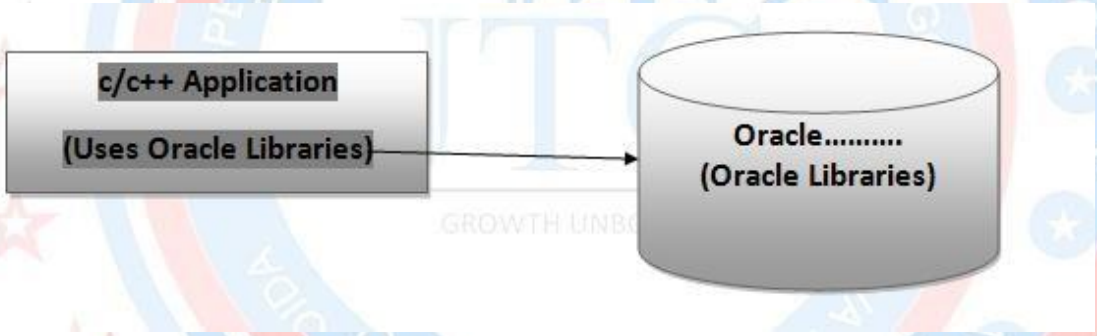
Introduction

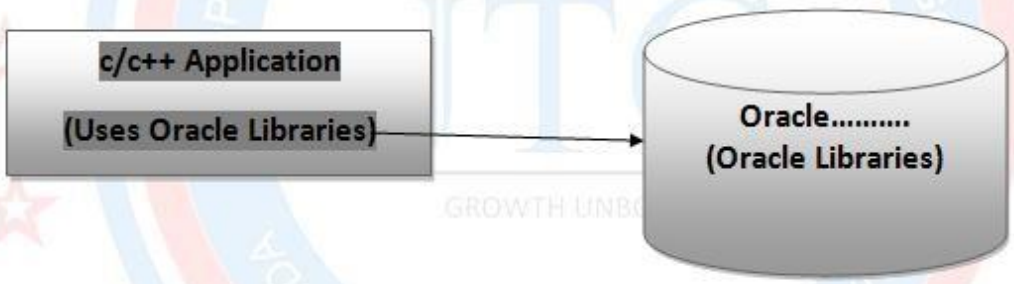
- JDBC is a technology which is used to interact with the database from java Application.
- JDBC Technology is a part of java Standard Edition.
- JDBC is a Specification provided by java vendor and implemented by java Vendor or DB Vendor.

JDBC Versions

- JDBC 3.0 is released under J2SE 1.4.2.
- No updation under J2SE 5.0.
- JDBC 4.0 is released under Java SE 6.
- JDBC 4.1 is released under Java SE 7.
- JDBC 4.2 is released under Java SE 8.
- If you want to intract with database using c or c++, you need to use database specific libraries in your application directly.
- In the below diagram, c or c++ application wants to intract with Oracle database. So it is using Oracle libraries directly

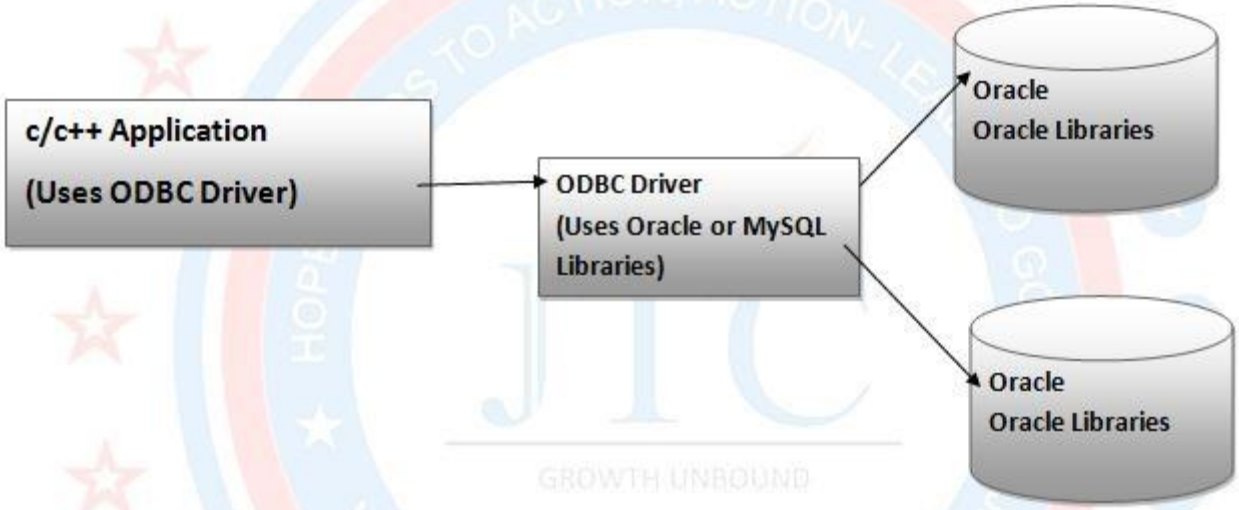


- Later when you migrate the database to another database then you then to rewrite the entire application using new database specific libraries.
- In the below diagram, your c or c++ application wants to intract with MySQL database. So you have to rewrite entire application using MySQL libraries.
- This increases the maintance of the application.



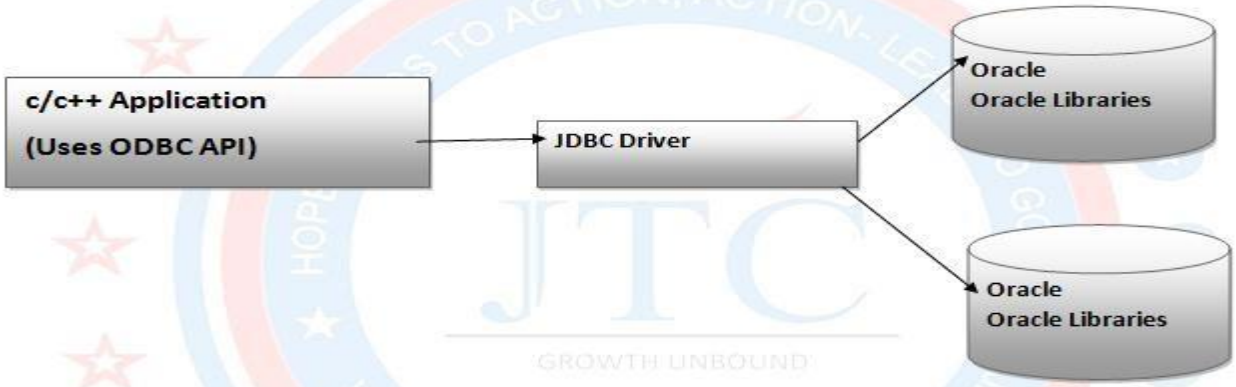
- To avoid the above said maintains problem, Microsoft has introduced ODBC Driver.
- ODBC stands for Open DataBase Connectivity.
- With ODBC driver, you no need to use database specific libraries directly in your application.
- Your application now intract with ODBC driver instead of using database specific librearies directly and ODBC Driver intracts with database specific libraries.

