CLASS-1

JDBC
=====
As if now it is known as trademark.
But earlier it is known as Java Database Connectivity.
RAM is a temperory storage device or medium.
During the program execution our data will store in RAM.
Once the program execution is completed we will loss the data.
To overcome this limitation we are making our applications writing the data into the files or database softwares.
Files and Database softwares act like a permanent storage device or medium.

Persistence

The process of storing and managing the data for a long period of time is called persistence.

import terminologies
==========
persistence store

It is a place where we can store and manage the data for a long period of time.

ex:
Files and Database software's.

persistence data

Data of a persistence store is called persistence data.

ex:
Records
tables

persistence operation

Insert,Update,Delete,Create and Select are called persistence operations.
In realtime this operation is also known as CURD operation,CRUD operation or SCUD operation.

ex:

C - Create	S - Select
U - Update	C - Create
R - Read	U - Update
D - Delete	D - Delete

persistence logic

A logic which is capable to perform persitence operations is called persistence logic.
ex:
JDBC code
Hibernate code
Ibatis code
IOStream

persistence technology

A technology which is used to develop persistence logic is called persistence technology.

ex:

JDBC
Hibernate
JPA
and etc.

Q)What is JDBC?

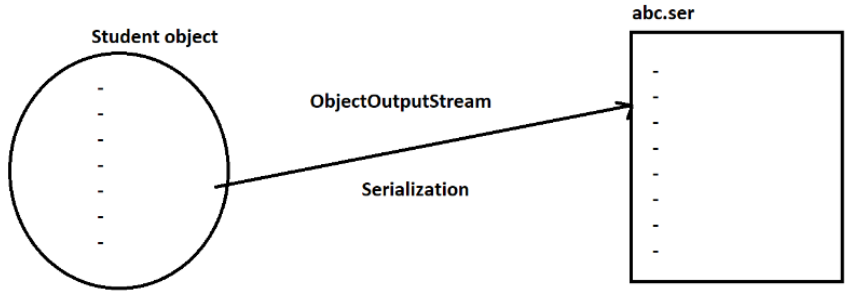
JDBC is a persistence technology which is used to develop persistence logic having the capability to perform persistence operations on persistence data of a persistence store.

Note:

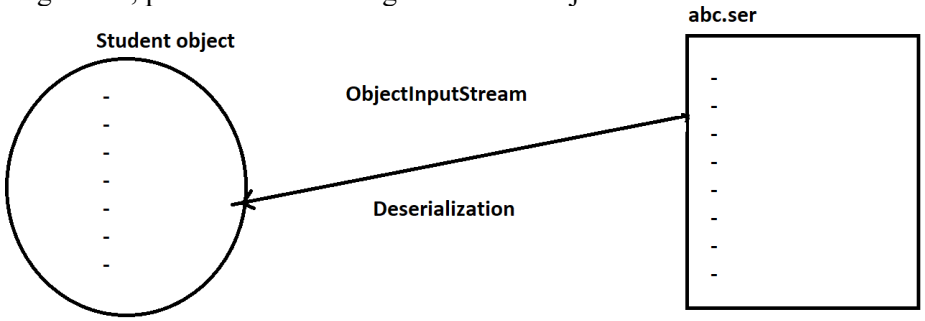
IOStream
JavaApp ---------- Files
Serialization/Deserialization

JDBC Code
JavaApp ---------- DB s/w

Serialization
==========
The process of converting from one state to another state is called serialization.
In general , process of converting object state to file state is called serialization.
In serialization ,object will not store in a file.Object data will store in a file.



Deserialization
==========
The process of taking the data from the file and representing an object is called deserialization.
In general , process of converting file state to object state is called Deserialization.



Limitations with file as persistence store
==========
> We can store limited amount of data.
> There is no security.
> Fetching the data with multiple conditions is not possible.

> It does not show an application with relationships (1-1,1-m,m-2,m-m).
> It does not allows us to apply constraints (not null,unique,primary key,foreign key, check).
> Updation and Deletion of data can't be done directly.
> Merging and comparision of data can't be done easily.

Advantages of database as persistence store

==========

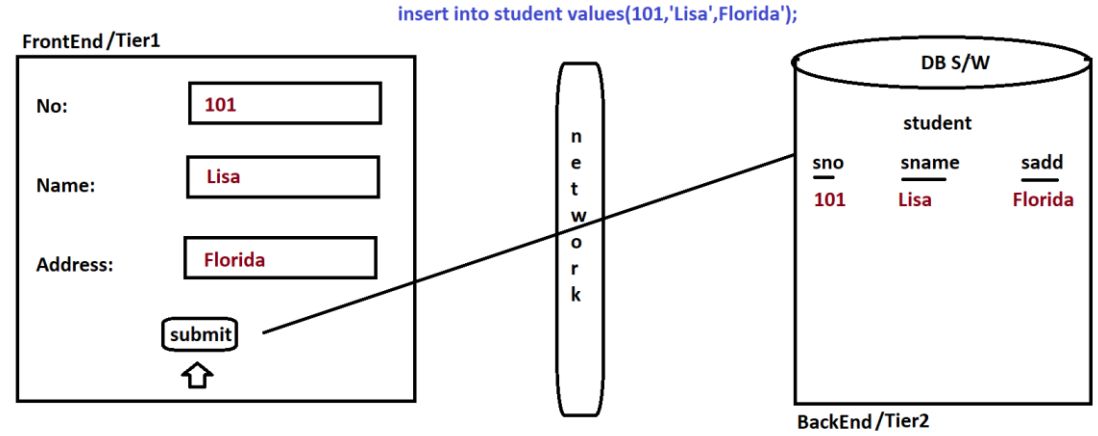
> We can store unlimited amount of data.
> There is a security.
> It supports common query language.
> Fetching the data with multiple conditions is possible.
> It shows an application with relationships.
> It allows us to apply constraints.
> Updation and Deletion of data can be done directly.
> Merging and comparision of data can be done easily.

