What is JDBC

JDBC API is a Java API that can access any kind of tabular data, especially data stored in a Relational Database. JDBC works with Java on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX.

Why to Learn JDBC?

JDBC stands for Java Database Connectivity, which is a standard Java API for database-independent connectivity between the Java programming language and a wide range of databases.

The JDBC library includes APIs for each of the tasks mentioned below that are commonly associated with database usage.

Making a connection to a database.

Creating SQL or MySQL statements.

Executing SQL or MySQL queries in the database.

Viewing & Modifying the resulting records.

Common JDBC Components

The JDBC API provides the following interfaces and classes −

- DriverManager − This class manages a list of database drivers. Matches connection requests from the java application with the proper database driver using communication sub protocol. The first driver that recognizes a certain subprotocol under JDBC will be used to establish a database Connection.
- Driver − This interface handles the communications with the database server. You will interact directly with Driver objects very rarely. Instead, you use DriverManager objects, which manages objects of this type. It also abstracts the details associated with working with Driver objects.
- Connection − This interface with all methods for contacting a database. The connection object represents communication context, i.e., all communication with database is through connection object only.
- Statement − You use objects created from this interface to submit the SQL statements to the database. Some derived interfaces accept parameters in addition to executing stored procedures.
- ResultSet − These objects hold data retrieved from a database after you execute an SQL query using Statement objects. It acts as an iterator to allow you to move through its data.
- SQLException − This class handles any errors that occur in a database application.

Interview Question: What are the steps to connect with Database?

1. Import JDBC packages:

For Using any external class we have to import the class from the other packages, sometimes the same classes could be present under different packages.

To use JDBC we have to import the below packages.

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

2. Load and register the JDBC driver:

We have to load the class to register our JDBC with database

using Class.forName("com.mysql.jdbc.Driver");, this line of code tells the db that we are going use JDBC driver.

Internally this Driver class will register the driver by using static method called registerDriver().

```
Class.forName("com.mysql.jdbc.Driver");
```

3. Open a connection to the database:

To connect with the database we should call getConnection() static method present in DriverManager Class.

This getConnection accepts three parameters of String Type:

- url : "jdbc:mysql://host:port/dbName"
- username : user name for the provided database
- password : password for the given database.

getConnection method also returns the database object, we have to store this return value for future data operations in the database.

```
Connection conn = DriverManager.getConnection("jdbc:mysql://host:port/dbName","username","password")
```

If your database is present in local machine then you can provide you localhost details and the port number.

Below query connect to testuser database which is present in localhost (local system) using 3306 port number and username :rootUser, password :rootPassword.

```
DriverManager.getConnection("jdbc:mysql://localhost:3306/testuser","rootUser","rootPassword");
```

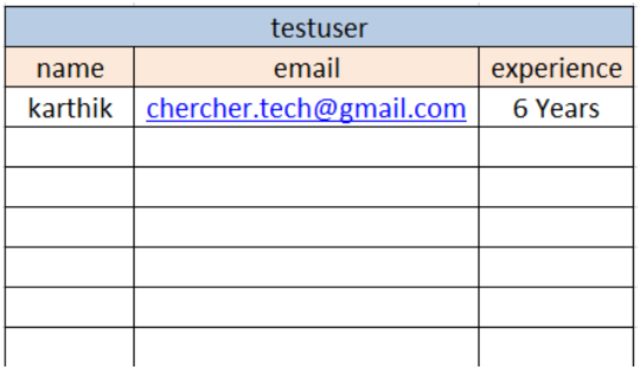
4. Create a statement object to perform a query :

We have to create a Statement object for executing our SQL queries on the database, We can create the object for Statement by calling the createStatement() method from the connection object (we created in step 3).

```
// Object of Statement. It is used to create a Statement to execute the query
Statement stmt = conn.createStatement();
```

5. Execute the statement object and return a query resultset : Now we have to form our SQL queries to execute them on the database, Queries like Select, Create, Insert, Update, Delete.

testuser table looks like below



testuser		
name	email	experience
karthik	chercher.tech@gmail.com	6 Years

```
SELECT * from testuser
```

We can execute our formed SQL query using executeQuery method present in the statement object, Results from the executed query are stored in the ResultSet Object.

The Resultset maintains a cursor that points to the current row in the result set.

```
//Object of ResultSet => 'It maintains a cursor that points to the current row in the result set'
ResultSet resultSet = stmt.executeQuery("SELECT * from testuser");
```

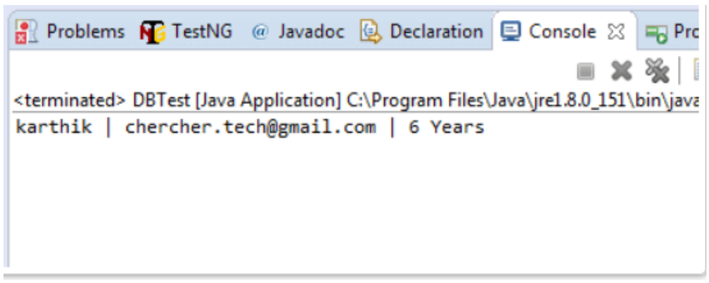
6. Process the resultset:

Once we receive our result set we can manipulate the data, now below code will display all the records present in the testuser table from the database.

Note: Index for table columns starts from 1, not from 0

```
while (resultSet .next()) {
   System.out.println(resultSet .getString(1) + " | " + resultSet .getString(2) +" | "+ resultSet .getString(3));
}
```

Output of above code



```
<terminated> DBTest [Java Application] C:\Program Files\Java\jre1.8.0_151\bin\java
karthik | chercher.tech@gmail.com | 6 Years
```

7. Close the resultset and statement objects :

After execution, we must close the connection to the resultset and statement irrespective of whether the execution passes or fails

```
resultSet.close();

stmt.close();
```

8. Close the connection :

We also should close the connection to the database, we can keep it open when we are executing something after the first operation

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

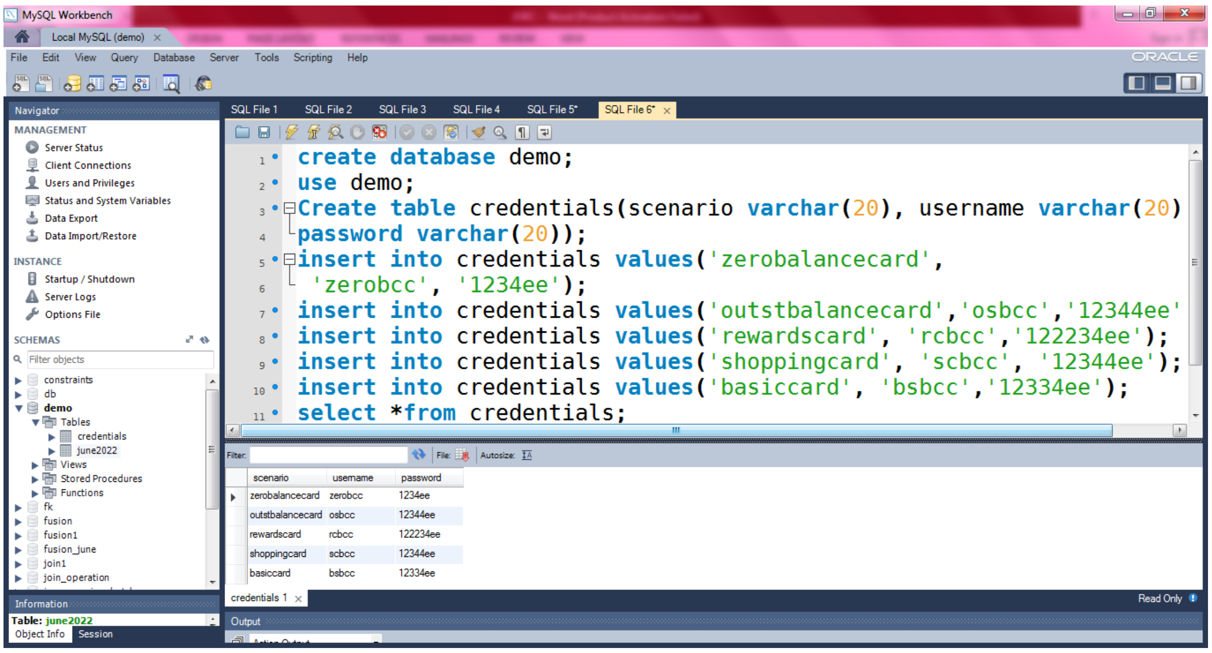
MySQL Workbench

create database demo;
use demo;

