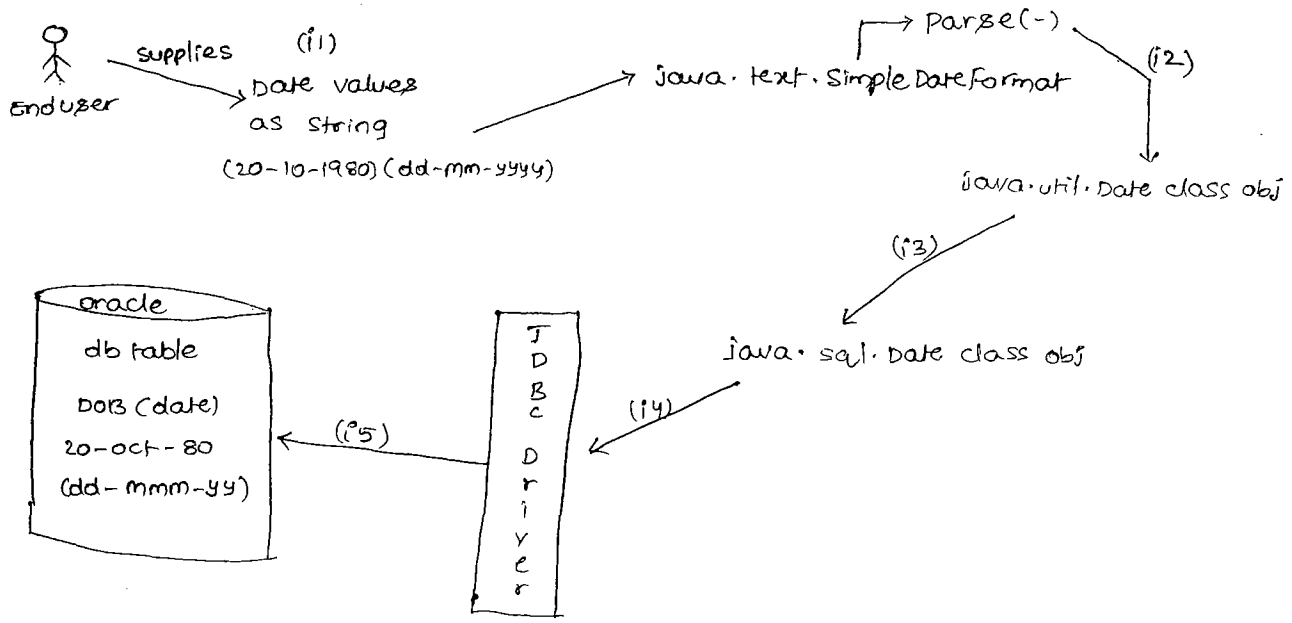
<u>wrong database table design</u>

name	dob
(vc2)	(vc2)
raja	20-10-1982
ramesh	30-10-1987

<u>Good database table design</u>

name	dob
(vc2)	(date)
raja	20-oct-1982
ramesh	30-oct-1987

<u>Database software and Jdbc driver independent logic to insert Date value in date data type column of database table in the pattern that is supported by underlaying database software.</u>



with respect to diagram

i1 – End user supply string datavalue to application

i2 – The parse(-) method of simple Date format class converts string date value to java.util.Date class object.

i3 – Application converts java.util.Date class object to java.sql.Date class object.

i4 – Application gives this java.sql.Date class object to jdbc driver using PreparedStatement object

i5 – This jdbc driver inserts the date value to database table column in the pattern it is supported by database software.

What is the difference between java.util.Date class and java.sql.Date class?

java.util.Date	java.sql.Date
* This class object represents ordinary date values. but it can't represent db table column date values.	* Represents db table ~~date~~ column date values.
* Jdbc driver ~~the~~ can't recognize the date value of this class object.	* Jdbc driver can ~~repr~~recognize the date value of this class object.