Every JDBC application is a two-tier application.Where java with JDBC code acts like a frontend/tier1/layer1 and database software acts like a backend/tier2/layer2.

The one which is visible to the enduser to perform some operations is called frontend.

The one which is not visible to the enduser but it performs some operations based on the instructions given by frontend is called backend.

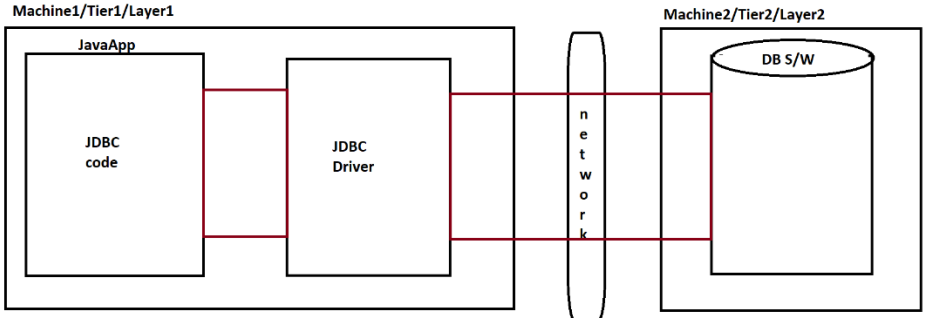
Enduser is a non-technical person,They can't prepare and execute SQL query in database software.They depends upon frontend developers having the capability to do that work them.



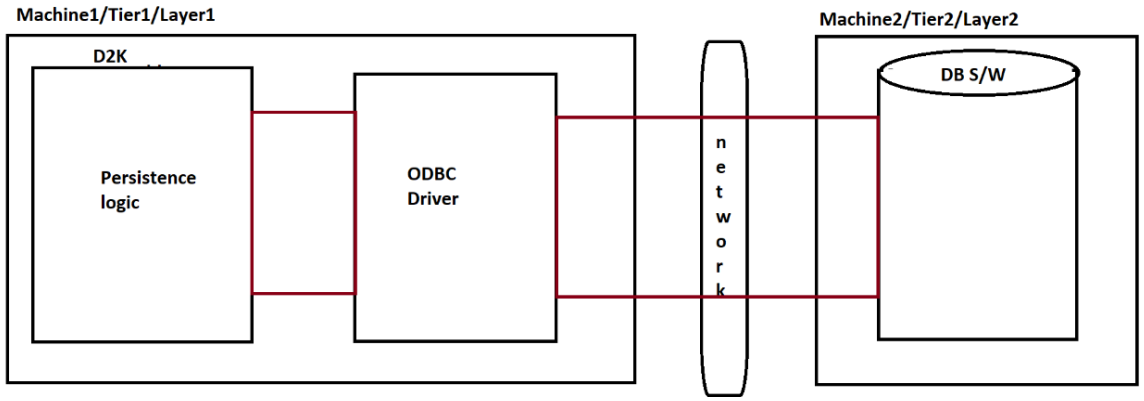
JDBC Driver

==========

JDBC driver acts like a bridge between java application and database software.
IT is used to converts java calls to database calls and vice versa.
Here calls means instructions.



ODBC Driver
==========
Perl,VB2,D2k and etc uses ODBC drivers to locate and communicate with database software.



ODBC drivers developed in c langauge by taking the support of pointers.
But java does not support pointers.To overcome this limitation sun micro system company introduced JDBC drivers exclusively.

We will get JDBC software from three parties.

1)Sun micro system company (creator of jdbc driver)
2)Database vendor
3)Third party vendor

We will get ODBC software from three parties.

1)Xopen company (creator of ODBC driver)
2)Database vendor
3)Third party vendor

Q)How many drivers are there in jdbc?

We have four types of JDBC drivers.
1)Type1 JDBC driver (JDBC-ODBC bridge driver /partly java driver)
2)Type2 JDBC driver (Native API / partly java driver)
3)Type3 JDBC driver (Net Protocol / Java Driver)
4)Type4 JDBC driver (Native Protocol / Java Driver)

To use any JDBC driver we need to register JDBC driver with DriverManager service.
Every jdbc application contains one built-in service called DriverManager service.
This service we can achieve by using DriverManager class

Class.forName()
==========
It is always recommanded to use Class.forName() method to register JDBC driver with DriverManager service.
It is used to load the driver class but it won't create an object.

ex:

```
Class.forName("driver-class-name");
```

Connection object
==========
Connection is an interface which is present java.sql package.
It is an object of underlying supplied java class which implements java.sql.Connection interface.
To perform any operation in a database we required Connection object.
Once the work with database is completed we need to close the Connection object.

DriverManager.getConnection()
==========
It is a static method present in DriverManager class.
It is used to connect with database and return one JDBC Connection object representing connectivity between java application and database software.

ex:

```
Connection con=DriverManager.getConnection("driver-url","uname","pwd");
```

Statement object
==========
Statement is an interface which is present in java.sql package.
It is an object of underlying supplied java class which implements java.sql.Statement interface.
It acts like a vehicle between java application and database software.
It is used to sends and executes SQL query in database software.
Statement object we can create by using createStatement() method of Connection object.
ex:

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

CLASS-2

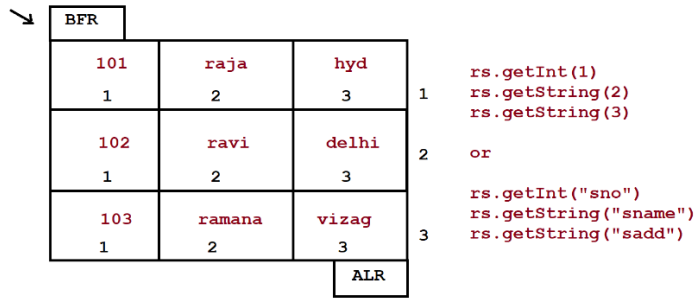
ResultSet object
==========
ResultSet is an interface which is present in java.sql package.
Every ResultSet contains two positions.