- Now when you migrate the database to another database then you no need to rewrite the entire application.you can just change ODBC Configuration.



- ODBC Driver setup is available only on windows operating system and also ODBC is not good in terms of performance.
- To avoid these limitations, SUN has provided JDBC API and JDBC Drivers.

- JDBC API and JDBC Drivers are Java Based Programs which runs on any Operating System.



- Java Program which is using JDBC API is called as JDBC Program.
- Two packages provided under JDBC API called:
  - Java.sql
  - Javax.sql
- Various classes and interfaces are provided under above two packages.

Java.sql package

DriverManager	Driver	Connection	Statement
PreparedStatement	CallableStatement	ResultSet	DatabaseMetadata
ResultSetMetadata	Types		

Javax.sql package

RowSet	JdbcRowSet	CachedRowSet	DataSource

Steps to Write JDBC Program

- Step 1: Load the Driver class.
- Step 2:Establish the Connection between JDBC Program and Database.
- Step 3:Prepare the SQL Statement.
- Step 4:Create the JDBC Statement.
- Step 5:Submit the SQL Statement to Database using JDBC Statement.
- Step 6:Process the result.
- Step 7:Close all the resources.

Types Of JDBC Drivers

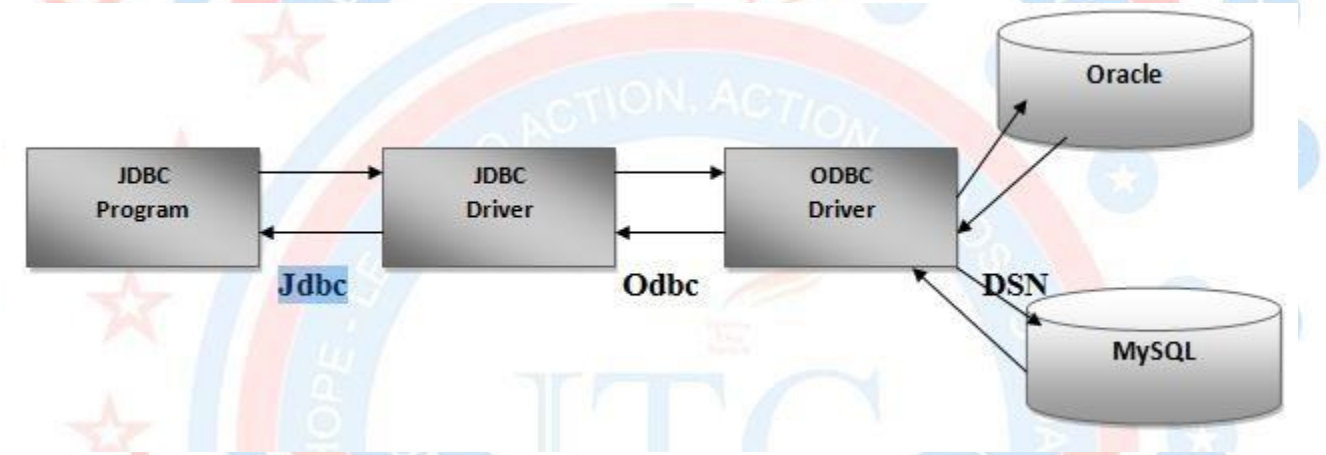
- There are 4 types of JDBC Drivers.
  - Type 1 Driver JDBC ODBC Bridge Driver
  - Type 2 Driver Partial Java and Partial Native Driver

- Type 3 Driver Net Protocol Driver
- Type 4 Driver Pure Java Driver

Type 1 Driver

Name	JDBC ODBC Bridge Driver
Vendor	Java Vendor
Driver Class	Sun.jdbc.odbc.JdbcOdbcDriver
URL	Jdbc:odbc:<Data Source Name>
Username	<Database Username>
Password	<Database Password>
Softeware Required	DB,Java,ODBC Drivers

Architecture:



Create the following table in Oracle and MySQL:

Create table jtcstudents(sid int primary key,sname varchar(10),email varchar(15),phone long);

Steps to Configure ODBC Data Source Name for Oracle

- Open controlpanel
- Open Administrative Tools
- Open Data Sources(ODBC)
- Click on add button under User DSN tab
- Select the Oracle in XE from the list

- Click on Finish
- Provide the following information
  - Data Source Name JTCORADSN
  - TNS Service Name XE
  - User Id system(Oracle username)
- To test the connection click on Test Connection
  - Provide the Oracle Password
  - Click o Ok
- Click on the OK button on the Configuration Window.
- Click on OK Button of ODBC Administrator Window

Steps to Configure ODBC Data Source Name for MySQL

- Install ODBC Driver using MySQL-connector-odbc-5.2.5- win-32.msi from Student DVD
- Open Control Panel
- Open Administrative Tools
- Open Data Sources (ODBC)
- Click on Add button under the User DSN tab
- Select the MySQL ODBC 5.2 ANSI Driver from the list
- Click on Finish
- Provide the following information
  - Data Source Name JTCMYDSN
  - TCP/IP Server localhost
  - User root (MySQL username)
  - Password JTCindia (MYSQL password)
- Select the database from the list jtcdb
- To test the connection click on Test
  - Click on OK button on the Configuration Window
- Click on OK Button of ODBC Administration Window