* java.sql.Date is the sub class of java.util.Date.

```
//Converting string date value to java.util.Date class object
String s1 = "20-10-99"; // dd-MM-yy
SimpleDateFormat sdf1 = new SimpleDateFormat("dd-MM-yy");
java.util.Date ud1 = sdf1.parse(s1);
sop("util date is:" + ud1.toString());
```

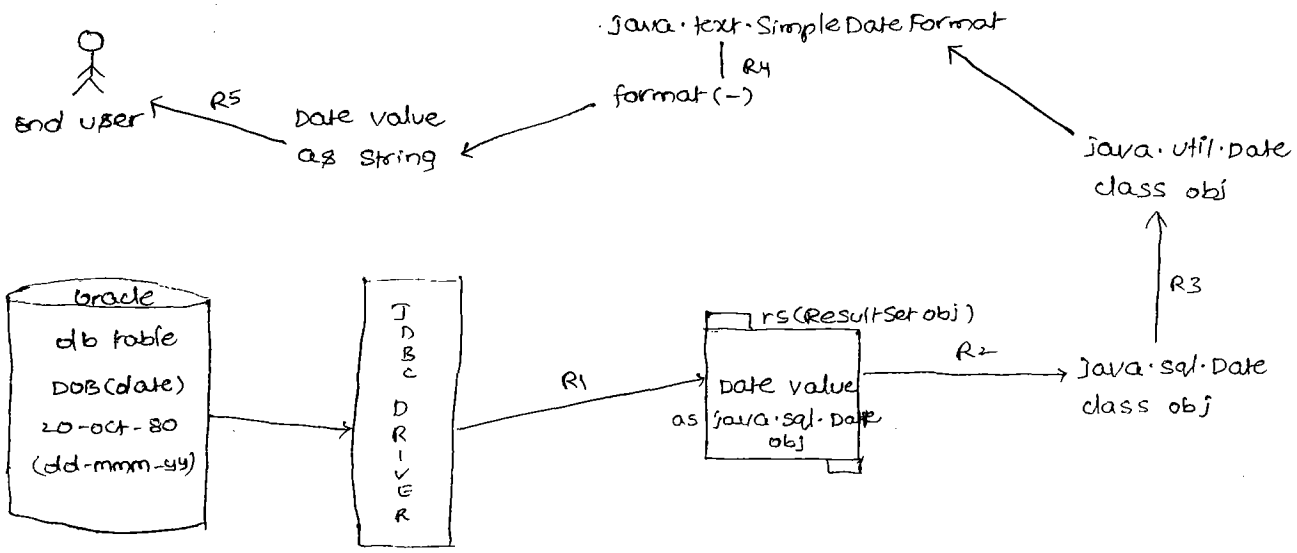
The date pattern of input date value

```
// Converting java.util.Date class obj to java.sql.Date class object
long ms = ud1.getTime(); ①
java.sql.Date sqd1 = new java.sql.Date(ms);
sop("sql Date is:" + sqd1.toString());
```

① - gives no. of milliseconds that are elapsed between january 1st 1970 00:00 hours (Epoch standard) and the date and time represented by udl object (java.util.Date class object)

```
//Converting String date value directly to java.sql.Date class object
// (note: the String value must be there in yyyy-MM-dd pattern)
String s2 = "1982-10-20"; //yyyy-MM-dd
java.sql.Date sqd2 = java.sql.Date.valueOf(s2);
sop("the sql date is:" + sqd2.toString());
```

The db s/w independent and jdbc driver independent procedure to retreive dates from database table column



R1 - Application executes select query on database table and gets jdbc resultset object having date value

R2 - Application retreives date value from ResultSet object as java.sql.Date class object.

R3 - Application converts java.sql.Date class object to java.util.Date class object.

R4 - The format(-) method of simpleDateFormat class converts java.util.Date class object to string date value in the pattern that is expected by end user.

R5 - Application displays string date value for end user.

```
// Converting java.sql.Date class obj to java.util.Date class obj
        java.util.Date ud2 = (java.util.Date) sqd2;
        sop("the util date value is:" + ud2.toString());
// Converting java.util.Date class object to string date value
    SimpleDateFormat sdf2 = new SimpleDateFormat("MMM-yy-dd");
```

Expected date pattern of o/p string date value

```
        String s3 = sdf2.format(ud2);
        sop("String date value is:" + s3);
```

Example application to insert and retrieve data values

DB Table (oracle)

```
create table person_tab (no number, name varchar2(20), DOB date,
                                                    DOJ date);
```

DB Table (mysql)

```
create table person_tab (no int(5), name varchar(20), DOB date,
                                                    DOJ date);
```