Create table credentials(scenario varchar(20), username varchar(20),
password varchar(20));

insert into credentials values('zerobalancecard',
 'zerobcc', '1234ee');

insert into credentials values('outstbalancecard','osbcc','12344ee');
insert into credentials values('rewardscard', 'rcbcc','122234ee');
insert into credentials values('shoppingcard', 'scbcc', '12344ee');
insert into credentials values('basiccard', 'bsbcc','12334ee');
select *from credentials;



```
create database demo;
use demo;
Create table credentials(scenario varchar(20), username varchar(20)
password varchar(20));
insert into credentials values('zerobalancecard',
 'zerobcc', '1234ee');
insert into credentials values('outstbalancecard','osbcc','12344ee'
insert into credentials values('rewardscard', 'rcbcc','122234ee');
insert into credentials values('shoppingcard', 'scbcc', '12344ee');
insert into credentials values('basiccard', 'bsbcc','12334ee');
select *from credentials;
```

scenario	username	password
zerobalancecard	zerobcc	1234ee
outstbalancecard	osbcc	12344ee
rewardscard	rcbcc	122234ee
shoppingcard	scbcc	12344ee
basiccard	bsbcc	12334ee

JDBC - Select Records Example

MySQL Workbench 6.0

First Need to Download jar File

MySQL Jar File - https://code.google.com/archive/p/find-ur-pal/downloads

mysql-connector-java-5.1.18-bin.jar	the driver for connecting MySQL to Java

How to configure jar file in Eclipse

1. New java project

2. Create package

3. Right click on project

4. Go to properties

5. Go to java build path

6. Go to library

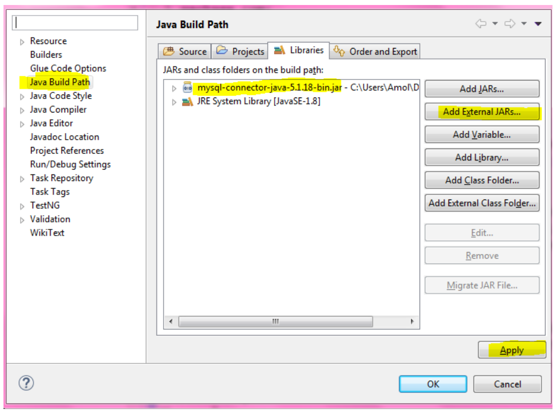
7. Click on class path

8. Add external jar file

9. Choose the correct jar file

10. Apply and close

11. You will see reference library folder below JRE



Java Build Path

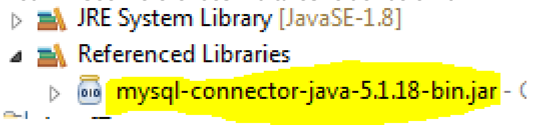
Resource
Builders
Glue Code Options
Java Build Path
Java Code Style
Java Compiler
Java Editor
Javadoc Location
Project References
Run/Debug Settings
Task Repository
Task Tags
TestNG
Validation
WikiText

Source | Projects | Libraries | Order and Export

JARs and class folders on the build path:

mysql-connector-java-5.1.18-bin.jar - C:\Users\Amol\D
JRE System Library [JavaSE-1.8]

You will see References Libraries folder below JRE



JRE System Library [JavaSE-1.8]
Referenced Libraries
mysql-connector-java-5.1.18-bin.jar - (

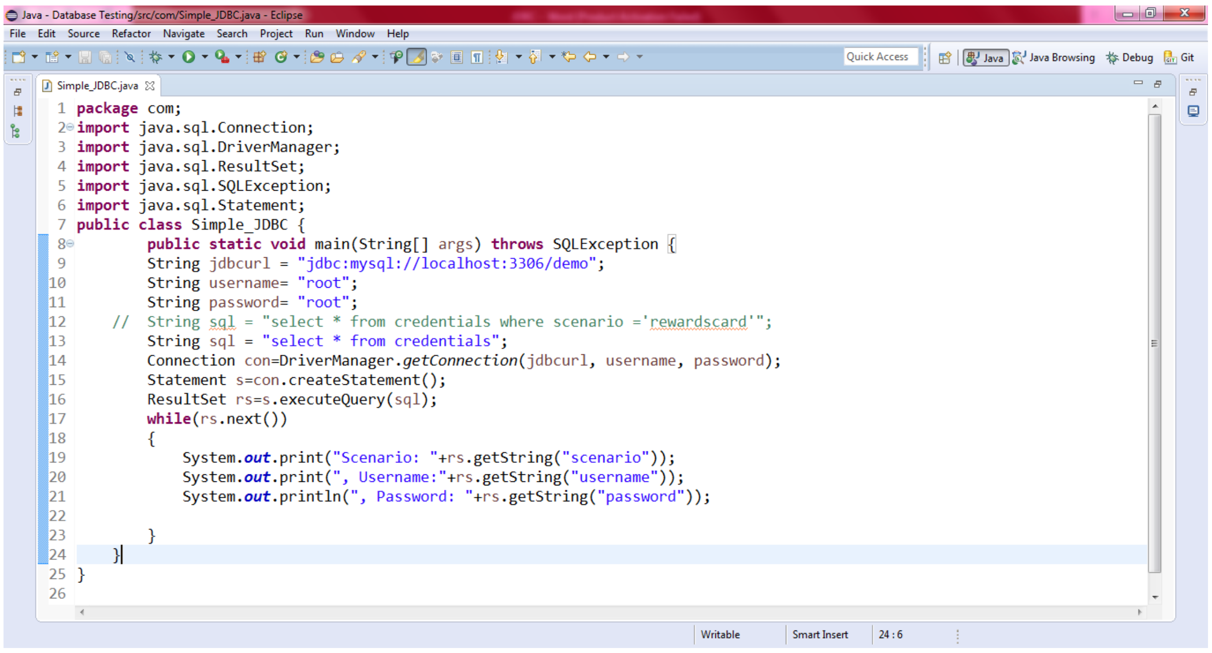
MySQL Workbench 8.0.30 jar for MySQL-connector-java

https://jar-download.com/artifacts/mysql/mysql-connector-java/8.0.30

JAVA PROGRAM

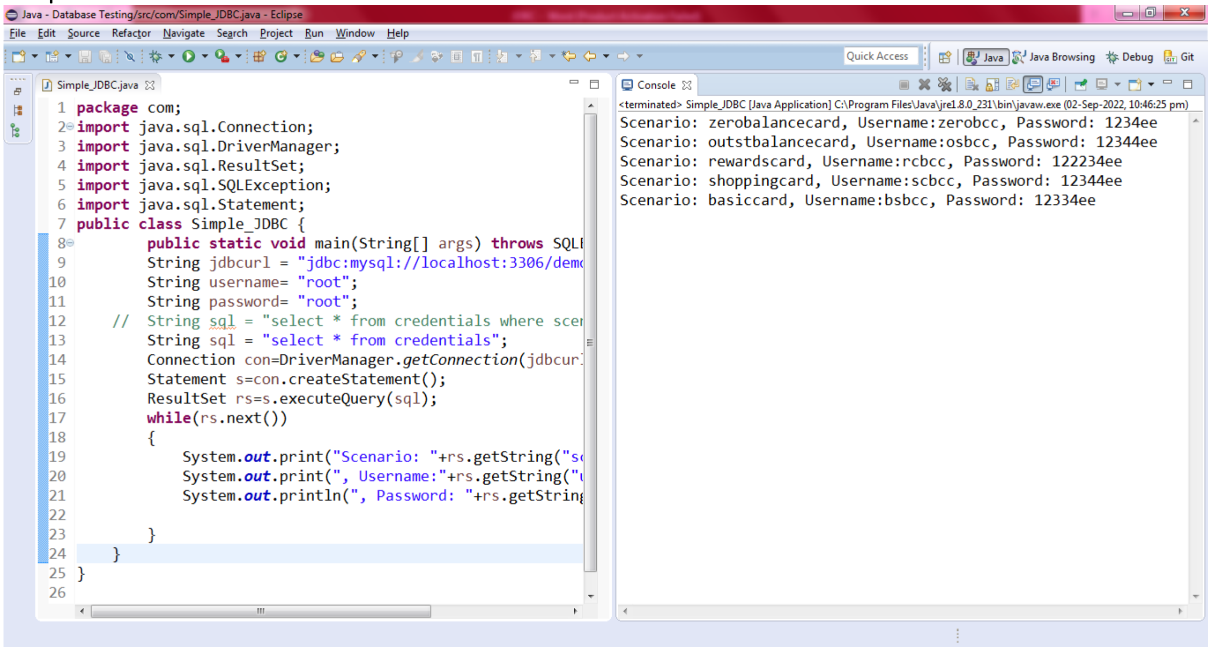
```
package com;

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

public class Simple_JDBC {

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

                String jdbcurl = "jdbc:mysql://localhost:3306/demo";
                String username= "root";
                String password= "root";
        //      String sql = "select * from credentials where scenario ='rewardscard'";

                String sql = "select * from credentials";
                Connection con=DriverManager.getConnection(jdbcurl, username, password);
```