JTC 1.java

```
package com.jtcindia.jdbc;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
/*
```

```
 * @Author:somprakash rai
 *@company: java training Center
 *@see:www.jtcindia.org
*/
public class Jtc1 {
	static{

		Connection con=null;
		Statement st=null;
		try{
			//step 1:Load the drive class.
			Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
			//step2. Establish the Connection.
			con=DriverManager.getConnection("jdbc:odbc:JTCORADSN",
"root","root");
			// step 3: Prepare the SQL Statement.
			String sql="insert into jtcstudents
values(99,'som','som@jtc.com','123345567'";
			// Step 4: Create the JDBC Statement.
			st=con.createStatement();
			//Step 5: Submit the SQL Statement to Database using JDBC
Statement.
			int x=st.executeUpdate(sql);
			// Step 6:Process the result.
			if(x==1){
				System.out.println("Record inserted");
			}else{
				System.out.println("Record Not Inserted");
			}

		}catch(Exception e){
			e.printStackTrace();
		}finally{
			// Step 7:Close all the resources.
			try{
				if(st!=null) st.close();
				if(con!=null) con.close();
			}catch(Exception e){
				e.printStackTrace();
```

```
                }
            }
        }

}
```

JTC 2.java

```
package com.jtcindia.jdbc;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;
/*
 * @Author:somprakash rai
 *@company: java training Center
 *@see:www.jtcindia.org
*/
public class Jtc2 {
        static{

                Connection con=null;
                Statement st=null;
                try{
//step 1:Load the drive class.
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
//step2. Establish the Connection.
        con=DriverManager.getConnection("jdbc:odbc:JTCORADSN", "root","root");
// step 3: Prepare the SQL Statement.
        String sql="insert into jtcstudents values(88,'som','som@jtc.com','123345567'";
// Step 4: Create the JDBC Statement.
        st=con.createStatement();
//Step 5: Submit the SQL Statement to Database using JDBC Statement.
                int x=st.executeUpdate(sql);
// Step 6:Process the result.
        if(x==1){
        System.out.println("Record inserted");
                        }else{
```

```
                System.out.println("Record Not Inserted");
                        }
                }catch(Exception e){
                        e.printStackTrace();
                }finally{
                        // Step 7:Close all the resources.
                        try{
                                if(st!=null) st.close();
                                if(con!=null) con.close();
                        }catch(Exception e){
                                e.printStackTrace();
                        }
                }
        }

}
```

JTC3.java

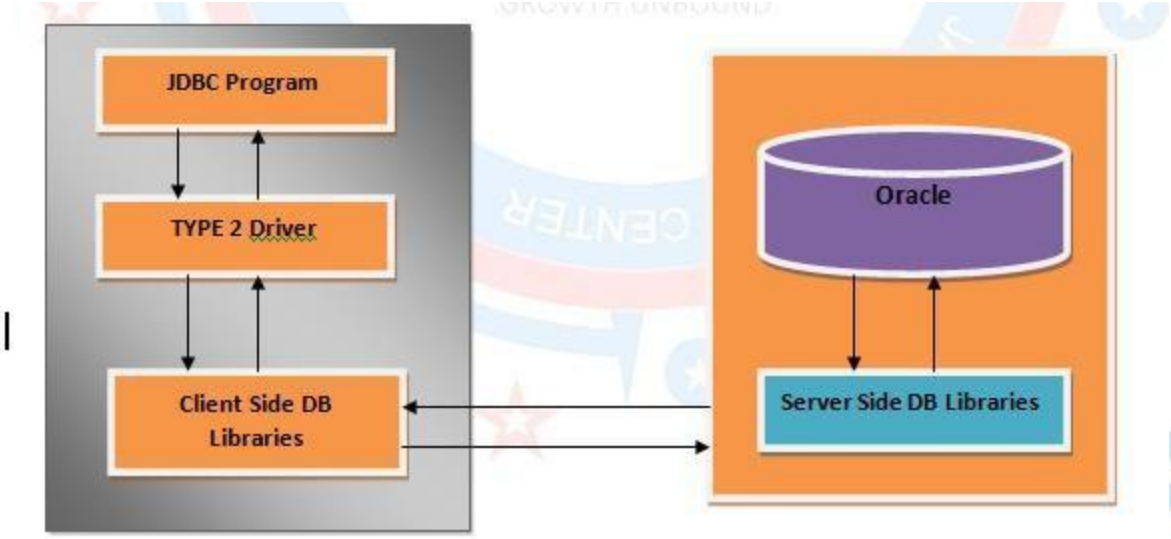
```
package com.jtcindia.jdbc;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
/*
 * @Author:somprakash rai
 *@company: java training Center
 *@see:www.jtcindia.org
*/
public class Jtc3 {
        static{

                Connection con=null;
                Statement st=null;
                try{

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