DateInsert.java

```
import java.util.*;
import java.sql.*;
import java.text.*;
public class DateInsert
{
    public static void main(String args[]) throws Exception
    {
        // read input values from key board
        Scanner sc = new Scanner(System.in);
        sop("Enter person number:");
        int no = sc.nextInt();
        sop("Enter person name:");
        String name = sc.next();
        sop("Enter person date of birth");
        String pdob = sc.next();
        sop("Enter person date of joining");
        String pdoj = sc.next();
```

```
//convert string date values to java.sql.Date class object.
// for Date of Birth
  SimpleDateFormat sdf1 = new SimpleDateFormat("dd-mm-yy");
  java.util.Date udob = sdf1.parse(pdob);
  java.sql.Date sqdob = new java.sql.Date(udob.getTime());
// for Date of joining (yyyy-MM-dd)
  java.sql.Date sqdoj = java.sql.Date.valueOf(pdoj);
// create Jdbc Connection object
  class.forName("oracle:jdbc:driver:OracleDriver");
  Connection con = DriverManager.getConnection("jdbc:oracle:thin:@local
                                                host:1521:ord","scott","tiger");
// create Prepared statement object pointing to insert query
  PreparedStatement ps = con.prepareStatement("insert into person_tab
                                                values(?,?,?,?)");
// set values to the parameters of query
  ps.setInt(1,no);
  ps.setString(2,name);
  ps.setDate(3,sqdob);
  ps.setDate(4,sqdoj);
// execute the query
  int res = ps.executeUpdate();
// processing the result
  if (res==0)
    sop("record not inserted");
  else
    sop("record is inserted");
// close jdbc objects
    ps.close();
    con.close();
}//main
}//class
//javac DateInsert.java
//java DateInsert
```

* using simple statement object we cannot insert date, large object values (files in database table). But we can do this work by using PreparedStatement object.

Application

```
//DateRetrive.java
import java.sql.*;
import java.util.*;
import java.text.*;
public class DateRetrive
{
  public static void main(String args[]) throws Exception
  {
    //create jdbc connection object
    class.forName("oracle.jdbc.driver.OracleDriver");
    Connection con = DriverManager.getConnection("jdbc:oracle:thin:
                                    @localhost:1521:orcl","scott","tiger");
    //create jdbc statement object
    Statement st = con.createStatement();
    // execute the query
    ResultSet rs = st.executeQuery("select * from person_tab");
    // process the resultset
    while (rs.next())
      {
        int no = rs.getInt(1);
        String name = rs.getString(2);
        java.sql.Date sqdob = rs.getDate(3);
        java.sql.Date sqdoj = rs.getDate(4);
        // Convert java.sql.Date class object to java.util.Date class
        //object
        java.util.Date udob = (java.util.Date) sqldob;
        java.util.Date udoj = (java.util.Date) sqdoj;
        // Convert java.util.Date class objects to String class values
        SimpleDateFormat sdf1 = new SimpleDateFormat("MMM-yy-dd");
        String pdob = sdf1.format(udob);
        SimpleDateFormat sdf2 = new SimpleDateFormat("yyyy-dd-MMM");
        String pdoj = sdf2.format(udoj);
```

```
        sop("no " + name + " " + pdob + " " + pdoj);
      }//while
      //close jdbc objects
        rs.close();
        con.close();
        st.close();
    }//main
  }//class.
```

* Don't hard code any values in your java application that are changeble in the future, pass them to application from outside the application by taking the support of properties file.
* To make jdbc code of java application as flexible code to modify collect the following details from outside the application by taking the support of properties file the following
    * Jdbc classname
    * URL
    * Database username
    * database password
    * sql query

```
E:\
  ↳ Napps
      ↳ Apps
          ↳ adv.java
              ↳ jdbc
                  ├→ properties.java
                  ├→ myfile.properties
```

<u>properties file</u>

```
# jdbc details
  my.driver = oracle.jdbc.driver.OracleDriver
  my.dburl = jdbc:oracle:thin:@localhost:1521:orcl
  my.dbuser = scott
  my.dbpwd = tiger
  my.sql = select * from student
```