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

                  ResultSet rs=s.executeQuery(sql);

                  while(rs.next())
                  {
                          System.out.print("Scenario: "+rs.getString("scenario"));
                          System.out.print(", Username:"+rs.getString("username"));
              System.out.println(", Password: "+rs.getString("password"));

              }
          }
}
```


Java - Database Testing/src/com/Simple_JDBC.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Java Browsing
Debug
Git
Simple_JDBC.java
Writable
Smart Insert
24 : 6

```
package com;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
public class Simple_JDBC {
        public static void main(String[] args) throws SQLException {
        String jdbcurl = "jdbc:mysql://localhost:3306/demo";
        String username= "root";
        String password= "root";
    //  String sql = "select * from credentials where scenario ='rewardscard'";
        String sql = "select * from credentials";
        Connection con=DriverManager.getConnection(jdbcurl, username, password);
        Statement s=con.createStatement();
        ResultSet rs=s.executeQuery(sql);
        while(rs.next())
        {
            System.out.print("Scenario: "+rs.getString("scenario"));
            System.out.print(", Username:"+rs.getString("username"));
            System.out.println(", Password: "+rs.getString("password"));

        }
    }
}
```

Output

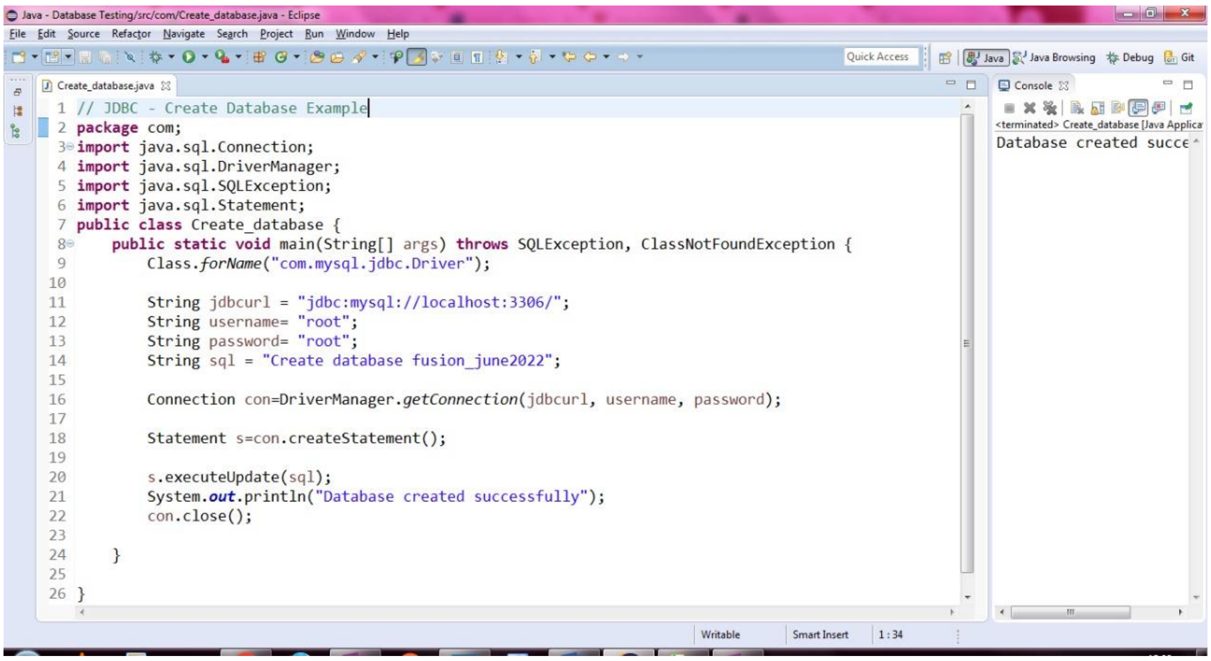

Java - Database Testing/src/com/Simple_JDBC.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Java Browsing
Debug
Git
Simple_JDBC.java
Console

```
package com;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
public class Simple_JDBC {
        public static void main(String[] args) throws SQLE
        String jdbcurl = "jdbc:mysql://localhost:3306/dem
        String username= "root";
        String password= "root";
    //  String sql = "select * from credentials where scer
        String sql = "select * from credentials";
        Connection con=DriverManager.getConnection(jdbcur
        Statement s=con.createStatement();
        ResultSet rs=s.executeQuery(sql);
        while(rs.next())
        {
            System.out.print("Scenario: "+rs.getString("s
            System.out.print(", Username:"+rs.getString("u
            System.out.println(", Password: "+rs.getString

        }
    }
}
```

```
<terminated> Simple_JDBC [Java Application] C:\Program Files\Java\jre1.8.0_231\bin\javaw.exe (02-Sep-2022, 10:46:25 pm)
Scenario: zerobalancecard, Username:zerobcc, Password: 1234ee
Scenario: outstbalancecard, Username:osbcc, Password: 12344ee
Scenario: rewardscard, Username:rcbcc, Password: 122234ee
Scenario: shoppingcard, Username:scbcc, Password: 12344ee
Scenario: basiccard, Username:bsbcc, Password: 12334ee
```

JDBC Create Database Example

```
// JDBC - Create Database Example
package com;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
public class Create_database {
        public static void main(String[] args) throws SQLException, ClassNotFoundException {
            Class.forName("com.mysql.jdbc.Driver");

                String jdbcurl = "jdbc:mysql://localhost:3306/";
                String username= "root";
                String password= "root";
                String sql = "Create database fusion_june2022";

                Connection con=DriverManager.getConnection(jdbcurl, username, password);

                Statement s=con.createStatement();

                s.executeUpdate(sql);
                System.out.println("Database created successfully");
                con.close();

        }

}
```

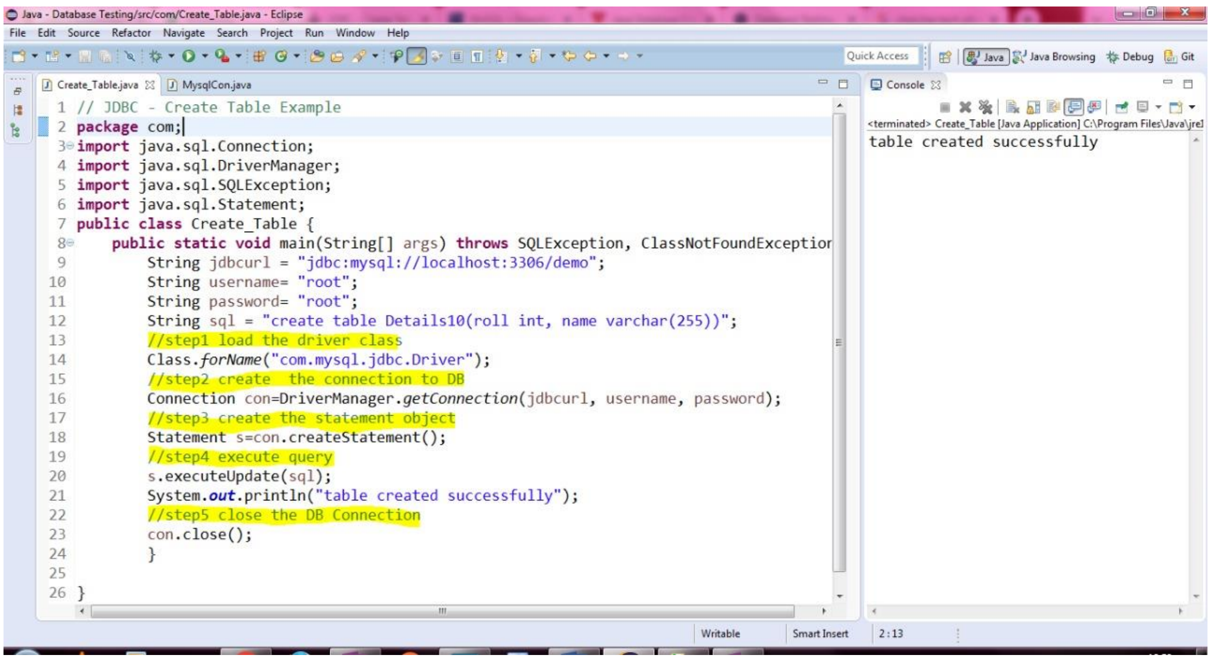

Java - Database Testing/src/com/Create_database.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Java Browsing
Debug
Git
Create_database.java
Console
<terminated> Create_database [Java Applica
Writable
Smart Insert
1 : 34