```
con=DriverManager.getConnection("jdbc:odbc:JTCORADSN", "root","root");
                String sql="select * from jtcstudents";
                st=con.createStatement();
                ResultSet rs=st.executeQuery(sql);
                while(rs.next()){
                        int sid=rs.getInt(1);
                        String sn=rs.getString(2);
                        String em=rs.getString(3);
                        String ph=rs.getString(4);
                        System.out.println(sid+"\t"+sn+"\t"+em+"\t"+ph);
                }
        }catch(Exception e){
                e.printStackTrace();
        }finally{
                try{
                        if(st!=null) st.close();
                        if(con!=null) con.close();
                }catch(Exception e){
                        e.printStackTrace();
                }
        }
    }
}
```

Type 2 Driver:

Name	**Partial native partial java Driver**
Vendor	DB Vendor
Driver Class	**Oracle.jdbc.driver.OracleDriver**
URL	Jdbc:oracle:oci8:@hostname:port:serviceName Ex: jdbc:oracle:oci8:@localhost:1521:XE
Username	<Database Username>
Password	<Database Password>
Software Required	Database client Server Edition,Java

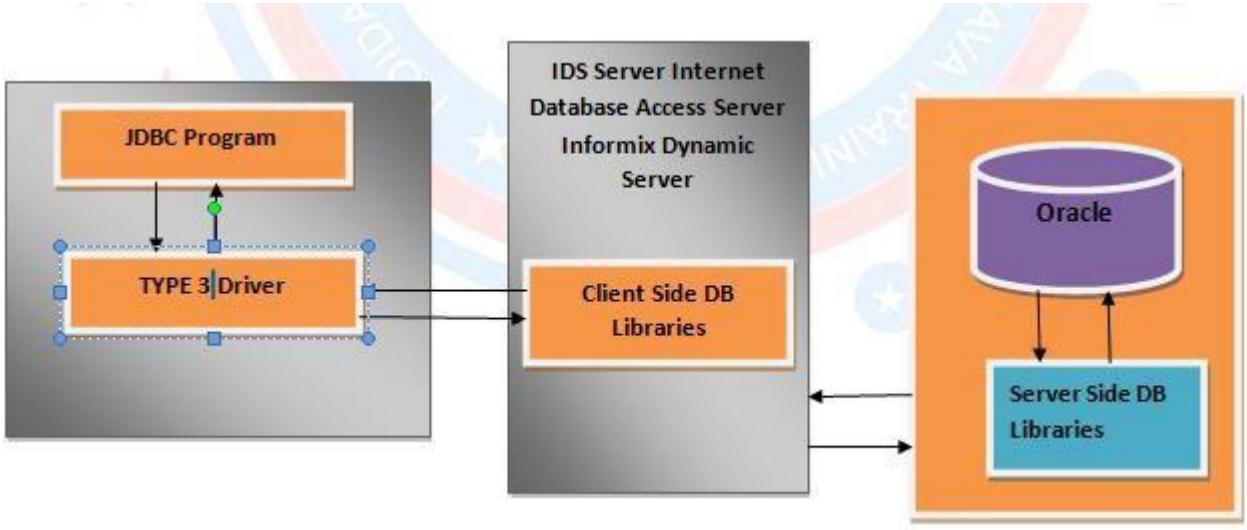
Architecture:



Type 3 Driver

Name	Net Protocol
Vendor	Java Vendor
Driver Class	Com.ids.Driver
URL	Jdbc:ids://hostname…
Username	<Database Username>
Password	<Database Password>
Softeware Required	IDS Server, Database Client Server Edition,java

Architecture:



Type 4 Driver

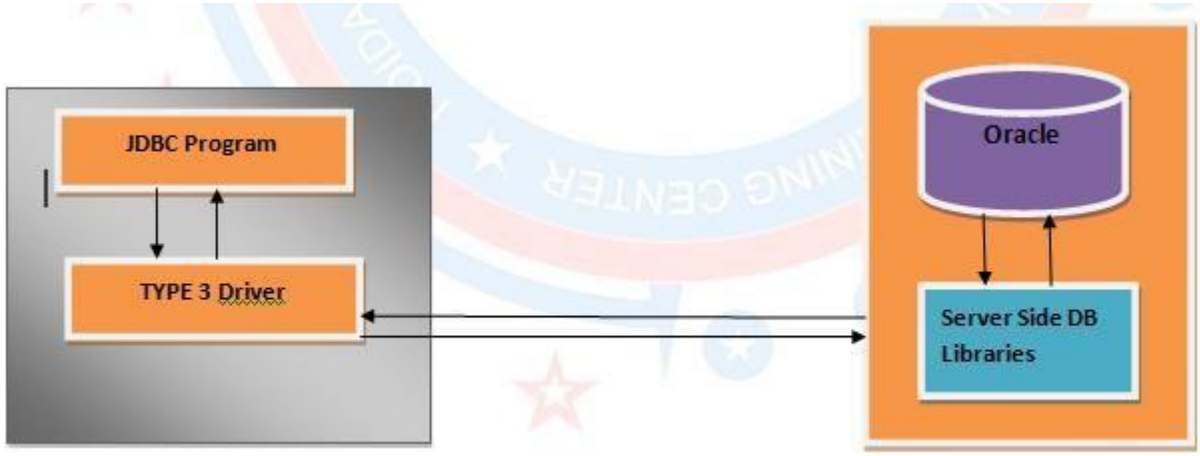
Name	Pure Java Driver
Vendor	Java Vendor
Username	<Database Username>
Password	<Database Password>
Softeware Required	Database,Java

For Oracle Driver

Driver class	**Oracle.jdbc.driver.OracleDriver**
Url	**Jdbc:oracle:thin:@<host>:<port>:<serviceName>** **Ex: jdbc:oracle:thin:@localhost:1521:XE**
Class Path	**Ojdbc14.jar** **Ojdbc6.jar**

For MySQL Driver

Driver class	**Com.mysql.jdbc.Driver**
Url	**Jdbc:mysql://<host>:<port>:/<dbName>** **Ex: jdbc:mysql://localhost:3306/jdbc**
Class Path	**Mysql.jar**



JDBCUtil.java

```
package jdbcType4Driver;

import java.sql.Connection;

import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
/**@Author:somprakash rai
*@company:java Training Center
*@see:www.jtcindia.org*/

public class JdbcUtil {
	public static Connection getOraConnection() throws SQLException,
ClassNotFoundException{
		Class.forName("oracle.jdbc.driver.OracleDriver");
		String url="jdbc:oracle:thin:@localhost:1521:XE";
		Connection con=DriverManager.getConnection(url,"system","jtcsom");
		return con;

	}

	public static Connection getMySQLConnection() throws ClassNotFoundException,
SQLException{
		Class.forName("com.mysql.jdbc.Driver");
		String url="jdbc:mysql://localhost:3306/jdbc";
		Connection con=DriverManager.getConnection(url,"root","root");
		return con;

	}
	public static void cleanup(Statement st,Connection con){
		try{
			if(st!=null) st.close();
```

```
                if(con!=null) con.close();
            }catch(Exception e){
                e.printStackTrace();
            }
        }


        public static void cleanup(ResultSet rs,Statement st,Connection con){
            try{
                if(rs!=null) rs.close();
                if(st!=null) st.close();
                if(con!=null) con.close();
            }catch(Exception e){
                System.out.println(e);
            }
        }

}
```

Jtc4.java

```
package jdbcType4Driver;

import java.sql.Connection;
import java.sql.Statement;

public class Jtc4 {
public static void main(String ar[]){
Connection con=null;
Statement st=null;

try{
//con=JdbcUtil.getMySQLConnection();
con=JdbcUtil.getOraConnection();
String qry="insert into students
values(77,'Som','som@jtc.com','9990399111')";
st=con.createStatement();
int x=st.executeUpdate(qry);
if(x==1){
System.out.println("Record inserted");
        }else{
System.out.println("Record Not Inserted");
        }
}catch(Exception e){
e.printStackTrace();
        }finally{
JdbcUtil.cleanup(st, con);
        }
```

Jtc5.java