1)BFR (Before First Record/Row)
2)ALR (After Last Record/Row)

Bydefault every record pointer points to BFR position.
Every record ResultSet having 1 as base index and 1 as column index.

rs.next()

It is used to move from current position to next position
If next position is a record then it will return true.
If next position is a ALR then it will return false.
We can read record ResultSet by using getXxx() method by using column names or index numbers.



Steps to develop JDBC application

==========

There are six steps to develop JDBC application.

step1:

Register JDBC Driver with DriverManager service.

step2:

Establish the connection with database software.

step3:

Create Statement object.

step4:

Sends and Executes SQL Query in database software.

step5:

Gather the result from database software to process the result.

step6:

Close all jdbc connection objects.

Types of Queries in JDBC

==========

According to JDBC point of view, we have two types of queries.

1)Select query
2)Non-Select query

1)Select query

Select query will give bunch of records from database.

ex:

```
select * from emp;
```

To execute select query we need to use executeQuery() method of Statement object.

ex:

```
ResultSet rs=st.executeQuery("select * from emp");
```

2)Non-Select query

It will return numeric value representing number of records effected in database table.

ex:

```
delete from emp;
```

To execute non-select query we need to use executeUpdate() method of Statement object.

ex:

```
int result=st.executeUpdate("delete from emp");
```

Type4 JDBC Driver

==========

```
                                          class name
                                        ------------
driver name      : oracle.jdbc.driver.OracleDriver
                   -------------------
                         pkg name

                                  protocol
                  ----------------------------------
driver url       : jdbc:oracle:thin:@localhost:1521:XE
                   -----------------  |       |   |
                         sub protocol  hostname portno logical database name
```

database username : system

database password : admin

Eclipse
==========
Environment : JAVA JEE IDE (Integrated Development Environment)
Flavour : Kepler, Indigo , Luna , Mars and etc.
Vendor : Eclipse organization
website : www.eclipse.org
Download link :

https://www.eclipse.org/downloads/packages/release/kepler/sr2/eclipse-ide-java-ee-developers

Extract Eclipse IDE
==========
Right click to Eclipse Zip file --> extract files -->
select any drive from computer --> ok.

Steps to develop first jdbc application using eclipse IDE to select the records from student table
==========
step1:

Make sure Eclipse IDE is installed in our computer.
step2:

Launch eclipse IDE by choosing workspace location.
step3:

Create a "IH-JAVA-014" project inside eclipse IDE.
ex:
File --> new --> Project --> Java project --> next -->
Project Name : IH-JAVA-014 --> next --> libraries --> add external jars
--> select "ojdbc14.jar" file --> open -->Finish.
step4:
---
create "com.ihub.www" package inside "src" folder.
ex:
Rightclick to src folder --> new --> package --->
package name : com.ihub.www --> finish.
step5:
---
create "SelectApp.java" file inside "com.ihub.www" package.
ex:
Right click to com.ihub.www package --> new --> class -->
class name : SelectApp --> Finish.

SelectApp.java

```
package com.ihub.www;

// ctrl + shift + o
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
```

```
public class SelectApp
{
	public static void main(String[] args)throws Exception
	{
		Class.forName("oracle.jdbc.driver.OracleDriver");
		Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");
		Statement st=con.createStatement();
		ResultSet rs=st.executeQuery("select * from student");
		while(rs.next())
		{
			System.out.println(rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3));
		}
		rs.close();
		st.close();
		con.close();
	}
}
```

step6:

Run the jdbc application.

ex:

Right click to SelectApp.java file --> run as --> Java application.

approach2

```
package com.ihub.www;

// ctrl + shift + o
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class SelectApp
{
	public static void main(String[] args)throws Exception
	{
		Class.forName("oracle.jdbc.driver.OracleDriver");
		Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");
		Statement st=con.createStatement();
		ResultSet rs=st.executeQuery("select * from student order by sno desc");
		while(rs.next())
		{
			System.out.println(rs.getInt("sno")+" "+rs.getString("sname")+"
"+rs.getString("sadd"));
		}
		rs.close();
		st.close();
		con.close();
	}
}
```

ex:3

```
package com.ihub.www;

// ctrl + shift + o
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class SelectApp
{
	public static void main(String[] args)
	{
		Connection con=null;
		Statement st=null;
		ResultSet rs=null;
		String qry=null;
		try
		{
			Class.forName("oracle.jdbc.driver.OracleDriver");

	con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin"
);
			st=con.createStatement();
			qry="select * from student";
			rs=st.executeQuery(qry);
			while(rs.next())
			{
				System.out.println(rs.getInt("sno")+" "+rs.getString("sname")+"
"+rs.getString("sadd"));
			}
			rs.close();
			st.close();
			con.close();
		}
		catch(Exception e )
		{
			e.printStackTrace();
		}

	}
}
```

CLASS-3

select query

==========

Q)Write a jdbc application to select student record based on student number?

```
package com.ihub.www;

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

```
import java.sql.Statement;
import java.util.Scanner;

public class SelectApp2
{
	public static void main(String[] args)throws Exception
	{
		Scanner sc=new Scanner(System.in);
		System.out.println("Enter the Student Number :");
		int no=sc.nextInt();

		Class.forName("oracle.jdbc.driver.OracleDriver");
		Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");
		Statement st=con.createStatement();

		String qry="select * from student where sno="+no;