```
// JDBC - Create Database Example
package com;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
public class Create_database {
    public static void main(String[] args) throws SQLException, ClassNotFoundException {
        Class.forName("com.mysql.jdbc.Driver");

        String jdbcurl = "jdbc:mysql://localhost:3306/";
        String username= "root";
        String password= "root";
        String sql = "Create database fusion_june2022";

        Connection con=DriverManager.getConnection(jdbcurl, username, password);

        Statement s=con.createStatement();

        s.executeUpdate(sql);
        System.out.println("Database created successfully");
        con.close();

    }

}
```

```
Database created succe
```

JDBC Create table Example

```java
// JDBC - Create Table Example
package com;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
public class Create_Table {
        public static void main(String[] args) throws SQLException, ClassNotFoundException {
                String jdbcurl = "jdbc:mysql://localhost:3306/demo";
                String username= "root";
                String password= "root";
                String sql = "create table Details10(roll int, name varchar(255))";
                //step1 load the driver class
                Class.forName("com.mysql.jdbc.Driver");
        //step2 create  the connection to DB
                Connection con=DriverManager.getConnection(jdbcurl, username, password);
                //step3 create the statement object
                Statement s=con.createStatement();
                //step4 execute query
                s.executeUpdate(sql);
                System.out.println("table created successfully");
                //step5 close the DB Connection
                con.close();
                }

}
```

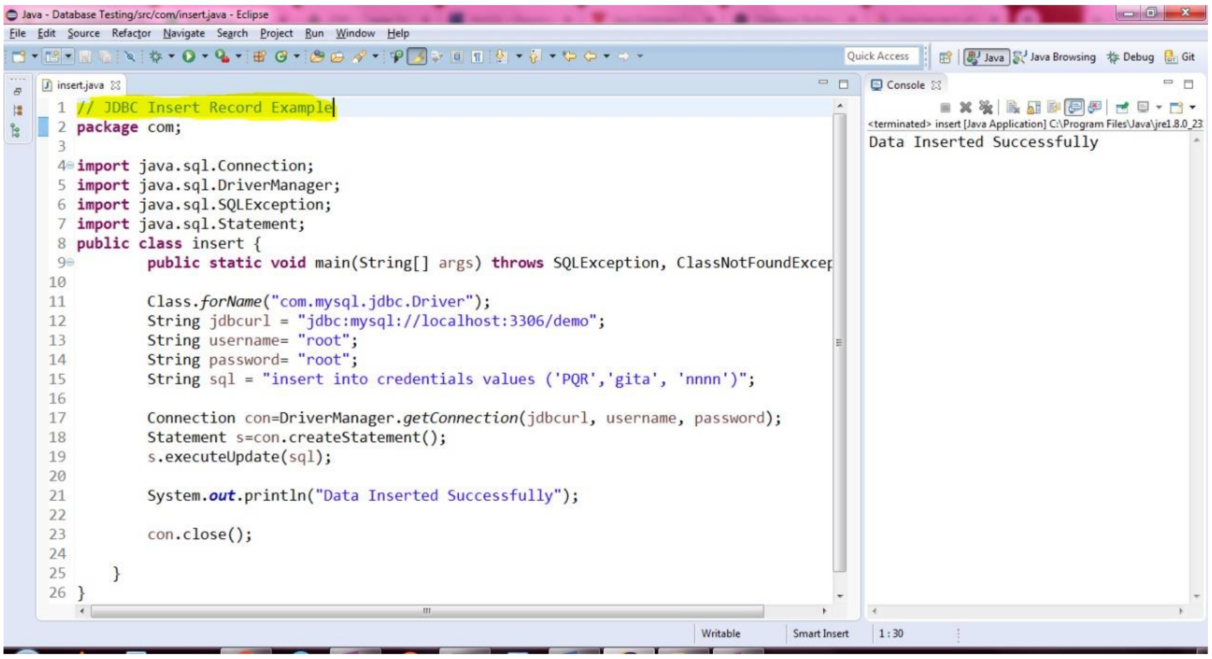

Java - Database Testing/src/com/Create_Table.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Java Browsing
Debug
Git
Create_Table.java
MysqlCon.java
Console
Writable
Smart Insert
2:13

```java
// JDBC - Create Table Example
package com;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
public class Create_Table {
    public static void main(String[] args) throws SQLException, ClassNotFoundExceptio
        String jdbcurl = "jdbc:mysql://localhost:3306/demo";
        String username= "root";
        String password= "root";
        String sql = "create table Details10(roll int, name varchar(255))";
        //step1 load the driver class
        Class.forName("com.mysql.jdbc.Driver");
        //step2 create  the connection to DB
        Connection con=DriverManager.getConnection(jdbcurl, username, password);
        //step3 create the statement object
        Statement s=con.createStatement();
        //step4 execute query
        s.executeUpdate(sql);
        System.out.println("table created successfully");
        //step5 close the DB Connection
        con.close();
        }

}
```

```
<terminated> Create_Table [Java Application] C:\Program Files\Java\jre
table created successfully
```

JDBC Insert Record Example

```java
// JDBC Insert Record Example
package com;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
public class insert {
                public static void main(String[] args) throws SQLException, ClassNotFoundException {

                Class.forName("com.mysql.jdbc.Driver");
                String jdbcurl = "jdbc:mysql://localhost:3306/demo";
                String username= "root";
                String password= "root";
                String sql = "insert into credentials values ('PQR','gita', 'nnnn')";

                Connection con=DriverManager.getConnection(jdbcurl, username, password);
                Statement s=con.createStatement();
                s.executeUpdate(sql);

            System.out.println("Data Inserted Successfully");

            con.close();

        }
}
```

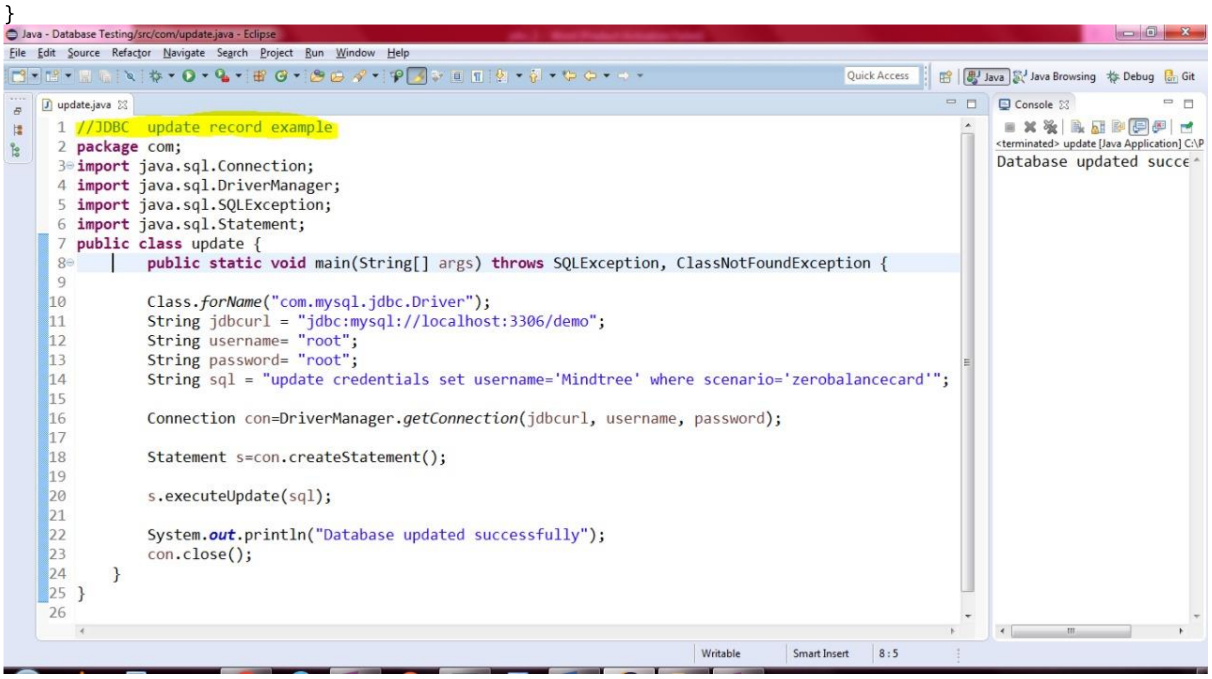

Java - Database Testing/src/com/insert.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Java Browsing
Debug
Git
insert.java
Console
Writable
Smart Insert
1 : 30