```
package jdbcType4Driver;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.Statement;
/**@Author:somprakash rai
*@company:java Training Center
*@see:www.jtcindia.org*/
public class Jtc5 {
public static void main(String ar[]){
        Connection con=null;
        Statement st=null;
        ResultSet rs=null;

try{
//con=JdbcUtil.getMySQLConnection();
con=JdbcUtil.getOraConnection();
String qry="select * from students";
st=con.createStatement();
rs=st.executeQuery(qry);
if(rs.next()){do{
int id=rs.getInt(1);
String  name=rs.getString(2);
String  email=rs.getString(3);
String phone=rs.getString(4);
System.out.println(id+"\t"+name+"\t"+email+"\t"
+phone);
```

```
        }
}
```

```
} while(rs.next());
}else
System.out.println("Record Not Inserted");

}catch(Exception e){
e.printStackTrace();
}finally{
JdbcUtil.cleanup(st, con);
        }
        }
}
```

Pros and Cons of Types Of Drivers

	Advantages	Disadvantages
Type 1 Driver	• Type 1 is very easy to use and maintain. • Type 1 is suitable for migrating application to java without changing existing ODBC setup. • No Extra software is required for the Type 1 implementation. • Performance of the Type 1 is acceptable.	• Type 1 driver implementation is possible in window OS only becouse ODBC driver avaialable only with windows. • Performance of this driver is not excellent but acceptable.
Type 2 Driver	• Type 2 is faster than all other drivers.	• In type 2 both client and server machine will be have the database library. • When database is migrated then you will be grt much maintance because you need to re-install client side libraries in all the client machine.
Type 3 Driver	• In type 3,client side DB libraries are moved to middleware server called IDS server. • Because of this,client side maintance is reduce.	• You need to purches extra software called IDS server. • Because of having middlware server between your program and database server,performance will be reduced.
Type 4 Driver	• This driver is best among all the drivers and highly recommended to use	• Negglible.

JDBC Statement

There are 3 Type of JDBC Statement:

- **Statement**
- **PreparedStatement**
- **CallableStatement**

Statement :-

- **Statement is an interface available in java.sql package.**
- **Subclass of statement interface is provide by Drive vendor.**
- **You can create the Statement using the following methods of Connection interface.**
  - **public Statement createStatement()**
  - **public Statement createStatement(int , int)**
  - **public Statement createStatement(int ,int ,int)**

- **After creating the Statement object , you can call one of the following methods to submit the SQL Statement to Database.**
  - **public int executeUpdate(String sql)**
  - **public boolean execute(String sql)**
  - **public ResultSet executeQuery(String sql)**
- **Public int executeUpdate(String sql)When you want to submit insert or update or delete SQL Statements then use executeUpdate() method which returns the number of records inserted or updated or deleted.**
- **Public ResultSet executeQuery(String sql) When you want to submit Select SQL Statements then use executeQuery() method which returns the number of records fetched by select statement interms of ResultSet object.**

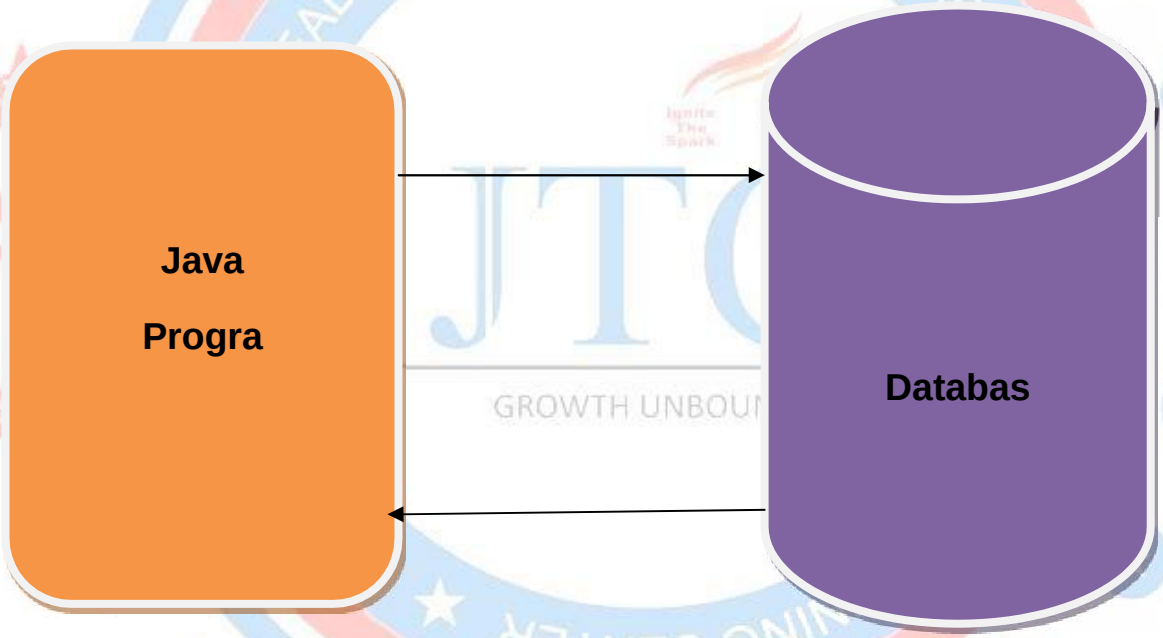
- **Public boolean execute(String sql)When you want to submit insert ,update,delete or select SQL Statements then use execute() method which returns the boolean value saying whether the ResultSet object is created or not(The SQL Statement is SELECT or not).**

  - if return value is true which means that SELECT SQL statement is submitted and ResultSet object is created.
    - Public ResultSet getResultSet()
  - If return value false which means that INSERT,UPDATE,or DELETE SQL statement is submitted and integer number is available which represent number of records inserted update or deleted.
    - Public int getUpdateCount()

- **Using the Single Statement Object , you can submit any type of SQL statement and any number of SQL statements.**
  - **ex:**

```
Statement st=con.createStatement();
 String sql1="insert.... ";
 String sql2="update.... ";
 String sql3="delete ....";
String sql4="select .... ";
booleab b1=st.execute(sql1);
int x=st.executeUpdate(sql2);
int y=st.executeUpdate(sql3);
ResultSet rs=st.executeQuery(sql4);
```

- **When you submit the SQL Statement using Statement object then SQL Statement will be compiled and executed every time.**



- **Total time = req.time + compile time + exec time + res.time**
  = 5 ms+5 ms+5 ms+5 ms = 20 ms.
  1 SQL Stmt = 20 ms.
  100 times = 2000ms.
- **If you are providing dynamic values for the query then you need to use concatination operator, Formattor or StringBuffer etc to format the query.**
- **If you are providing the value that format is database dependent (May be Date) then you need to provide depending on Database.**

Jtc6.java