		ResultSet rs=st.executeQuery(qry);

		while(rs.next())
		{
			System.out.println(rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3));
		}

		rs.close();
		st.close();
		con.close();

	}
}
```

ex:2

```
package com.ihub.www;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.Scanner;

public class SelectApp2
{
	public static void main(String[] args)throws Exception
	{
		Scanner sc=new Scanner(System.in);
		System.out.println("Enter the Student Number :");
		int no=sc.nextInt();

		Class.forName("oracle.jdbc.driver.OracleDriver");
		Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");
		Statement st=con.createStatement();
```

```
		String qry="select * from student where sno="+no;

		ResultSet rs=st.executeQuery(qry);

		int cnt=0;

		while(rs.next())
		{
			System.out.println(rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3));
			cnt=1;
		}

		if(cnt==0)
			System.out.println("No rows selected");

		rs.close();
		st.close();
		con.close();

	}
}
```

Non-Select Query

==========

Q)Write a jdbc application to insert a record into student table?

```
package com.ihub.www;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
import java.util.Scanner;

public class InsertApp
{
	public static void main(String[] args)throws Exception
	{
		Scanner sc=new Scanner(System.in);
		System.out.println("Enter the Student No :");
		int no=sc.nextInt();

		System.out.println("Enter the Student Name :");
		String name=sc.next();

		System.out.println("Enter the Student Address :");
		String add=sc.next();

		//converting string inputs according to sql query
		name="'"+name+"'";
		add="'"+add+"'";

		Class.forName("oracle.jdbc.driver.OracleDriver");
		Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");
```

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

		String qry="insert into student values("+no+","+name+","+add+")";

		int result=st.executeUpdate(qry);

		if(result==0)
			System.out.println("No Record Inserted");
		else
			System.out.println(result+" Record Inserted");

		st.close();
		con.close();
	}
}
```

Q)Write a jdbc application to update student name based on student no?

ex:

```
package com.ihub.www;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
import java.util.Scanner;

public class UpdateApp
{
	public static void main(String[] args)throws Exception
	{
		Scanner sc=new Scanner(System.in);
		System.out.println("Enter the Student No :");
		int no=sc.nextInt();

		System.out.println("Enter the Student Name :");
		String name=sc.next();

		//converting string inputs according to sql query
		name="'"+name+"'";

		Class.forName("oracle.jdbc.driver.OracleDriver");
		Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");
		Statement st=con.createStatement();

		String qry="update student set sname="+name+" where sno="+no;

		int result=st.executeUpdate(qry);

		if(result==0)
			System.out.println("No Record Updated");
		else
			System.out.println(result +" Record updated");
```

```
                st.close();
                con.close();

        }
}
```

Q)Write a jdbc application to delete student record based on student no?

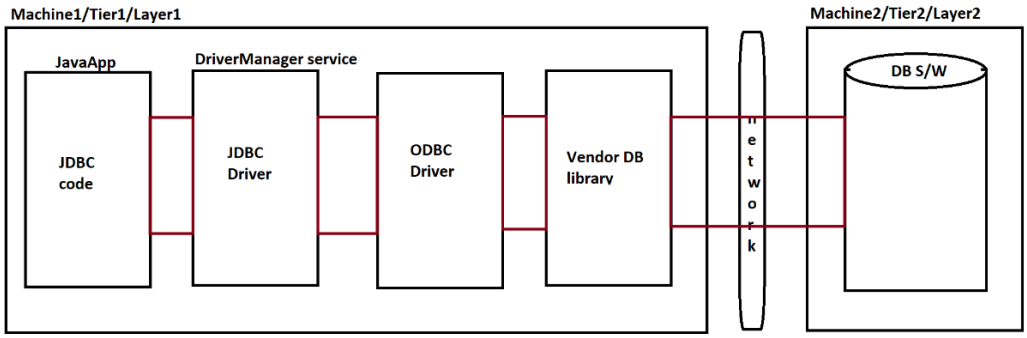
```
package com.ihub.www;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
import java.util.Scanner;

public class DeleteApp
{
        public static void main(String[] args)throws Exception
        {
                Scanner sc=new Scanner(System.in);
                System.out.println("Enter the Student No :");
                int no=sc.nextInt();

                Class.forName("oracle.jdbc.driver.OracleDriver");
                Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");
                Statement st=con.createStatement();

                String qry="delete from student where sno="+no;

                int result=st.executeUpdate(qry);

                if(result==0)
                        System.out.println("No Record Deleted");
                else
                        System.out.println(result+ " Record Deleted");

                st.close();
                con.close();

        }
}
```

Type1 JDBC Driver architecture (JDBC-ODBC bridge driver)

==========

Type1 JDBC driver is not designed to interact with database software directly.

It is designed to interact with ODBC driver and Vendor DB library to locate and interact with database software.



Advantages:

It is a built-in driver of JDK.
Using Type1 jdbc driver we can interact with any database software.

Disadvantages:

This driver performance is low. It is not suitable for medium and large scale projects.Hence it is not a industry standard driver.

Since ODBC driver and Vendor db library present at client so it is not suitable for untrusted applets to database communication.