```java
// JDBC Insert Record Example
package com;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
public class insert {
        public static void main(String[] args) throws SQLException, ClassNotFoundExcep

        Class.forName("com.mysql.jdbc.Driver");
        String jdbcurl = "jdbc:mysql://localhost:3306/demo";
        String username= "root";
        String password= "root";
        String sql = "insert into credentials values ('PQR','gita', 'nnnn')";

        Connection con=DriverManager.getConnection(jdbcurl, username, password);
        Statement s=con.createStatement();
        s.executeUpdate(sql);

        System.out.println("Data Inserted Successfully");

        con.close();

    }
}
```

```
<terminated> insert [Java Application] C:\Program Files\Java\jre1.8.0_23
Data Inserted Successfully
```

JDBC Update Record Example

```
//JDBC  update record example
package com;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
public class update {
                public static void main(String[] args) throws SQLException, ClassNotFoundException {

                Class.forName("com.mysql.jdbc.Driver");
                String jdbcurl = "jdbc:mysql://localhost:3306/demo";
                String username= "root";
                String password= "root";
                String sql = "update credentials set username='Mindtree' where 
scenario='zerobalancecard'";

                Connection con=DriverManager.getConnection(jdbcurl, username, password);

                Statement s=con.createStatement();

                s.executeUpdate(sql);

                System.out.println("Database updated successfully");
                con.close();
        }
}
```


Java - Database Testing/src/com/update.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Java Browsing
Debug
Git
update.java
Console
Writable
Smart Insert
8 : 5

```
//JDBC  update record example
package com;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
public class update {
        public static void main(String[] args) throws SQLException, ClassNotFoundException {

        Class.forName("com.mysql.jdbc.Driver");
        String jdbcurl = "jdbc:mysql://localhost:3306/demo";
        String username= "root";
        String password= "root";
        String sql = "update credentials set username='Mindtree' where scenario='zerobalancecard'";

        Connection con=DriverManager.getConnection(jdbcurl, username, password);

        Statement s=con.createStatement();

        s.executeUpdate(sql);

        System.out.println("Database updated successfully");
        con.close();
    }
}
```

```
<terminated> update [Java Application] C:\P
Database updated succe
```

What is PreparedStatement in JDBC?

The **PreparedStatement** interface extends the Statement interface it represents a precompiled SQL statement which can be executed multiple times. This accepts parameterized SQL quires and you can pass 0 or more parameters to this query.

Initially this statement uses place holders **"?"** instead of parameters, later on you can pass arguments to these dynamically using the setXXX() methods of the **PreparedStatement** interface.

Difference between Statement and PreparedStatement

Statement:

It is used for accessing your database. Statement interface cannot accept parameters and useful when you are using static SQL statements at runtime. If you want to run SQL query only once then this interface is preferred over PreparedStatement.

Example –

```
//Creating The Statement Object

Statement s = con.createStatement();

//Executing The Statement

s.executeUpdate("CREATE TABLE STUDENT(ID NUMBER NOT NULL, NAME VARCHAR)");
```

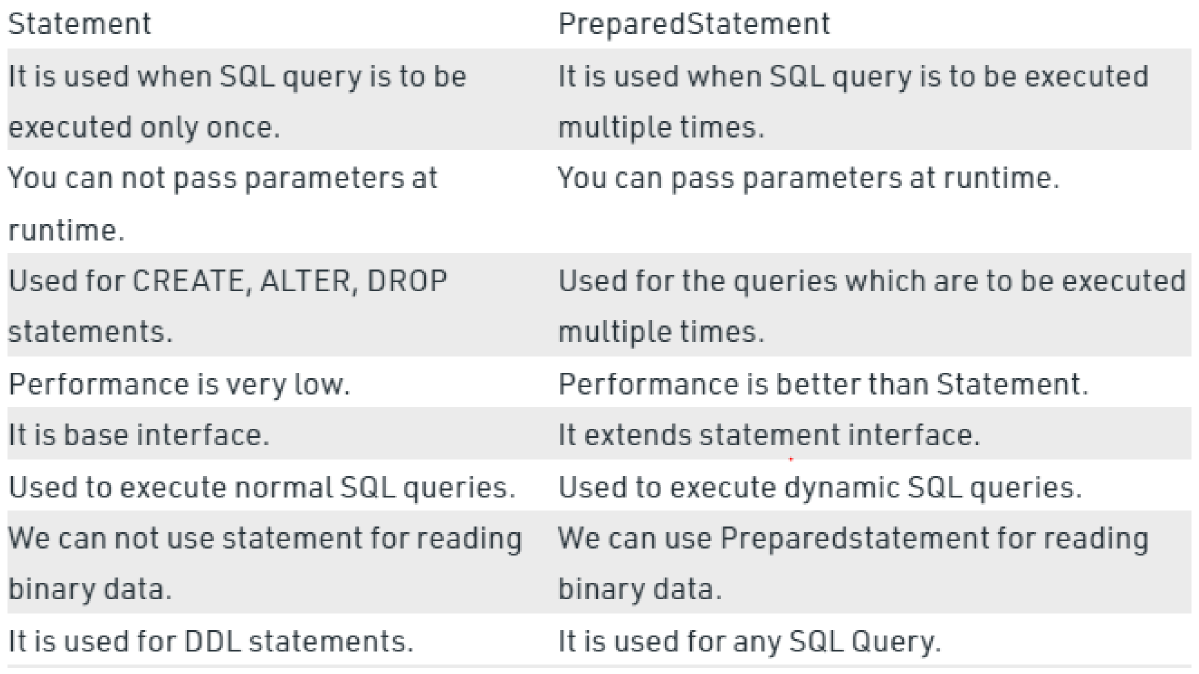
PreparedStatement:

It is used when you want to use SQL statements many times. The PreparedStatement interface accepts input parameters at runtime.

Example –

```
//Creating the PreparedStatement object

PreparedStatement s = con.prepareStatement("update STUDENT set NAME = ? where ID = ?");

//Setting values to place holders

//Assigns "RAM" to first place holder

s.setString(1, "RAM");

//Assigns "512" to second place holder

s.setInt(2, 512);

 //Executing PreparedStatement

s.executeUpdate();
```

Difference between Statement and PreparedStatement :



Statement	PreparedStatement
It is used when SQL query is to be executed only once.	It is used when SQL query is to be executed multiple times.
You can not pass parameters at runtime.	You can pass parameters at runtime.
Used for CREATE, ALTER, DROP statements.	Used for the queries which are to be executed multiple times.
Performance is very low.	Performance is better than Statement.
It is base interface.	It extends statement interface.
Used to execute normal SQL queries.	Used to execute dynamic SQL queries.
We can not use statement for reading binary data.	We can use Preparedstatement for reading binary data.
It is used for DDL statements.	It is used for any SQL Query.

Methods of PreparedStatement interface

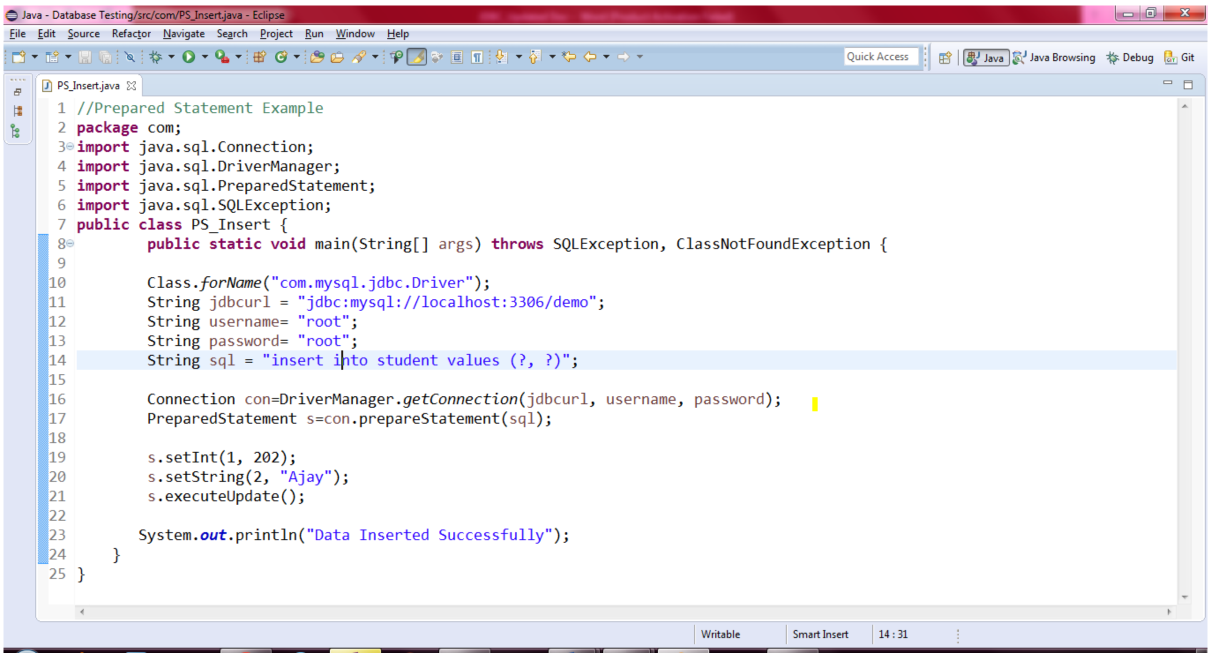
The important methods of PreparedStatement interface are given below:

Method	**Description**
public void setInt(int paramIndex, int value)	sets the integer value to the given parameter index.
public void setString(int paramIndex, String value)	sets the String value to the given parameter index.
public void setFloat(int paramIndex, float value)	sets the float value to the given parameter index.
public void setDouble(int paramIndex, double value)	sets the double value to the given parameter index.
public int executeUpdate()	executes the query. It is used for create, drop, insert, update, delete etc.
public ResultSet executeQuery()	executes the select query. It returns an instance of ResultSet.

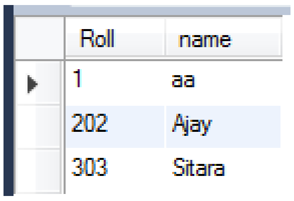
Example