```
package jdbcType4Driver;

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

public class Jtc6 {
public static void main(String arg[]){
Connection con=null;
Statement st=null;
try{
//con=JdbcUtil.getMySQLConnection();
con=JdbcUtil.getOraConnection();
Scanner sc=new Scanner(System.in);
System.out.println("Enter Id");
        int id=sc.nextInt();
        sc.nextLine();
System.out.println("Enter Name");
String name=sc.nextLine();
System.out.println("Enter Email:");
String email=sc.nextLine();
System.out.println("Enter Phone:");
String phone=sc.nextLine();
String qry=String.format("insert into students
values(%d,'%s','%s','%s')",id,name,email,phone)
;
System.out.println(qry);
st=con.createStatement();
int x=st.executeUpdate(qry);
if(x==1){
System.out.println("Record inserted
succesfully");
}else{
System.out.println("not inserted");

                                }
            }catch(Exception e){
            e.printStackTrace();
        }finally{
        JdbcUtil.cleanup(st, con);              }
        }

}
```

Jtc7.java

```
package jdbcType4Driver;

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

public class Jtc7 {
public static void main(String arg[]){
        Connection con=null;
        Statement st=null;
try{
//      con=JdbcUtil.getMySQLConnection();
        con=JdbcUtil.getOracleConnection();
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter Id");
        int id=sc.nextInt();
        sc.nextLine();
System.out.println("Enter Name");
String name=sc.nextLine();
System.out.println("Enter Email:");
String email=sc.nextLine();
System.out.println("Enter Phone:");
String phone=sc.nextLine();
String qry=String.format("select * from
students",id,name,email,phone);
System.out.println(qry);
ResultSet rs=st.executeQuery(qry);
        if(rs.next()){
        int id1=rs.getInt(1);
String name1=rs.getString(2);
String email1=rs.getString(3);
String phone1=rs.getString(4);
System.out.println(id1+"\t"+name1+"\t"+email1+"\t
"+phone1);

        }else{System.out.println("sorry,Student not
found");
                        }catch(Exception e){
        e.printStackTrace();
                }finally{JdbcUtil.cleanup(st, con);
                }
        }
}
```

Jtc8.java

```
        int id=rs.getInt(1);
```

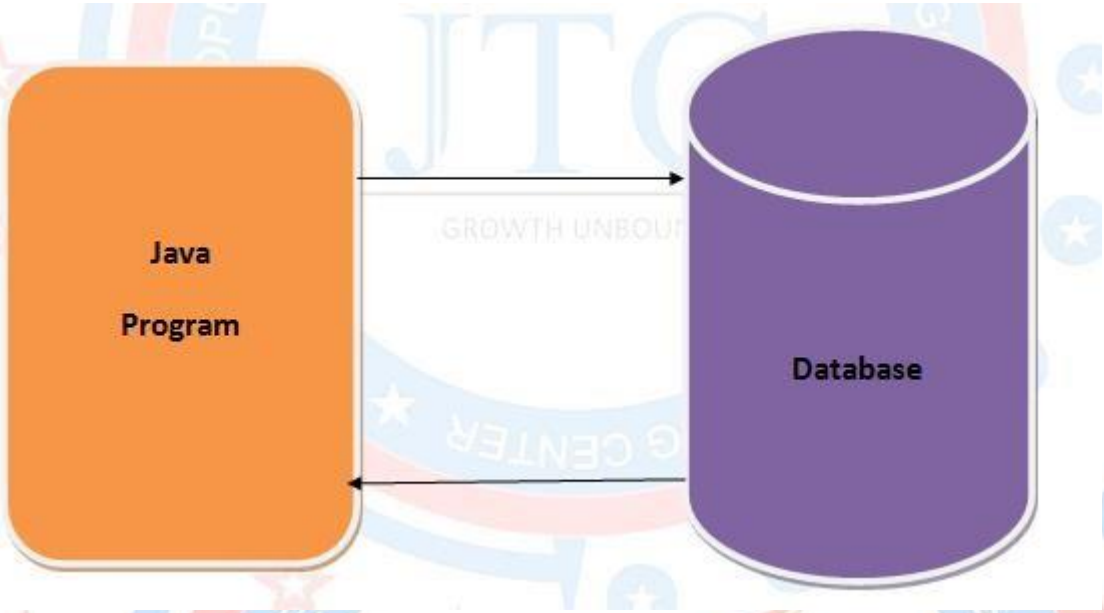
```
package jdbcType4Driver;

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

public class Jtc8 {
	public static void main(String ar[]){
		Connection con=null;
		Statement st=null;
		ResultSet rs=null;
		try{

	con=JdbcUtil.getOraConnection();
	Scanner sc=new Scanner(System.in);
System.out.println("Enter Query:");
	String qry=sc.nextLine();
	st=con.createStatement();
	boolean b1=st.execute(qry);
	if(b1){
	rs=st.getResultSet();
		if(rs.next()){
			do{
		String name=rs.getString(2);
		String email=rs.getString(3);
		String phone=rs.getString(4);
System.out.println(id+"\t"+name+"\t"+email+"\t"+p
hone);

		}while(rs.next());
					}
				}else{
		int x=st.getUpdateCount();
		System.out.println("Result:"+x);
					}

		}catch(Exception e){
				e.printStackTrace();
			}finally{
				JdbcUtil.cleanup(st, con);
			}


		}
}
```

==========

2. PreparedStatement :

- PreparedStatement is an interface available in java.sql package and extends Statement interface.
- You can create the PreparedStatement using the following methods of Connection interface.
  - public PreparedStatement prepareStatement(sql)
  - public PreparedStatement prepareStatement(sql,int , int)
  - public PreparedStatement prepareStatement(sql,int ,int ,int)
- After creating the PreparedStatement object , you can call one of the following methods to submit the SQL Statement to Database.
  - public int executeUpdate()
  - public boolean execute()
  - public ResultSet executeQuery()
- Using the Single PreparedStatement Object , you can submit only one SQL statement.