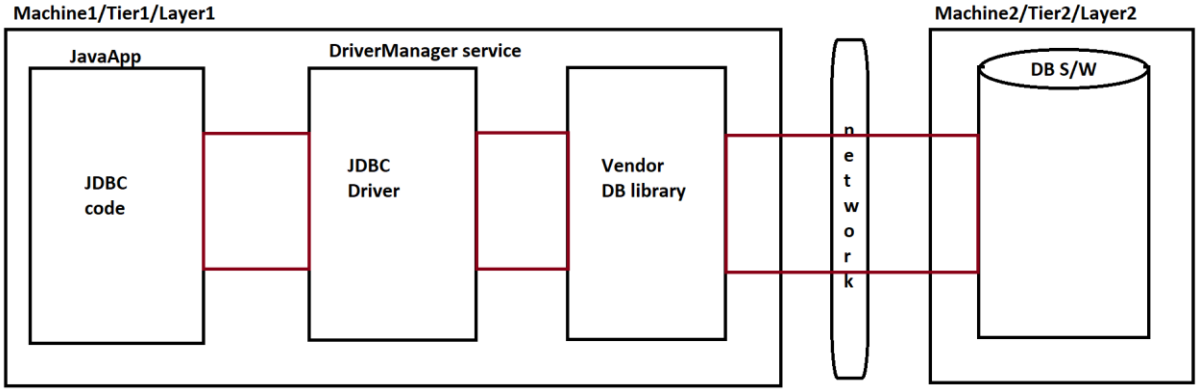
To work with type1 jdbc driver we need to arrange ODBC driver and Vendor db library.

Type2 JDBC driver architecture (Native API)

==========

Type2 JDBC driver is not designed to interact with database software directly.

It is designed to take the support of Vendor DB library to locate and interact with database software.



Advantages:

Type2 JDBC driver will give better performance when compare to Type1 JDBC driver.

Type2 JDBC driver will not take the support of ODBC driver.

Disadvantages:

This driver performance is quite slow. It is not suitable for medium and large scale projects.Hence it is not a industry standard driver.

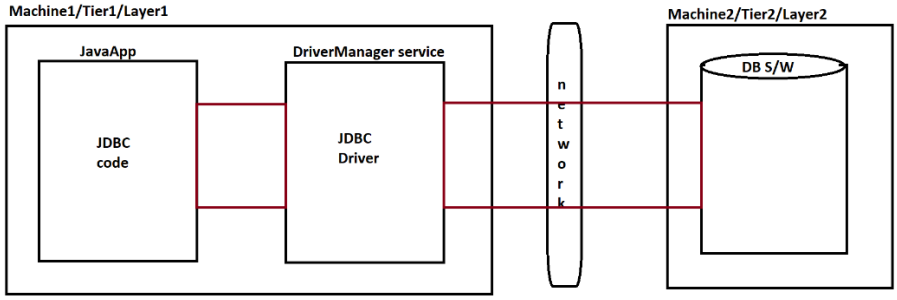
Since vendor db library present at client side so it is not suitable for untrusted applets to database communication.

For every database we need to arrange type2 jdbc driver seperately.

Type4 JDBC driver architecture (Native protocol)

==========

Type4 JDBC driver is not designed to interact with ODBC driver and Vendor db library.
It is designed to locate and interact with database software directly.



Type4 JDBC driver is also known as thin driver.

Advantages:

It will give better performance when compare to Type1 and Type2 jdbc driver.
It is suitable for medium and large scale projects.Hence it is a industry standard driver.
It is developed in java so it will give platform independency.
Since ODBC drivers and Vendor db libraries not present at client side so it is suitable for untrusted applets to database communication.

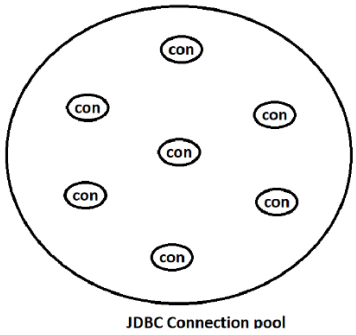
Disadvantages:

It is not a built-in driver of JDK.
For every database we need to arrange type4 jdbc driver seperately.

JDBC Connection pooling

==========

It is a factory containing set of readily available JDBC connection objects before actual being used.



JDBC connection pool represent connectivity with same database software.

Advantages:

It will create reusable JDBC Connection objects.
With minimum Connection objects we can interact with multiple clients.

A user is not responsible to create , manage and destroy JDBC Connection objects in JDBC Connection pool. JDBC Connection pool is responsible to create ,manage and destroy JDBC Connection object.

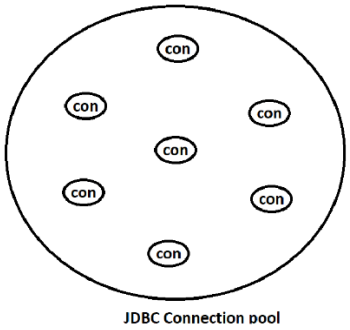
Type3 JDBC Driver Architecture (Net Protocol)

==========

Webserver or proxy server or IDE's server containing JDBC Connection pool represent JDBC Connection objects.

Type3 JDBC driver is not designed to interact with database software directly.

It is designed to interact with web server or proxy server to get one resuable JDBC Connection object from JDBC Connection pool.



With respect to the diagram

---

1)First webserver or proxy server intract with database software to get some reusable jdbc Connection objects in JDBC Connection pool.

2)Our application interacts with web server or proxyserver to get one reusable jdbc Connection object from jdbc Connection pool.

3)Our application uses reusable jdbc Connection object, to create other Connection objects.

4)Once if we close i.e con.close() then Connection object goes back to JDBC Connection pool.

Q)Types of Connection objects?

We have two types of Connection objects.

1)Direct Connection object

A connection object which is created by the user is called direct direct connection object.

ex:

```
        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");
```

2)Pooled JDBC Connection object

A connection object which is gathered from JDBC Connection pool is called pooled jdbc connection object.

CLASS-4

Q)Types of Statement objects in JDBC?

