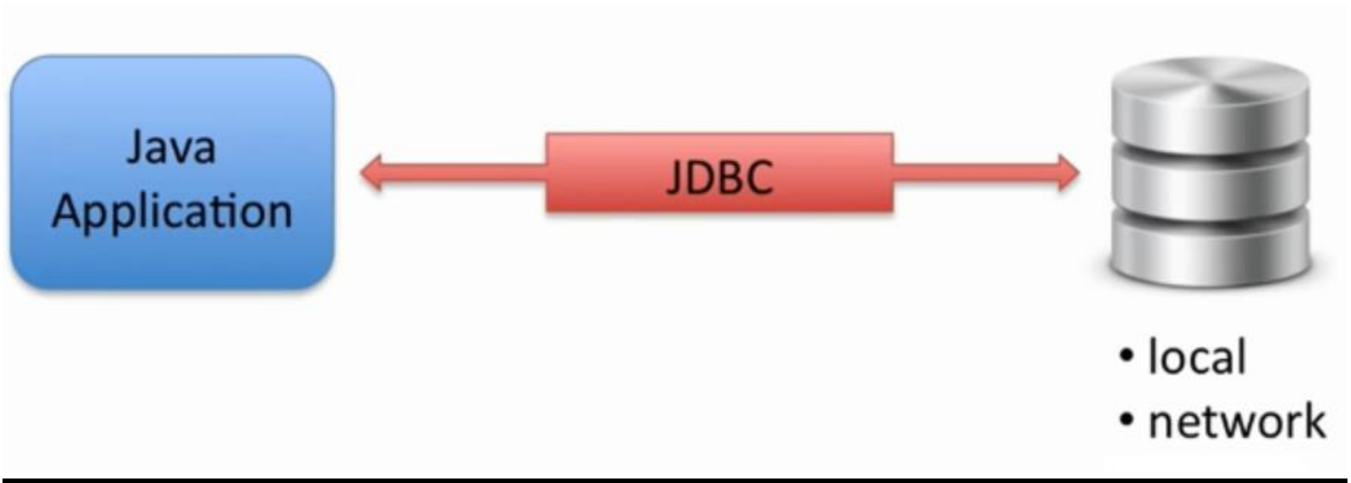
Lab 10: JDBC

Objectives:

At the end of this lab, you should be able to
- Use JDBC to read, write data to Oracle database

Introduction:

Java Database Connectivity (JDBC) is an application programming interface (API) which allows the programmer to connect and interact with databases, the database might be saved locally or on network. It provides methods to query and update data in the database through update statements like SQL's CREATE, UPDATE, DELETE and INSERT and query statements such as SELECT. Additionally, JDBC can run stored procedures.



- It supports a large number of databases like Oracle, MySql, SQLServer, Sysbase, DB2,..etc.
- To connect to a specific database type, you need to use JDBC driver that is responsible of converting API calls to low level calls for that specific database, the driver is usually is normally provided by database vendor.

Eclipse download: https://www.eclipse.org/downloads/packages/release/kepler/sr1/eclipse-ide-java-developers

JDBC API

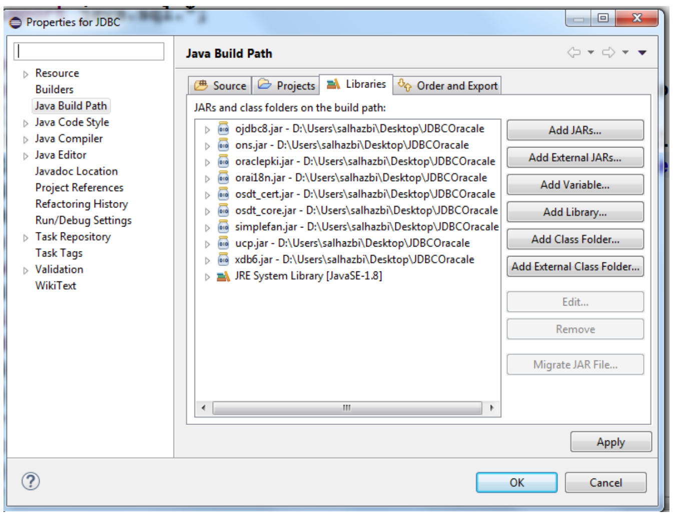
The main Java classes that implement JDBC:
java.sql.DriverManager //Driver Manager **provides a basic service for managing a set of JDBC drivers**
java.sql.Connection
java.sql.Statement
java.sql.ResultSet

They are defined in the packages: java.sql and javax.sql.

Installing and using Oracle JDBC driver in Eclipse

We need to use Oracle JDBC driver, it can be downloaded from
http://www.oracle.com/technetwork/database/application-development/jdbc/downloads/index.html
(For our lab, it is already on Blackboard)

To add them to the project (your project in Eclipse), go to “project” menu, select “properties”, then select “Java Build Path”, select the tap “Libraries”, then press “Add External JARs”, select the files you download.



JARs and class folders on the build path:

```
ojdbc8.jar - D:\Users\salhazbi\Desktop\JDBCOracale
ons.jar - D:\Users\salhazbi\Desktop\JDBCOracale
oraclepki.jar - D:\Users\salhazbi\Desktop\JDBCOracale
orai18n.jar - D:\Users\salhazbi\Desktop\JDBCOracale
osdt_cert.jar - D:\Users\salhazbi\Desktop\JDBCOracale
osdt_core.jar - D:\Users\salhazbi\Desktop\JDBCOracale
simplefan.jar - D:\Users\salhazbi\Desktop\JDBCOracale
ucp.jar - D:\Users\salhazbi\Desktop\JDBCOracale
xdb6.jar - D:\Users\salhazbi\Desktop\JDBCOracale
JRE System Library [JavaSE-1.8]
```

Writing JDBC program:

1- Create a connection to database
2- Create a statement object
3- Execute SQL query.
4- Process Result set.