```
ex:
String sql="insert.... ";
PreparedStatement ps=con.prepareStatement(sql)
int x=ps.executeUpdate();
```



- **When you submit the SQL Statement using PreparedStatement object then SQL Statement will be compiled only once first time and pre-compile SQL Statement will be executed every time.**
- **Total time = req.time + compile time + exec time + res.time**
  = 5 ms+5 ms+5 ms+5 ms = 20 ms.
  First time -> 1 SQL Stmt = 20 ms.
  next onwards -> 5ms + 0 ms+ 5 ms +5 ms= 15 ms.
  101 times = 20 ms + 1500ms.=> 1520.
- **PreparedStatement gives you the place holder mechanism for providing the data dynamic to the query.You need to use ? symbol for placeholder.**
- **To provide the value of place holder you need to invoke the following method depending the type of the value for place holder**
  public void setX(int paramIndex, X val)
  X can be Int,String,Long,Float,Date etc

- **If you want to specify the date type value then create the object of java.sql.Date type and invoke the following method**
  public void setDate(int paramINdex,Date dt)

```
String sql="insert into jtcstudents values(?,?,?,?,?,?)";
ps=con.prepareStatement(sql);
ps.setInt(1, id);
ps.setString(2,nm);
ps.setString(3,eml);
ps.setLong(4,phn);
ps.setString(5,fee);
ps.setDate(6,dt);
```

Jtc9.java

```
package jdbcType4Driver;

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

public class Jtc9 {
	public static void main(String arg[]){
		Connection con=null;
		Statement st=null;
		PreparedStatement ps=null;
		try{
		//
	con=JdbcUtil.getMySQLConnection();

	con=JdbcUtil.getOraConnection();
			Scanner sc=new
Scanner(System.in);
			System.out.println("Enter
Id");
			int id=sc.nextInt();
			sc.nextLine();
			System.out.println("Enter
Name");
			String
name=sc.nextLine();
			System.out.println("Enter
Email:");
			String
email=sc.nextLine();
			System.out.println("Enter
Phone:");
			String
phone=sc.nextLine();
			String
qry1=String.format("insert into students
values(?,?,?,?)");
System.out.println(qry1);
ps=con.prepareStatement(qry1);
		ps.setInt(1,id);
		ps.setString(2,name);
		ps.setString(3,email);
		ps.setString(4,phone);
	int x=ps.executeUpdate();
```

Jtc10.java

```
package jdbcType4Driver;

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

public class Jtc10 {
	public static void main(String arg[]){
		Connection con=null;
		Statement st=null;
		CallableStatement cs=null;
		try{
		//
	con=JdbcUtil.getMySQLConnection();

	con=JdbcUtil.getOraConnection();
			Scanner sc=new
Scanner(System.in);
			System.out.println("Enter
Id");
			int id=sc.nextInt();
			sc.nextLine();
			System.out.println("Enter
Name");
			String name=sc.nextLine();
			System.out.println("Enter
Email:");
			String email=sc.nextLine();
			System.out.println("Enter
Phone:");
			String phone=sc.nextLine();

				String
qry2=String.format("select * from students");

	ps=con.prepareStatement(qry2);
			System.out.println(qry2);
			ResultSet
rs=ps.executeQuery();
			if(rs.next()){
				int id1=rs.getInt(1);
				String
name1=rs.getString(2);
				String
email1=rs.getString(3);
```

```
        if(x==1){
System.out.println("Record inserted
succesfully");
                        }else{

        System.out.println("not inserted");

                                }

                        }

                        String qry3=sc.nextLine();

        ps=con.prepareStatement(qry3);
                        boolean b1=ps.execute();
                        if(b1){

        rs=ps.getResultSet();
                                if(rs.next()){
                                        do{
                                                int
id1=rs.getInt(1);

        String name1=rs.getString(2);

        String email1=rs.getString(3);

        String phone1=rs.getString(4);

        System.out.println(id1+"\t"+name1+"\t"
+email1+"\t"+phone1);

        }while(rs.next());
                                }
                                }else{
                                        int
x1=ps.getUpdateCount();

        System.out.println("Result:"+x1);
                        }


                }catch(Exception e){
                        e.printStackTrace();
                }finally{
                        JdbcUtil.cleanup(st, con);
                }
```

```
                                String
phone1=rs.getString(4);

        System.out.println(id1+"\t"+name1+"\t"+ema
il1+"\t"+phone1);

                                }else{
System.out.println("sorry,Student not found");


                        }catch(Exception e){
                                e.printStackTrace();
                        }finally{
                                JdbcUtil.cleanup(st, con);
                        }

        }

}
```

```
        }

}
```

Jtc11.java

```
package jdbcType4Driver;

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

public class Jtc9 {
        public static void main(String arg[]){
                Connection con=null;
                Statement st=null;
                PreparedStatement ps=null;
                try{
                //
        con=JdbcUtil.getMySQLConnection();

        con=JdbcUtil.getOraConnection();
                        Scanner sc=new
Scanner(System.in);
                        System.out.println("Enter
Id");
                        int id=sc.nextInt();
                        sc.nextLine();
                        System.out.println("Enter
Name");
                        String
name=sc.nextLine();
                        System.out.println("Enter
Email:");
                        String
email=sc.nextLine();
                        System.out.println("Enter
Phone:");
                        String
phone=sc.nextLine();
ps=con.prepareStatement(qry3);
                        boolean
                        if(b1){

                rs=ps.getResultSet();
                                if(rs.next()){
                                        do{
                                                int
id1=rs.getInt(1);
                                                String
name1=rs.getString(2);
                                                String
email1=rs.getString(3);
                                                String
phone1=rs.getString(4);

                System.out.println(id1+"\t"+name1+"\t"+
email1+"\t"+phone1);

                }while(rs.next());
                                }
                                }else{
                                        int
x1=ps.getUpdateCount();