→ save myfile.properties

```
//PropTest.java
import java.sql.*;
import java.util.*;
import
public class PropTest
{
    public static void main(String args[]) throws Exception
    {
      //Locate text properties file
      FileInputStream fis = new FileInputStream("myfile.properties");
      //create an empty object of java.util.Properties class
      Properties p = new Properties();
      //load the content of myfile.properties into java.util.Properties
      //class
       p.load(fis);
      //read values from java.util.Properties class of object (p)
       String dname = p.getProperty("my.driver");
       String url = p.getProperty("my.dburl");
       String user = p.getProperty("my.dbuser");
       String pwd = p.getProperty("my.dbpwd");
       String query = p.getProperty("my.sql");
      //create jdbc connection object
       Class.forName(dname);
       Connection con = DriverManager.getConnection(url, user, pwd);
      //create statement object
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery(query);
       while(rs.next())
        {
         sop(rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3)+" "
                                                  +rs.getString(4));
        }
       //close jdbc objects
        rs.close();
        con.close();
        st.close();
    }
```

* In database table files are called as large objects.
* There are two types of large objects
  (1) BLOB → Ex: image files, avi files
  (2) CLOB → Ex: text files, rich text files, ms word documents
* matrimony, job portal applications we must deal with these large objects.
* To insert LOB values to database table use Jdbc PreparedStatement object and take column data types has BLOB (or) CLOB.

<u>standard procedure to insert LOB values to database table column</u>

1. create inputstream object pointing to file

```
FileInputStream fis = new FileInputStream("c:\abc\ash.gif");
```

2. create PreparedStatement object pointing insert query based on jdbc con object

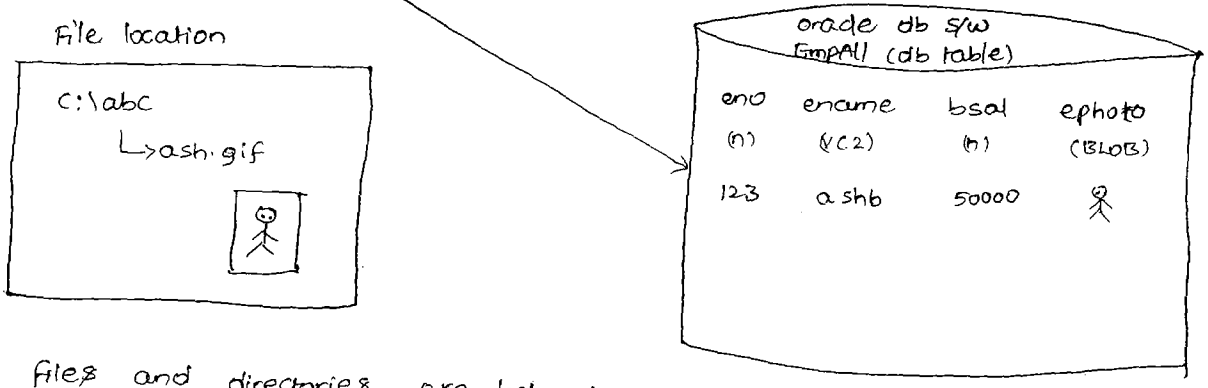
```
PreparedStatement ps = Con.PrepareStatement("insert into EmpAll values(?,?,?,?);
```

3. set values to parameters of query

```
ps.setInt(1,123); ps.setString(2,"ashb"); ps.setInt(3,50000);
ps.setBinaryStream(4, fis.length);
```

4. Execute the query

```
ps.executeUpdate();
```



* All files and directories are belonging to various drives together is called as File system.
* To binary large object values to query we can use setBinaryStream(3P) (or) setBlob(3P) method.
* Similarly to set character large object values to sql query we can use

setCharacterStream ( ) or setClob() methods.

* For example application on working with large object refer application 23 and 24 of handBook.

<u>Standard procedure to follow to retrive large object values from database table column</u>

<u>Step(1):</u> Execute select query on database table and get Jdbc ResultSet object.

```
Statement st = con.createStatement();
ResultSet rs = st.executeQuery("select * from EmpAll");
```

<u>Step(2):</u> Get access ephoto value from db table

```
rs.next();
FileInputStream fis = rs.getBinaryStream("ephoto");
```

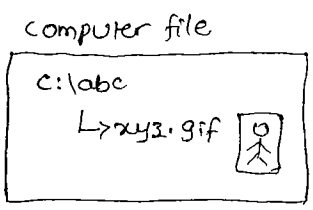
↑ Input stream object pointing photo db table column through ResultSet object