We have three statement objects in JDBC.
1)Simple Statement object
2)PreparedStatement object
3)CallableStatement object

SQL Injection problem
==========
Along with input values if we pass special SQL instructions which change the behaviour of a query and bahaviour of an application is called SQL Injection problem.
Here special SQL instruction means comment in SQL (--).
while dealing with simple Statement object there is a chance of raising SQL injection problem.

ex:
username : raja'--
password : hyd

Valid Credentials

userlist table
==========
```
drop table userlist;
create table userlist(uname varchar2(10),pwd varchar2(10));
insert into userlist values('raja','rani');
insert into userlist values('king','kingdom');
commit;
```

ex:

```
package com.ihub.www;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.Scanner;

public class SQLInjectionProbApp
{
	public static void main(String[] args)throws Exception
	{
		Scanner sc=new Scanner(System.in);
		System.out.println("Enter the Username :");
		String name=sc.next();

		System.out.println("Enter the Password :");
		String pass=sc.next();

		//convert inputs according to SQL query
		name="'"+name+"'";
		pass="'"+pass+"'";
```

```
                Class.forName("oracle.jdbc.driver.OracleDriver");
                Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");
                Statement st=con.createStatement();

                String qry="select count(*) from userlist where uname="+name+" and pwd="+pass;

                ResultSet rs=st.executeQuery(qry);
                int result=0;
                while(rs.next())
                {
                        result=rs.getInt(1);
                }
                if(result==0)
                        System.out.println("Invalid Credentails");
                else
                        System.out.println("Valid Credentials");

                rs.close();
                st.close();
                con.close();
        }
}
```

Limitations with Simple Statement object

==========

It is not suitable to execute same query for multiple times with same or different values.
It raises SQL injection problem.
Framing query with variables is quit complex.
We can't use string values directly in a query without conversion.
It does not allow us to insert date values in a database table column.
It does not allow us to insert LOB(Large Object) values in a database table column.
To overcome these limitations we need to use PreparedStatement object.

Pre-compiled SQL Query

==========

Our query goes to database software without input values and becomes parsed query
either it is executed or not is called pre-compiled SQL query.
PreparedStatement object deals with pre-compiled SQL query.

Working with PreparedStatement object

==========

step1:
        create a query with placeholders or parameters.
        ex:
        String qry="insert into student values(?,?,?)";
step2:
        convert SQL query to precompiled SQL query.
        ex:
        PreparedStatement ps=con.prepareStatement(qry);
step3:
        set the values to query parameters.
        ex:
                ps.setInt(1,no);
                ps.setString(2,name);

```
		ps.setString(3,add);
```
step4:

Execute precompiled SQL query.

ex:

```
		ps.executeUpdate();
```
step5:

Close PreparedStatement object.

ex:

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

Q)Write a jdbc application to insert a record into student table by using PreparedStatement object?

ex:

```
package com.ihub.www;

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

public class PSInsertApp
{
	public static void main(String[] args)throws Exception
	{
		Scanner sc=new Scanner(System.in);
		System.out.println("Enter the student no :");
		int no=sc.nextInt();

		System.out.println("Enter the student name :");
		String name=sc.next();

		System.out.println("Enter the student address :");
		String add=sc.next();


		Class.forName("oracle.jdbc.driver.OracleDriver");
		Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");

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

		PreparedStatement ps=con.prepareStatement(qry);

		//set the values
		ps.setInt(1,no);
		ps.setString(2,name);
		ps.setString(3,add);

		//execute
		int result=ps.executeUpdate();
		if(result==0)
			System.out.println("No Record Inserted");
		else
```

```
                        System.out.println("Record Inserted");

                ps.close();
                con.close();

        }
}
```

Q)Write a jdbc application to update student name based on student number?

```
package com.ihub.www;

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

public class PSUpdateApp
{
        public static void main(String[] args)throws Exception
        {
                Scanner sc=new Scanner(System.in);
                System.out.println("Enter the student no :");
                int no=sc.nextInt();

                System.out.println("Enter the student name :");
                String name=sc.next();

                Class.forName("oracle.jdbc.driver.OracleDriver");
                Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");

                String qry="update student set sname=? where sno=?";

                PreparedStatement ps=con.prepareStatement(qry);

                //set the values
                ps.setString(1,name);
                ps.setInt(2,no);

                //execute
                int result =ps.executeUpdate();
                if(result==0)
                        System.out.println("No Record Updated");
                else
                        System.out.println("Record updated");
                ps.close();
                con.close();
        }
}
```

Q)Write a jdbc application to delete student record based on student no?

```
package com.ihub.www;
```

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.util.Scanner;

public class PSDeleteApp
{
	public static void main(String[] args)throws Exception
	{
		Scanner sc=new Scanner(System.in);
		System.out.println("Enter the student no :");
		int no=sc.nextInt();

		Class.forName("oracle.jdbc.driver.OracleDriver");
		Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");

		String qry="delete from student where sno=?";

		PreparedStatement ps=con.prepareStatement(qry);

		//set the value
		ps.setInt(1,no);

		//execute
		int result=ps.executeUpdate();
		if(result==0)
			System.out.println("No Record Deleted");
		else
			System.out.println("Record Deleted");

		ps.close();
		con.close();
	}
}
```

Solution for SQL Injection problem

==========

```
package com.ihub.www;

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

public class SolutionForSQLInjectionPrb
{
	public static void main(String[] args)throws Exception
	{
		Scanner sc=new Scanner(System.in);

		System.out.println("Enter the username : ");
		String name=sc.next();
```

```
                System.out.println("Enter the password : ");
                String pass=sc.next();


                Class.forName("oracle.jdbc.driver.OracleDriver");
                Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");

                String qry="select count(*) from userlist where uname=? and pwd=?";

                PreparedStatement ps=con.prepareStatement(qry);

                //set the values
                ps.setString(1,name);
                ps.setString(2,pass);