```
//Prepared Statement Example
package com;
```

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
public class PS_Insert {
                public static void main(String[] args) throws SQLException, ClassNotFoundException {

                Class.forName("com.mysql.jdbc.Driver");
                String jdbcurl = "jdbc:mysql://localhost:3306/demo";
                String username= "root";
                String password= "root";
                String sql = "insert into student values (?, ?)";

                Connection con=DriverManager.getConnection(jdbcurl, username, password);
                PreparedStatement s=con.prepareStatement(sql);

                s.setInt(1, 202);
        s.setString(2, "Ajay");
        s.executeUpdate();

       System.out.println("Data Inserted Successfully");
        }
}
```


Java - Database Testing/src/com/PS_Insert.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Java Browsing
Debug
Git
PS_Insert.java
Writable
Smart Insert
14 : 31

```java
//Prepared Statement Example
package com;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
public class PS_Insert {
        public static void main(String[] args) throws SQLException, ClassNotFoundException {

        Class.forName("com.mysql.jdbc.Driver");
        String jdbcurl = "jdbc:mysql://localhost:3306/demo";
        String username= "root";
        String password= "root";
        String sql = "insert into student values (?, ?)";

        Connection con=DriverManager.getConnection(jdbcurl, username, password);
        PreparedStatement s=con.prepareStatement(sql);

        s.setInt(1, 202);
        s.setString(2, "Ajay");
        s.executeUpdate();

       System.out.println("Data Inserted Successfully");
    }
}
```

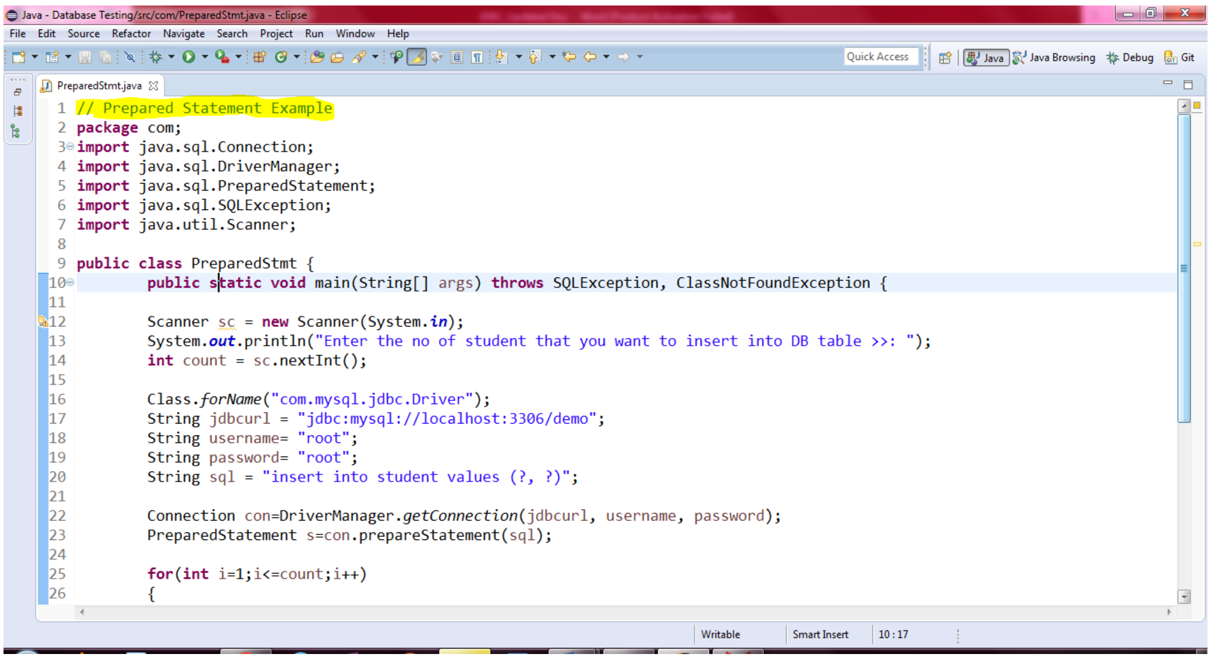


Roll	name
1	aa
202	Ajay
303	Sitara

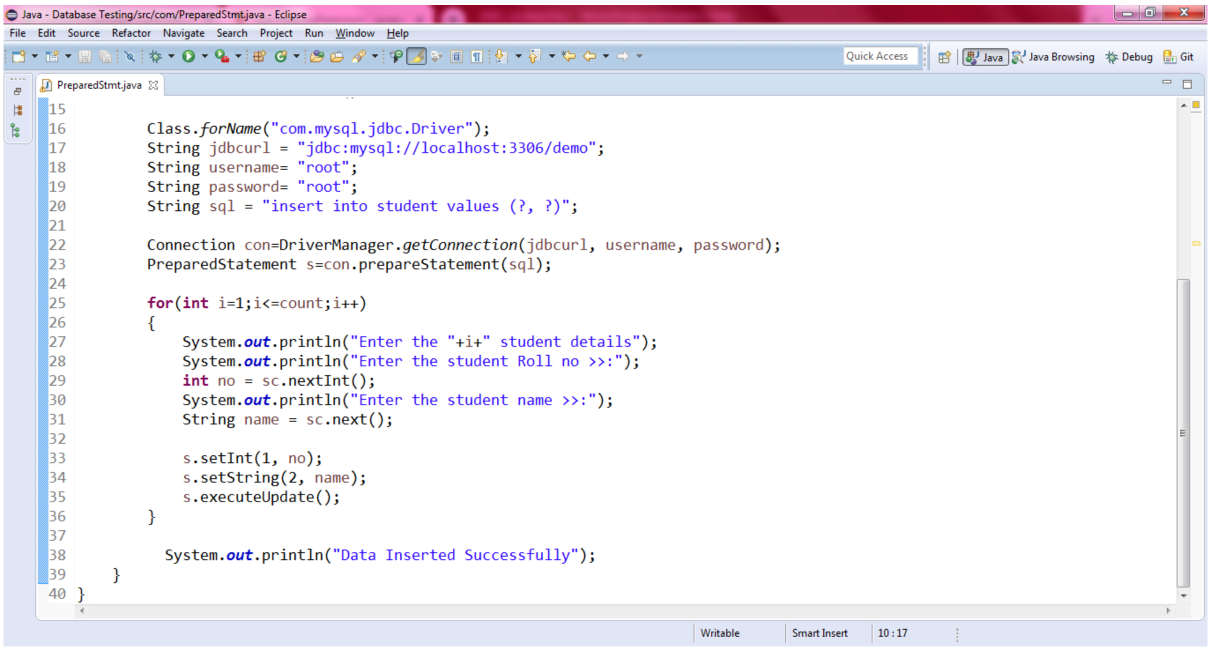
Java Program for PreparedStatement Interface

```java
// Prepared Statement Example
package com;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.Scanner;
```

```java
public class PreparedStmt {
                public static void main(String[] args) throws SQLException, ClassNotFoundException {

                Scanner sc = new Scanner(System.in);
                System.out.println("Enter the no of student that you want to insert into DB table >>:
");
                int count = sc.nextInt();

                Class.forName("com.mysql.jdbc.Driver");
                String jdbcurl = "jdbc:mysql://localhost:3306/demo";
                String username= "root";
                String password= "root";
                String sql = "insert into student values (?, ?)";

                Connection con=DriverManager.getConnection(jdbcurl, username, password);
                PreparedStatement s=con.prepareStatement(sql);

        for(int i=1;i<=count;i++)
        {
        System.out.println("Enter the "+i+" student details");
        System.out.println("Enter the student Roll no >>:");
        int no = sc.nextInt();
        System.out.println("Enter the student name >>:");
        String name = sc.next();

        s.setInt(1, no);
        s.setString(2, name);
        s.executeUpdate();
        }

          System.out.println("Data Inserted Successfully");
        }
}
```