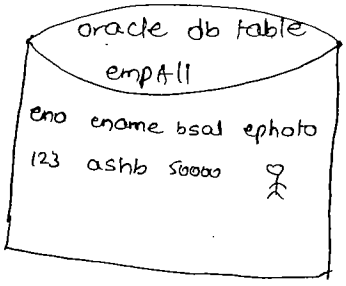
<u>Step(3):</u> create FileOutputStream object pointing to destination file (xyz.gif)

```
FileOutputStream fos = new FileOutputStream("c:\\abc\\xyz.gif");
```

<u>Step(4):</u> write db table column photo to the destination file (xyz.gif) using both Input, output stream objects.

```
int bytesRead=0;
byte[] buffer=new byte[4096];
while (bytesRead = fis.read(buffer)! = -1)
  {
    fos.write(buffer, 0, byteRead);
  }
```

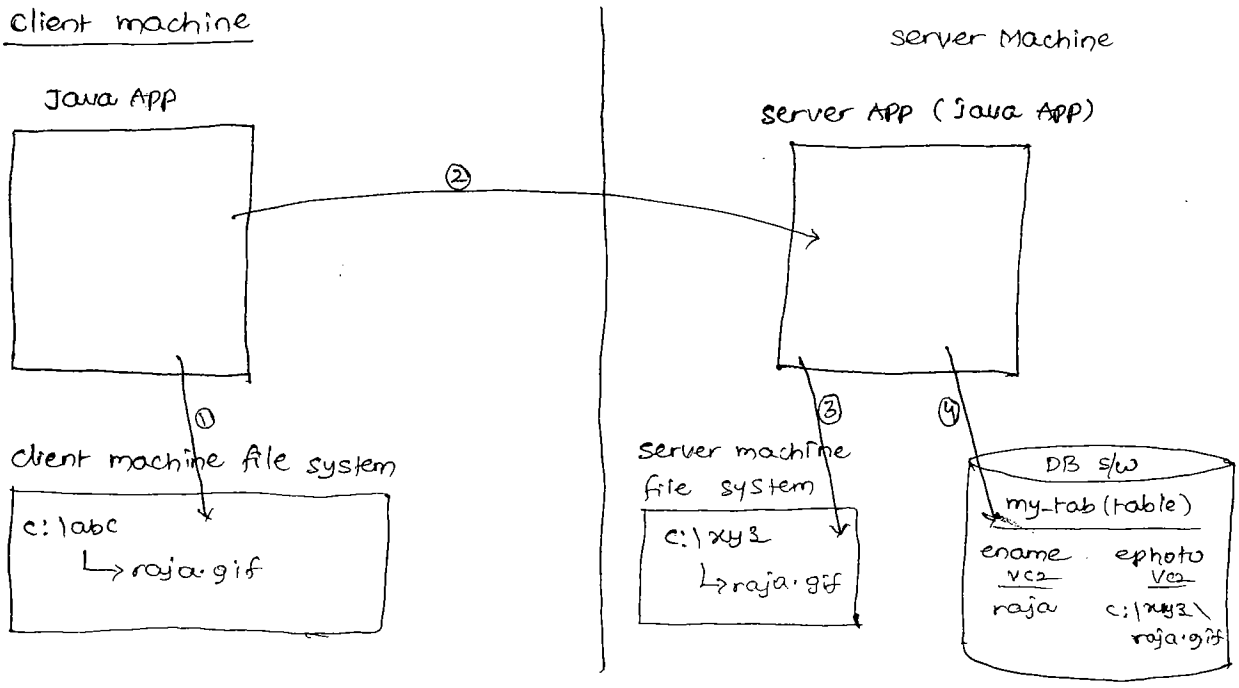




* Buffer is temporary memory to store temporary data.
* The process of storing data in buffer before sending to final destination is called as buffering.
* while transfering source file data to destination file if buffer support

is taken, we can reduce no. of read and write operations on source and destination files respectively.

* In the above code the byteRead variable keeps track of the no. of bytes that are stored into buffer each time by reading the content from source file (image of resultset object) so that we can use that data value while writing buffer content to destination file.
* To read BLOB value from ResultSet object use rs.getBinaryStream() or rs.getBlob() method.
* To read clob value from ResultSet object use either rs.getcharacterStream() or rs.getclob() method.
* In real time projects no one prefers to store large object value like image files text files etc... directly in database table column because they may degrade performance of database software, by allocating memory upto 4GB.
* In industry level projects instead of storing LOB files directly as VB table column values they store these files in computer file system/directories of hard disk drives and they will store the paths of these files as string values in string data type table columns.