                //execute
                ResultSet rs=ps.executeQuery();
                int result=0;
                while(rs.next())
                {
                        result=rs.getInt(1);
                }
                if(result==0)
                        System.out.println("Invalid Credentials ");
                else
                        System.out.println("Valid Credentails ");

                ps.close();
                con.close();
        }
}
```

DatabaseMetaData

==========

DatabaseMetaData is an interfac which is present in java.sql package.
DatabaseMetaData provides metadata of a database.
Data of a data is called metadata.
DatabaseMetaData gives information about database product name, database product version, database driver name, database driver version, username and etc.
We can create DatabaseMetaData object by using getMetaData() method of Connection object.

ex:

```
        DatabaseMetaData dbmd=con.getMetaData();
```

ex:

```
package com.ihub.www;

import java.sql.Connection;
import java.sql.DatabaseMetaData;
import java.sql.DriverManager;

public class DBMDApp
```

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

		Class.forName("oracle.jdbc.driver.OracleDriver");
		Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");

		DatabaseMetaData dbmd=con.getMetaData();

		System.out.println(dbmd.getDatabaseProductName());
		System.out.println(dbmd.getDatabaseProductVersion());
		System.out.println(dbmd.getDriverName());
		System.out.println(dbmd.getDriverVersion());
		System.out.println(dbmd.getUserName());

		con.close();
	}
}
```

CLASS-5

Working with Date values

==========

While dealing with DOB,DOA,DOD,DOR and etc we need to insert or retrieve date values.
It is never recommanded to insert date values in the form of String because it will not give proper comparision between two dates.

Every database software support different date pattern.
ex:

oracle --> dd-MMM-yy
mysql --> yyyy-MM-dd

We can't place Date values directly to query parameter by using Simple Statement object.

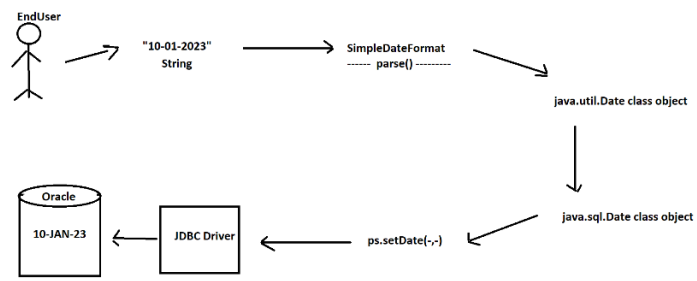
To overcome this limitation we need to use PreparedStatement object.
java.util.Date class is not suitable to perform date operations.
java.sql.Date class is suitable to perform database operations.

To we can set date values to query parameter by using setDate(-,-) method of PreparedStatement object.

Once the date value goes to jdbc driver then jdbc driver will insert date value in the pattern which is supported by underlying database software.

Standard procedure to insert date value

==========



1)First enduser will give date value in the form of String.

2)A java.text.SimpleDateFormat class contains parse() method to convert
String date to java.util.Date class object.

3)Our application converts java.util.Date class object to java.sql.Date class object.

4)By using ps.setDate(-,-) method, we can set the date value directly to query parameter.

5)Once the jdbc driver will get date value then it will insert in pattern
which is supported underlying database software.

emp1 table

==========

```
drop table emp1;
create table emp1(eid number(3), ename varchar2(10), edoj date);
```

ex:

```
package com.ihub.www;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.text.SimpleDateFormat;
import java.util.Scanner;

public class DateInsertApp
{
	public static void main(String[] args)throws Exception
	{
		Scanner sc=new Scanner(System.in);
		System.out.println("Enter the employee id :");
		int id=sc.nextInt();

		System.out.println("Enter the employeen name :");
		String name=sc.next();

		System.out.println("Enter the employee DOJ(dd-MM-yyyy) :");
		String sdoj=sc.next();

		//converting string date to util date
		SimpleDateFormat sdf=new SimpleDateFormat("dd-MM-yyyy");
		java.util.Date udoj=sdf.parse(sdoj);

		//converting util date to sql date
		long ms=udoj.getTime();
		java.sql.Date sqldoj=new java.sql.Date(ms);

		Class.forName("oracle.jdbc.driver.OracleDriver");
		Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");

		String qry="insert into emp1 values(?,?,?)";
		PreparedStatement ps=con.prepareStatement(qry);
```

```
		//set the values
		ps.setInt(1,id);
		ps.setString(2,name);
		ps.setDate(3,sqldoj);

		//execute
		int result=ps.executeUpdate();
		if(result==0)
			System.out.println("No record inserted");
		else
			System.out.println("Record inserted");

		ps.close();
		con.close();
	}
}
```

Date Retrieve application

==========

ex:

```
package com.ihub.www;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.text.SimpleDateFormat;

public class DateRetrieveApp {

	public static void main(String[] args)throws Exception
	{
		Class.forName("oracle.jdbc.driver.OracleDriver");
		Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");
		Statement st=con.createStatement();
		String qry="select * from emp1";
		ResultSet rs=st.executeQuery(qry);
		while(rs.next())
		{
			int id=rs.getInt(1);
			String name=rs.getString(2);
			java.sql.Date sqldoj=rs.getDate(3);

			//converting sql date to util date
			java.util.Date udoj=(java.util.Date)sqldoj;

			//converting util date to string date
			SimpleDateFormat sdf=new SimpleDateFormat("dd-MM-yyyy");
			String sdoj=sdf.format(udoj);

			System.out.println(id+" "+name+" "+sdoj);
		}
		rs.close();
```

```
                st.close();
                con.close();
        }

}
```

Working with LOB(Large Objects) values

==========

Files are known as LOB's.

We have two types of LOB's.

1)BLOB (Binary Large object)

---

ex:

images, audio,video, avi file and etc.

2)CLOB (Character large object)

---

ex:

text file ,adv text file , doc file and etc.

While dealing with matrimonial application, job portal application and profile management application we need to insert and retrieve lob values.