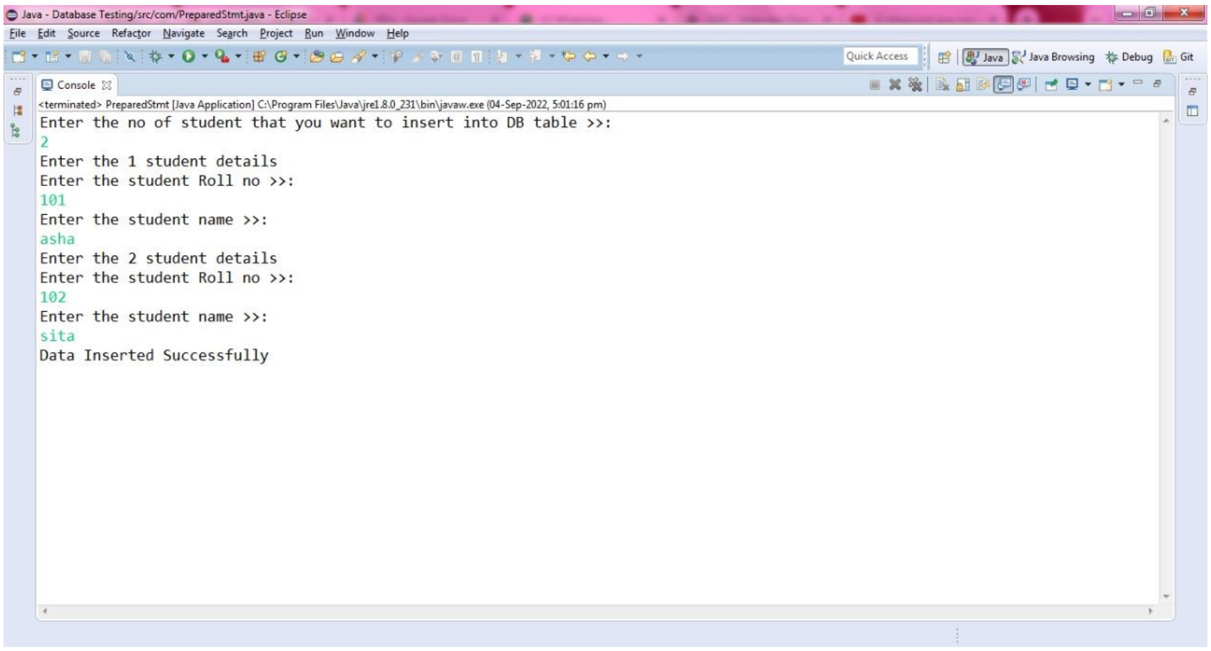

Java - Database Testing/src/com/PreparedStmt.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Java Browsing
Debug
Git
PreparedStmt.java
Writable
Smart Insert
10 : 17

```java
// Prepared Statement Example
package com;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.Scanner;

public class PreparedStmt {
        public static void main(String[] args) throws SQLException, ClassNotFoundException {

        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the no of student that you want to insert into DB table >>: ");
        int count = sc.nextInt();

        Class.forName("com.mysql.jdbc.Driver");
        String jdbcurl = "jdbc:mysql://localhost:3306/demo";
        String username= "root";
        String password= "root";
        String sql = "insert into student values (?, ?)";

        Connection con=DriverManager.getConnection(jdbcurl, username, password);
        PreparedStatement s=con.prepareStatement(sql);

        for(int i=1;i<=count;i++)
        {
```



```
        Class.forName("com.mysql.jdbc.Driver");
        String jdbcurl = "jdbc:mysql://localhost:3306/demo";
        String username= "root";
        String password= "root";
        String sql = "insert into student values (?, ?)";

        Connection con=DriverManager.getConnection(jdbcurl, username, password);
        PreparedStatement s=con.prepareStatement(sql);

        for(int i=1;i<=count;i++)
        {
            System.out.println("Enter the "+i+" student details");
            System.out.println("Enter the student Roll no >>:");
            int no = sc.nextInt();
            System.out.println("Enter the student name >>:");
            String name = sc.next();

            s.setInt(1, no);
            s.setString(2, name);
            s.executeUpdate();
        }

          System.out.println("Data Inserted Successfully");
    }
}
```

Output:



```
<terminated> PreparedStmt [Java Application] C:\Program Files\Java\jre1.8.0_231\bin\javaw.exe (04-Sep-2022, 5:01:16 pm)
Enter the no of student that you want to insert into DB table >>:
2
Enter the 1 student details
Enter the student Roll no >>:
101
Enter the student name >>:
asha
Enter the 2 student details
Enter the student Roll no >>:
102
Enter the student name >>:
sita
Data Inserted Successfully
```