* The process of sending file from client machine to server machine is called as file uploading and reverse is called as file downloading.

<u>Sample senario to know how realtime projects manages large objects (towards insertion)</u>

(1) client application locates and selects the file from client machine file system.

(2) client application sends the file to server application through file uploading process.

(3) server application writes / saves the received file to the file system of server machine.

(4) server application writes the path of the file on server machine as db table column value.

* Jdbc connection pool is a factory that contains set of readly available jdbc connection objects before actually being used.

* The advantage with jdbc connection pool is we can use minimum no. of connection objects to give service to maximum clients.

* There are two types of jdbc connection pools.

① JDBC Driver managed / stand alone Jdbc Connection

② web server / application server managed jdbc connection.

Jdbc connection pool

1. Jdbc driver managed / stand alone jdbc con pool (Resides outside the server)

2. web server / application server managed jdbc connection pool (Resides inside the server)

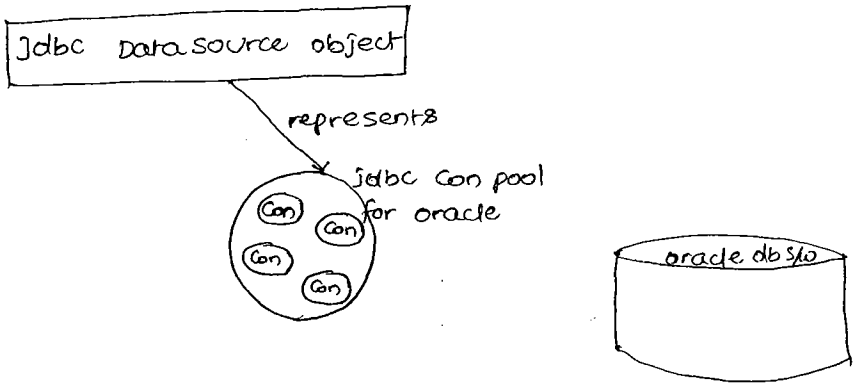
* while developing small scale applications like stand alone, desktop application we can use driver managed jdbc connection pool.

* while developing large scale applications like web sites and enterprise applications we can use server managed jdbc connection pool.

* All jdbc connection objects in a jdbc connection pool represents connectivity to same database software.

* Jdbc connection pool for oracle means all jdbc connection objects in that connection pool represents connectivity with oracle database software.

* Oracle thin driver/ oci driver gives one built-in jdbc Connection pool for oracle.
* Jdbc data ~~stores~~ source object represents jdbc connection pool so our application should always get a Jdbc Connection object from Connection pool through data source object



* Jdbc data source object means it is the object of a java class that implements javax.sql.DataSource (I).
* The jdbc Connection object that is created by the programmer manually is called as direct Jdbc Connection object.
* The jdbc Connection object that is collected from jdbc Connection pool is called as pooled Jdbc Connection object.

<u>Example application on oracle thin driver managed jdbc Connection pool</u>

```
//ConnectionPoolTest.java
 import java.sql.*;
 import javax.sql.*; // for pooledConnection (I)
 import oracle.jdbc.pool.*; // (for oracle Connection pool DataSource (c))
 public class ConnPoolTest
  {
    public static void main(String args[]) throws Exception
     {
      //create jdbc data source object representing empty driver managed
      //connection pool of oracle
       OracleConnectionPoolDataSource ds = new OracleConnectionPoolDataSource();

      //gives details to create jdbc Connection objects in the connection pool
       ds.setDriverType("thin");
```

```
ds.setServerName("localhost");
ds.setPortNumber(1521);
ds.setServiceName("orcl");
ds.setUser("scott");
ds.setPassword("tiger");
//gets Access to jdbc connection pool for oracle (this connection
                pool will be created based on the above details)
pooledConnection con = pcon.getConnection();
//write jdbc based persistance logic
  Statement st = con.createStatement();
  ResultSet rs = st.executeQuery("select * from student");
  while (rs.next())
   {
    sop(rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3));
   }
  //close jdbc objects
  rs.close();
  st.close();
  con.close(); // releases connection object back to jdbc con pool.
}//main
}// class
//add ojdbc14.jar file to classpath and then execute.
```

* scrollable ResultSet, sensitive ResultSet, updatable ResultSet features are given from jdbc 2.0 onwards.
* Drivermanager connection pool save point connections are given from jdbc 3.0 onwards.
* Auto loading of driver class annotations based jdbc programming features are given from jdbc 4.x onwards.
* To work with Drivermanaged jdbc connection pool there is no necessity of dealing with type3 protocol. while working with servermanager connection pool we must deal with type3 protocol.
- Savepoint is a point within the current transaction that can be referenced from the Connection.rollback method.