To deal with lob values we need to take the support of PreparedStement object.
We can set lob values directly to query parameter by using following methods.
ex:

setBLOB(-,-,-)/setBinaryStream(-,-,-)
setCLOB(-,-,-)/setCharacterStream(-,-,-)

emp2 table

```
drop table emp2;
create table emp2(eid number(3),ename varchar2(10),ephoto BLOB);
```

ex:

```
package com.ihub.www;

import java.io.File;
import java.io.FileInputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.util.Scanner;

public class PhotoInsertApp
{
        public static void main(String[] args)throws Exception
        {
                Scanner sc=new Scanner(System.in);
                System.out.println("Enter the employee id :");
                int id=sc.nextInt();
                System.out.println("Enter the employee name :");
```

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

		//locate a file
		File f=new File("src/com/ihub/www/girl3.jpg");
		FileInputStream fis=new FileInputStream(f);

		Class.forName("oracle.jdbc.driver.OracleDriver");
		Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");
		String qry="insert into emp2 values(?,?,?)";
		PreparedStatement ps=con.prepareStatement(qry);
		//set the values
		ps.setInt(1,id);
		ps.setString(2,name);
		ps.setBinaryStream(3, fis,(int)f.length());

		//execute
		int result=ps.executeUpdate();
		if(result==0)
			System.out.println("No record inserted");
		else
			System.out.println("Record inserted");
		ps.close();
		con.close();
	}
}
```

Photo Retrieve Application

```
package com.ihub.www;

import java.io.FileOutputStream;
import java.io.InputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class PhotoRetrieveApp
{
	public static void main(String[] args)throws Exception
	{

		Class.forName("oracle.jdbc.driver.OracleDriver");
		Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");
		Statement st=con.createStatement();
		String qry="select * from emp2";
		ResultSet rs=st.executeQuery(qry);
		while(rs.next())
		{
			InputStream is=rs.getBinaryStream(3);

			int byteReads=0;
			byte[] buff=new byte[400];
```

```
                FileOutputStream fos=new FileOutputStream("E:\\DEMO\\pic.jpg");

                        while((byteReads=is.read(buff))!=-1)
                        {
                                fos.write(buff,0,byteReads);
                        }
                        fos.close();
                }
                System.out.println("Please check the location");
                rs.close();
                st.close();
                con.close();
        }
}
```

ResultSetMetaData

==========

ResultSetMetaData is an interface which is present in java.sql package.
ResultSetMetaData provides metadata of a table.
It gives information about number of columns, types of columns, name of columns , size of columns and etc.
We can create ResultSetMetaData object by using getMetaData() method of ResultSet object.
ex:

```
        ResultSetMetaData rsmd=rs.getMetaData();
```

ex:

```
package com.ihub.www;

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

public class RSMDApp
{
        public static void main(String[] args)throws Exception
        {
                Class.forName("oracle.jdbc.driver.OracleDriver");
                Connection
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");
                Statement st=con.createStatement();

                String qry="select * from student";

                ResultSet rs=st.executeQuery(qry);

                ResultSetMetaData rsmd=rs.getMetaData();
                System.out.println(rsmd.getColumnCount());
                System.out.println(rsmd.getColumnTypeName(1));
                System.out.println(rsmd.getColumnName(1));
                System.out.println(rsmd.getColumnDisplaySize(1));
```

```
                rs.close();
                st.close();
                con.close();
        }
}
```

JDBC Flexible Application

==========

In JDBC , Connection object is a heavy weight object.
It is never recommanded to create Connection object in a every jdbc application.
We need to create a seperate class which returns JDBC Connection object.

ex:

DBConnection.java

```
package com.ihub.www;

import java.sql.Connection;
import java.sql.DriverManager;

public class DBConnection
{
        private static Connection con=null;

        private DBConnection()
        {

        }

        public  static Connection getConnection()
        {
                try
                {
                        Class.forName("oracle.jdbc.driver.OracleDriver");

        con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin"
);
                }
                catch(Exception e)
                {
                        e.printStackTrace();
                }
                return con;
        }
}
```

FlexibleApp.java

```
package com.ihub.www;

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

```
public class FlexibleApp
{
	public static void main(String[] args)throws Exception
	{
		Connection con=DBConnection.getConnection();
		Statement st=con.createStatement();
		String qry="select * from student";
		ResultSet rs=st.executeQuery(qry);
		while(rs.next())
		{
			System.out.println(rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3));
		}
		rs.close();
		st.close();
		con.close();
	}
}
```

Working with Properties file

==========

In regular intervals, Our DBA will change username and password for security reason.
It is never recommanded to pass database properties directly to jdbc application.
It is always recommanded to read database properties from properties file.
A properties file contains key and value pair.

dbdetails.properties

```
driver=oracle.jdbc.driver.OracleDriver
url=jdbc:oracle:thin:@localhost:1521:XE
username=system
password=admin
```

PropertiesFileApp.java

```
package com.ihub.www;

import java.io.FileInputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.Properties;

public class PropertiesFileApp {

	public static void main(String[] args)throws Exception
	{
		//locate  a properties file
		FileInputStream fis=new FileInputStream("src/com/ihub/www/dbdetails.properties");

		//declare properties class object
		Properties p=new Properties();

		//load the data from properties file to properties class
		p.load(fis);
```

```
		//read the data from properties class
		String s1=p.getProperty("driver");
		String s2=p.getProperty("url");
		String s3=p.getProperty("username");
		String s4=p.getProperty("password");

		Class.forName(s1);
		Connection con=DriverManager.getConnection(s2,s3,s4);
		Statement st=con.createStatement();
		String qry="select * from student";
		ResultSet rs=st.executeQuery(qry);
		while(rs.next())
		{
			System.out.println(rs.getRow()+" "+rs.getInt(1)+" "+rs.getString(2)+"
"+rs.getString(3));
		}
		rs.close();
		st.close();
		con.close();
	}

}
```