                System.out.println("Result:"+x1);
                        }
```

```
b1=ps.execute();
```

CallableStatement :

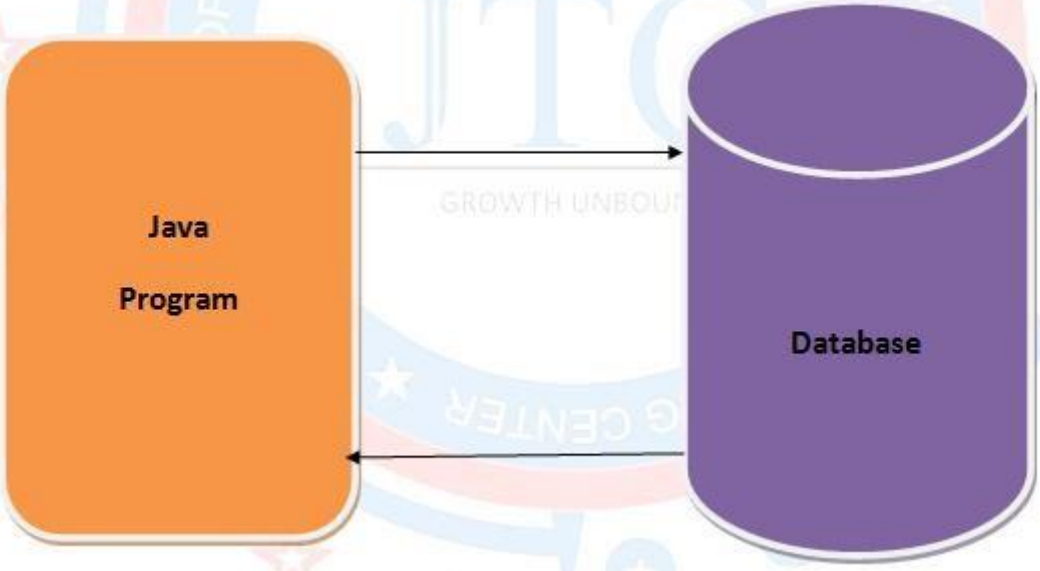
- **CallableStatement is an interface available in java.sql package and extends PreparedStatement interface.**
- **You can create the CallableStatement using the following methods of Connection interface.**
  - **CallableStatement prepareCall(String)**
  - **CallableStatement prepareCall(String,int , int)**

- CallableStatement prepareCall(String,int ,int ,int)

- After creating the CallableStatement object , you can call one of the following methods to submit the SQL Statement to Database.
  - int executeUpdate()
  - boolean execute()
  - ResultSet executeQuery()
- CallableStatement is designed mainly to invoke the stored procedures running in the database.
- Stored procedure is pre-compiled procedure .i.e When you create the procedure then that procedure will be compiled and stored in database memory. When you make call to the procedure then that pre-compiled procedure will be executed directly.
- Using the Single CallableStatement Object , you can make a call to only one stored procedure.
  ex:

```
String sql="call p1(?,?)";
CallableStatement cs=con.prepareCall(sql)
cs.setInt(1,10);
cs.setInt(2,20);
int x=cs.executeUpdate();
```

- Use Stored Procedures when you want to run some logic in database.



with SP using CS.

- **Total time = req.time + compile time + exec time + res.time**
  = 5 ms+0 ms+20 ms+5 ms = 30 ms.101 times -> 3030ms.

with SQL using PS.

- **Total time = req.time + compile time + exec time + res.time**
  = 5 ms+0 ms+20 ms+5 ms = 30 ms.

One JDBC Program with 4 SQL.
4 SQL's -> 4 * 20 ms =80 ms. (With Statement)
-> 4 * 15 ms= 60 ms
101 times = 80 ms + 6000ms.=> 6080 ms.

- **CallableStatement gives you the place holder mechanism.**

A) jtcstudents table.
B)insertStudentInfo() Procedure

For MySQL:

```
delimiter $
create procedure insertStudentInfo(id int,nm varchar(20),eml varchar(50),phn long,fee float,dob date)
begin
insert into jtcstudents values(id,nm,eml,phn,fee,dob);
end;
$
delimiter ;
```

For Oracle:

```
create or replace procedure insertStudentInfo(id in int,nm in varchar,eml varchar,phn long,fee float,dob
date)
as
begin
insert into jtcstudents values(id,nm,eml,phn,fee,dob);
end;
/
```

Jtc12.java

```
package jdbcType4Driver;

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

Jtc13.java

```
package jdbcType4Driver;

import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.Statement;
import java.sql.Types;
import java.util.Scanner;
```

```
public class Jtc10 {
	public static void main(String arg[]){
		Connection con=null;
		Statement st=null;
		CallableStatement cs=null;
		try{
		//
	con=JdbcUtil.getMySQLConnection();

	con=JdbcUtil.getOraConnection();
			Scanner sc=new
Scanner(System.in);

	System.out.println("Enter Id");
			int id=sc.nextInt();
			sc.nextLine();

	System.out.println("Enter Name");
			String
name=sc.nextLine();

	System.out.println("Enter Email:");
			String
email=sc.nextLine();

	System.out.println("Enter Phone:");
			String
phone=sc.nextLine();


	cs=con.prepareCall("insert
students(?,?,?,?");
			cs.setInt(1,id);
			cs.setString(2,name);
			cs.setString(3,email);
			cs.setString(4,phone);
cs.execute();



	System.out.println("Record inserted
succesfully");

			//
	System.out.println("not inserted");
```

```
public class CopyOfJtc101 {
	public static void main(String arg[]){
		Connection con=null;
		Statement st=null;
		CallableStatement cs=null;
		try{
		//
	con=JdbcUtil.getMySQLConnection();

	con=JdbcUtil.getOraConnection();
			Scanner sc=new
Scanner(System.in);
			System.out.println("Enter Id");
			int id=sc.nextInt();
			sc.nextLine();
			System.out.println("Enter
Name");
			String name=sc.nextLine();
			System.out.println("Enter
Email:");
			String email=sc.nextLine();
			System.out.println("Enter
Phone:");
			String phone=sc.nextLine();

			cs=con.prepareCall("call
updateinfo(?,?,?");
			cs.setInt(1,id);
			cs.setString(3,email);

	cs.registerOutParameter(2,Types.VARCHAR);

	cs.registerOutParameter(3,Types.VARCHAR);
cs.execute();
String nm=cs.getString(name+"\t"+phone);
System.out.println("called Successfully");


	System.out.println("Record inserted
succesfully");

			//	System.out.println("not
inserted");


		}catch(Exception e){
```

```
                }catch(Exception e){
                        e.printStackTrace();
                }finally{
                        JdbcUtil.cleanup(st, con);
                }

        }
}
```

```
                        e.printStackTrace();
                }finally{
                        JdbcUtil.cleanup(st, con);
                }

        }
}
```