* When a transaction is rolled back to a save point point all the changes made after that savepoint point will be under.
* After executing 10 queries if you want to commit first 5 query results and if you want to rollback last 5 query results then create save point after first 5 queries and rollback upto that savepoint position.
* There are two types of save points that can be created in jdbc environment.

  (1) named savepoint

  (2) unnamed savepoint

<u>NOTE:</u> For unnamed savepoint the db s/w automatically generates an identity value.

* In jdbc queries based transaction save point based rollbacks are possible but save point based commits are not possible.

```
//SavepointTest.java
import java.sql.*;
public class SavepointTest
  {
    public static void main(String args[]) throws Exception
     {
        //create jdbc connection object
        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost
                                                    :1521:orcl", "scott", "tiger");
        Statement st=Con.createStatement();
        //Begin Tx
        con.setAutoCommit(false);
        //query1
        st.executeUpdate("insert into student values (671, 'xyz', 'hydl')");
        //set a named savepoint to Connection object
        Savepoint svpt=con.setSavepoint("mysp"); //mysp is savepoint
                                                   name
        //query2 in savepoint area
        st.executeUpdate("update student set name='mahesh2' where
                                                   sno=345");
```

```
//rollback upto savepoint  con.rollback(svpt);
//Con.Commit(svpt); // error
//Commit Tx
Con.Commit();
//record insertion will be Committed (query1) but record
//updation (query2) will be rollbacks because query2 is
//there in save point area.
}//main
}//class
//> javac SavepointTest.java
//> java SavepointTest
```

* To work with jdbc 3.0 features we need. jdk 1.5 or higher version similarly to work with jdbc 4.0 features we need jdk 1.6 or higher version setup.

* The auto driver class loading feature of jdbc 4.0 can load the make the application loading Driver class automatically in one of the following two situations.

(1) If Driver class is specified in META-INF/service/java.sql.Driver file of application ~~directory~~ related work directory.

(2) If jar file that represents jdbc driver contains java.sql.Driver file having jdbc driver class name.

* ojdbc14.jar does not contains java.sql.Driver file so you must arrange that file explicitly as shown below to make application loading oracle thin Driver class automatically as shown below.

<u>step(1):</u> create java.sql.Driver file

```
E:\....\APP1
    |--> META-INF
        |--> services
            |--> java.sql.Driver (file)
```

<u>java.sql.Driver (file)</u>

oracle.jdbc.driver.OracleDriver

step(2): Develop java application as shown below

```
//SelectTest.java
import java.sql.*;
public class SelectTest
{
    public static void main (String args[]) throws Exception
    {
        //do notice that there is no class.forName(-)
        Connection con = DriverManager.getConnection ("oracle.jdbc:oracle:
                                                thin:@localhost:1521:orcl", "Scott", "tiger");

        if (con != null)
            sop ("Connection is established");
        if (con == null)
            sop ("connection is not established");
    } //main
} //class
//>javac SelectTest.java
//> java SelectTest
```

Example(2):

* The postgresql related thin Driver jar file is having java.sql.Driver file having the Driver class name. By just adding this jar file to class path we can make our java applications loading jdbc driver class automatically.

```
E:\ ---- \APP2
            |-> SelectTest.java
```

```
//SelectTest.java
import java.sql.*;
public class SelectTest
{
    public static void main (String args[]) throws Exception
    {
        //do notice that there is no class.forName(-)
        Connection con = DriverManager.getConnection("jdbc:postgresql:
                                                postgres", "postgres", "postgres");
```

```
        if (con != null)
            sop("connection established");
        else
            sop(connection not established");
    }// main
}// class
// > javac Select-Test.java
// > java Select-Test.java
```

* Before executing above example add postgresql-8.4-701.jdbc4.jar file to class path.

——— * * * ———