<u>Step 1: Create a connection to database:</u>

In order to connect to a database, you need URL string to indicate location of the database, this might be locally on the same device, or maybe on network.
The syntax is

```
jdbc:<driver protocol>:<driver connection details>
```

for example with Oracle database

$$\texttt{jdbc:oracle:thin@}\underbrace{\texttt{myserver:}}_{\text{Server name}}\underbrace{\text{1521}}_{\text{Oracle Listener port}}/\underbrace{\texttt{demodb}}_{\text{Service name}}$$

To create a connection, we also need user name and password :

```
Connection conn =
DriverManager.getConnection("jdbc:oracle:thin:@coestudb.qu.edu.qa:1521/STUD.qu.edu.
qa","user", "pw");
```

<u>Step 2: Create a statement object:</u>

```
Statement stmt = conn.createStatement();
```

<u>Step 3: Execute SQL query.</u>

```
ResultSet rs = stmt.executeQuery("select ename, empno, sal from emp");
```

Step 4: Process Result set.

```
     while (rs.next()) {
               String name = rs.getString(1);
               int number = rs.getInt(2);
               double salary = rs.getDouble(3);
               System.out.println(name + " " + number + " " + salary);
          }
```

- Method *ResultSet.next()* moves forward for one row, it returns true if there are more rows to process.
- We will use this method in a loop to process all retrieved data.
- To get specific field data from the retrieved data, we use *getXXX* methods, we can pass the name of the field, or a number corresponding the orders of the fields in SQL statement.
- We need to close the resultset and connection

```
    rs.close();
    conn.close();
```

```
import java.sql.*;
public class Test {
     public static void main(String args[]) throws SQLException {

     Connection conn = DriverManager.getConnection("jdbc:oracle:thin:@
                    coestudb.qu.edu.qa:1521/STUD.qu.edu.qa","user", "pw");
          Statement stmt = conn.createStatement();
          ResultSet rs = stmt.executeQuery("select ename, empno, sal from emp");
          while (rs.next()) {
               String name = rs.getString(1);
               int number = rs.getInt(2);
               double salary = rs.getDouble(3);
               System.out.println(name + " " + number + " " + salary);
          }
          rs.close();
          conn.close();
     }
}
```

DML through JDBC

To insert, update, or delete records in database, instead of *executeQueury*, we use *executeUpdate*

Insert:

```
String sql=" insert into emp (empno,ename,sal,deptno)"
          +"values (333,'Khaled',4000,10)";
          stmt.executeUpdate(sql);
```

Update:

```
String sql=" update emp set sal=9000"+
                " where empno=333";
          int affectedRows =stmt.executeUpdate(sql);
          System.out.println("Number of updated records "+affectedRows);
```

Delete:

```
String sql=" delete from emp where empno=333";
          int affectedRows=stmt.executeUpdate(sql);
```

```
            System.out.println("Number of updated records "+affectedRows);
```

Prepared Statement

It is a precompiled SQL statement used to execute parameterized query. For example instead of hardcoding the values in the SQL statement:

```
select * from emp where sal>2000 and job='MANAGER'
```

You can use placeholders where you can use question mark as placeholder symbol

```
select * from emp where sal>?  and job=?
```

Create the prepared statement

```
PreparedStatement stmt=conn.prepareStatement("select empno, ename,sal,hiredate from
emp where sal>? and job=? ");

Stmt.setInt(1,V1);
stmt.setString(2,V2);
```

Exercise 1:

Using the prepared statement above, write a program to display empno, ename, sal, hiredate of employees whose salaries are greater than **"Entered by User"** and their jobs are **"Entered by User".**

Call Stored Procedure

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

public class CallProcedure {
	public static void main(String args[]) {

		Connection conn;
		try {
			conn =
DriverManager.getConnection("jdbc:oracle:thin:@coestudb.qu.edu.qa:1521/STUD.qu.edu.qa",
					"username", "password");
			CallableStatement stmt = conn.prepareCall("begin emp_sal(?, ?); end;");
			stmt.setInt(1, 20);
			stmt.setDouble(2, 0.5);
			stmt.execute();

			conn.close();
		} catch (SQLException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
	}
}
```

Call Function

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

public class CallFunction {
	public static void main(String args[]) {

		Connection conn;
		try {
			conn =
DriverManager.getConnection("jdbc:oracle:thin:@coestudb.qu.edu.qa:1521/STUD.qu.edu.qa",
					"username", "password");
			CallableStatement stmt = conn.prepareCall("{? = call totalEmps()}");

			stmt.registerOutParameter (1, Types.INTEGER);

			stmt.execute();

			int totalEmps = stmt.getInt (1);

			System.out.println("Total Employees : " + totalEmps);

			conn.close();
		} catch (SQLException e) {
			e.printStackTrace();
		}
	}
}
```

EXERCISE:

- Write a java program that insert three departments, they must be inserted simultaneously in a database.
- Write a java program that display all information in department table.
- Write a java program that :
  - Add new employee with empno and employee name who is hired today
  - Show ename, job, and hiredate for all employees who work in Research Department