Java Selenium MySQL integration

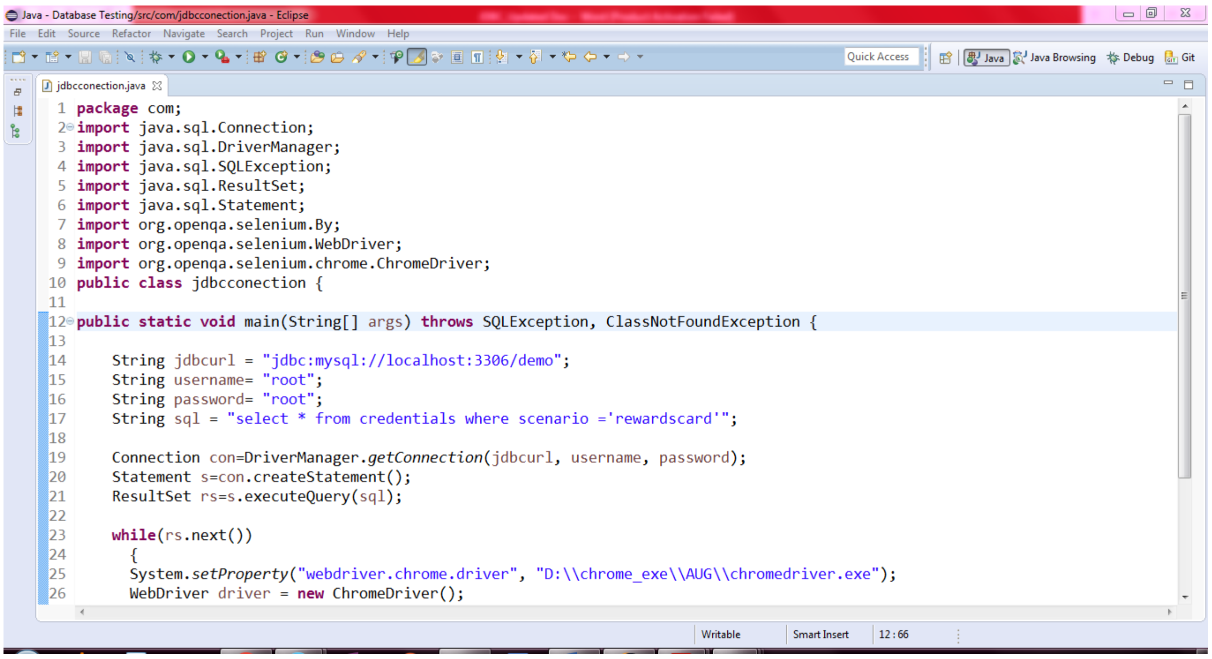
Need 2 things

1) **Selenium Jar**
2) **ChromeDriver.exe file**

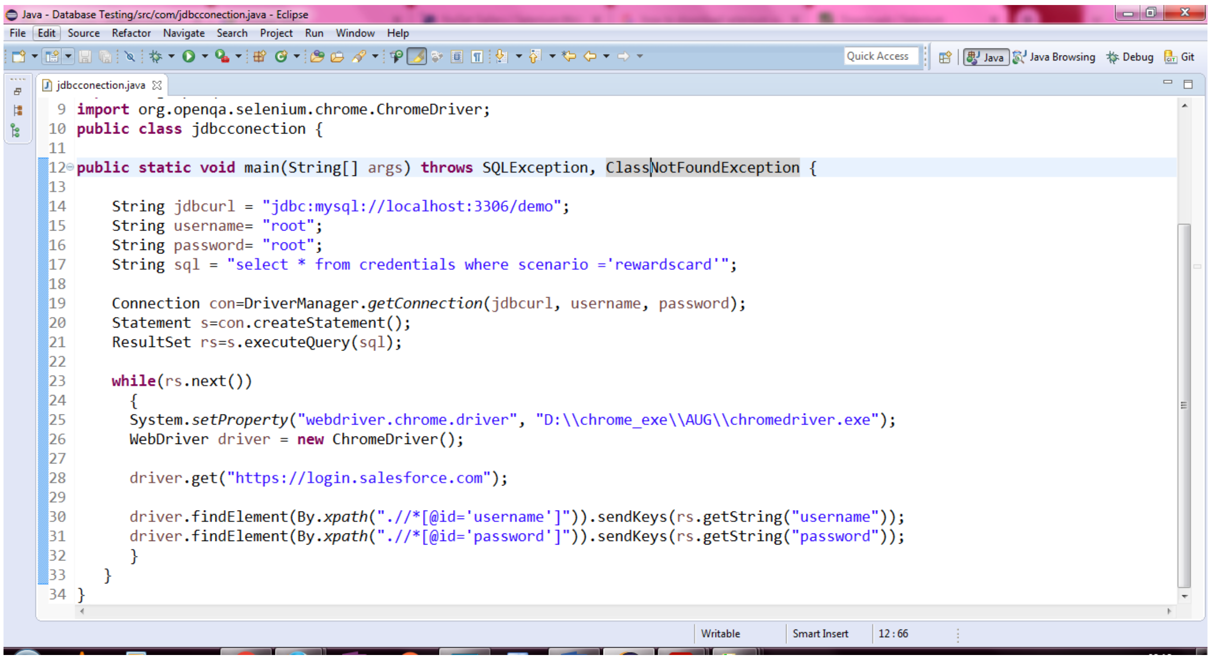
Note – Do this program when your Automation Testing(Selenium) is done

```
package com;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
```

```java
import java.sql.ResultSet;
import java.sql.Statement;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
public class jdbcconection {

public static void main(String[] args) throws SQLException, ClassNotFoundException {

        String jdbcurl = "jdbc:mysql://localhost:3306/demo";
        String username= "root";
        String password= "root";
        String sql = "select * from credentials where scenario ='rewardscard'";

        Connection con=DriverManager.getConnection(jdbcurl, username, password);
    Statement s=con.createStatement();
    ResultSet rs=s.executeQuery(sql);

    while(rs.next())
      {
      System.setProperty("webdriver.chrome.driver", "D:\\chrome_exe\\AUG\\chromedriver.exe");
      WebDriver driver = new ChromeDriver();

      driver.get("https://login.salesforce.com");

      driver.findElement(By.xpath(".//*[@id='username']")).sendKeys(rs.getString("username"));
      driver.findElement(By.xpath(".//*[@id='password']")).sendKeys(rs.getString("password"));
      }
   }
}
```

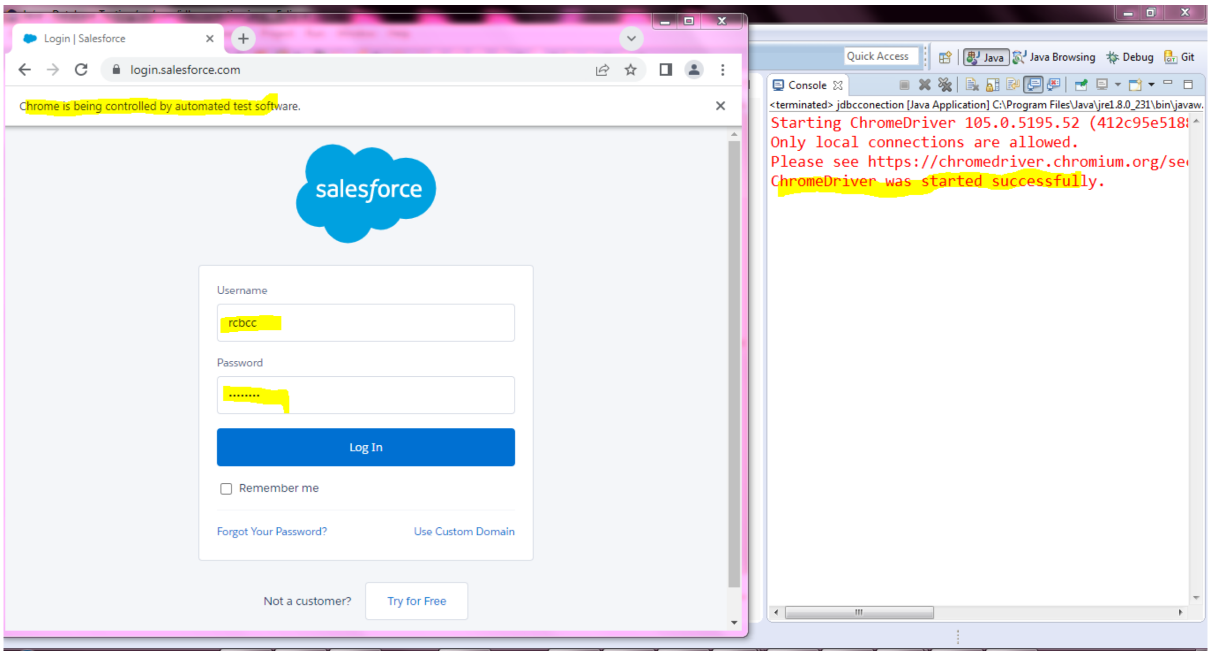

Java - Database Testing/src/com/jdbcconection.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Java Browsing
Debug
Git
jdbcconection.java

```java
package com;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.ResultSet;
import java.sql.Statement;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
public class jdbcconection {

public static void main(String[] args) throws SQLException, ClassNotFoundException {

    String jdbcurl = "jdbc:mysql://localhost:3306/demo";
    String username= "root";
    String password= "root";
    String sql = "select * from credentials where scenario ='rewardscard'";

    Connection con=DriverManager.getConnection(jdbcurl, username, password);
    Statement s=con.createStatement();
    ResultSet rs=s.executeQuery(sql);

    while(rs.next())
      {
      System.setProperty("webdriver.chrome.driver", "D:\\chrome_exe\\AUG\\chromedriver.exe");
      WebDriver driver = new ChromeDriver();
```



```java
import org.openqa.selenium.chrome.ChromeDriver;
public class jdbcconection {

public static void main(String[] args) throws SQLException, ClassNotFoundException {

    String jdbcurl = "jdbc:mysql://localhost:3306/demo";
    String username= "root";
    String password= "root";
    String sql = "select * from credentials where scenario ='rewardscard'";

    Connection con=DriverManager.getConnection(jdbcurl, username, password);
    Statement s=con.createStatement();
    ResultSet rs=s.executeQuery(sql);

    while(rs.next())
      {
      System.setProperty("webdriver.chrome.driver", "D:\\chrome_exe\\AUG\\chromedriver.exe");
      WebDriver driver = new ChromeDriver();

      driver.get("https://login.salesforce.com");

      driver.findElement(By.xpath(".//*[@id='username']")).sendKeys(rs.getString("username"));
      driver.findElement(By.xpath(".//*[@id='password']")).sendKeys(rs.getString("password"));
      }
    }
}
```

Output



Chrome is being controlled by automated test software.

```
<terminated> jdbcconection [Java Application] C:\Program Files\Java\jre1.8.0_231\bin\javaw.
Starting ChromeDriver 105.0.5195.52 (412c95e518
Only local connections are allowed.
Please see https://chromedriver.chromium.org/se
ChromeDriver was started successfully.
```