Developing a JDBC Application for Employee Information Management

In March 2022, I developed a JDBC (Java Database Connectivity) server and application using Apache NetBeans to manage basic employee information. This project involved setting up a MySQL database named emsdb with a table called ems, designed to store details such as Employee ID, Name, City, Position, and Salary.

Overview

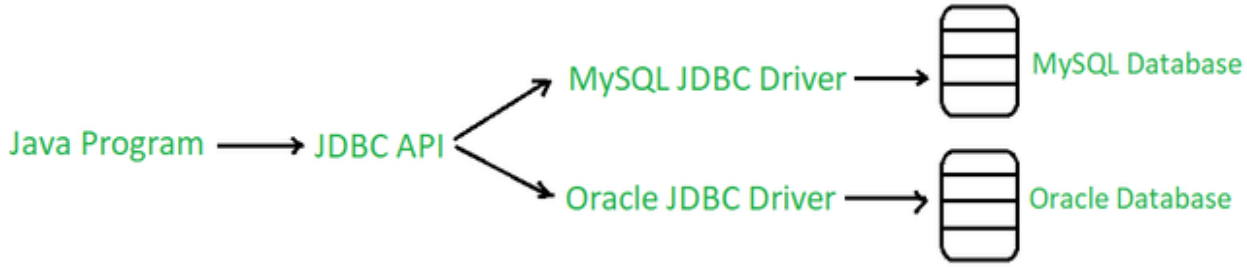
Consider a scenario where the CEO and other authorities need to review employee records for various purposes, such as searching for a particular employee's details. Manually going through records is tedious and time-consuming. Hence, developing software that allows users to insert, update, search, or delete records without manually handling documents is beneficial.

In this article, we will explore how to create an application using Java Swing to perform operations like creating, retrieving, and deleting records in the database using JDBC.

Prerequisites

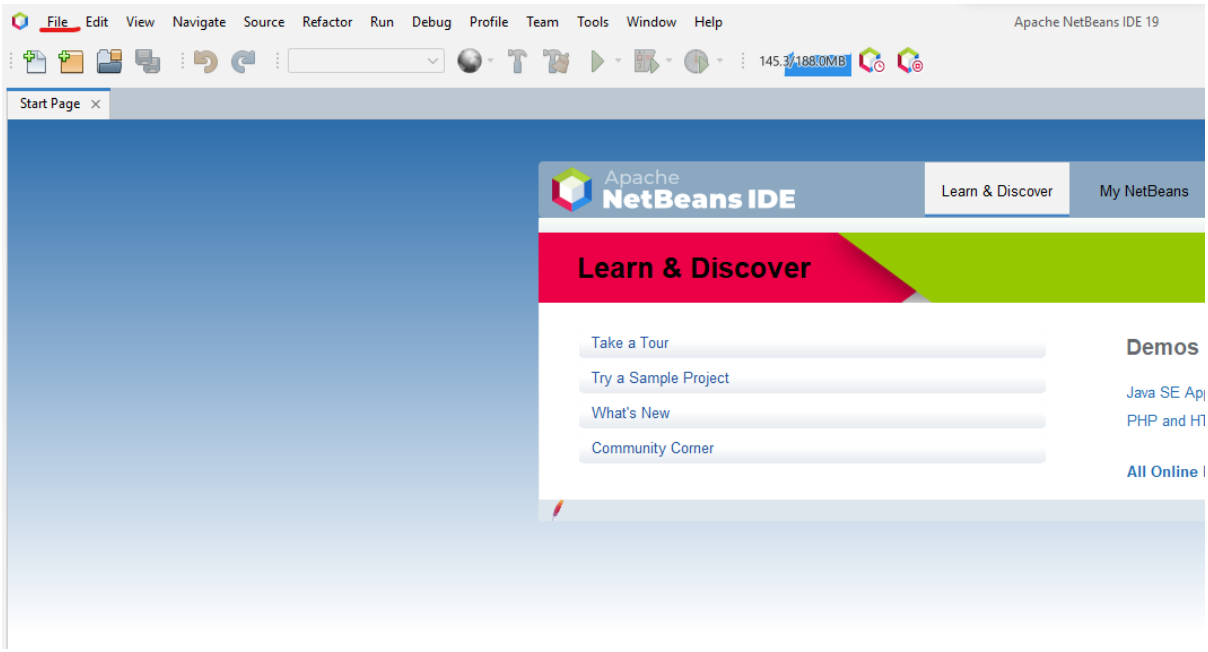
Before writing the code, a few things need to be kept in mind:

1. **JDBC API**: Java Database Connectivity Application Program Interface is a set of interfaces and classes that enable Java programs to access and manipulate databases. It acts as a communication layer between the application and the database.
2. **JDBC Driver**: It enables a Java application to interact with the database. Different JDBC drivers are required for different databases.

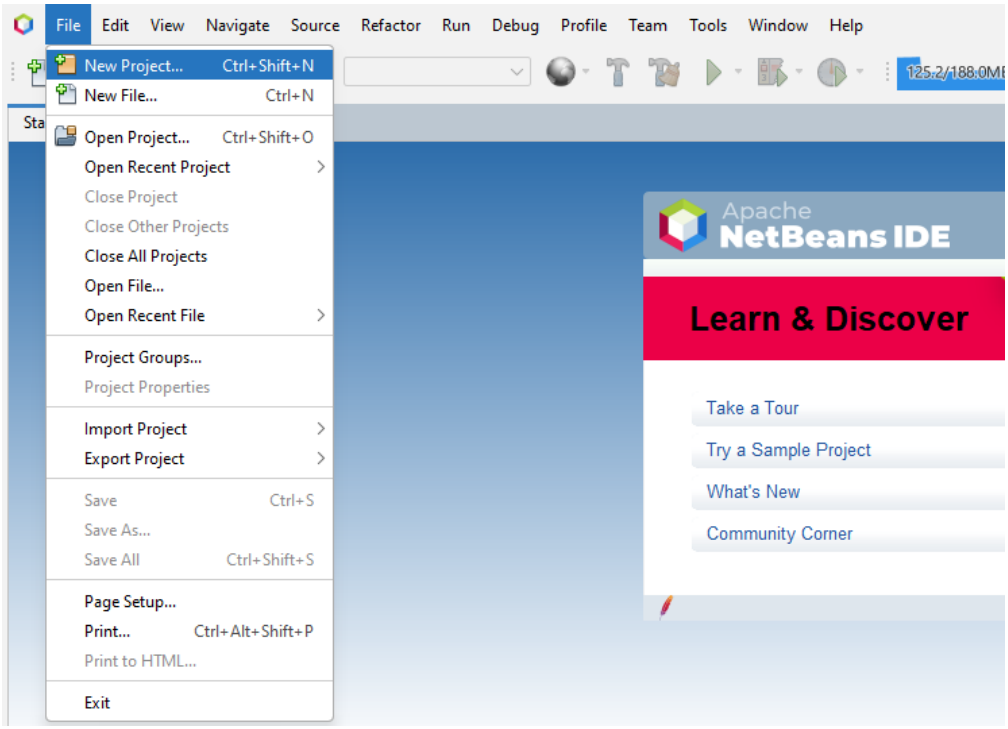


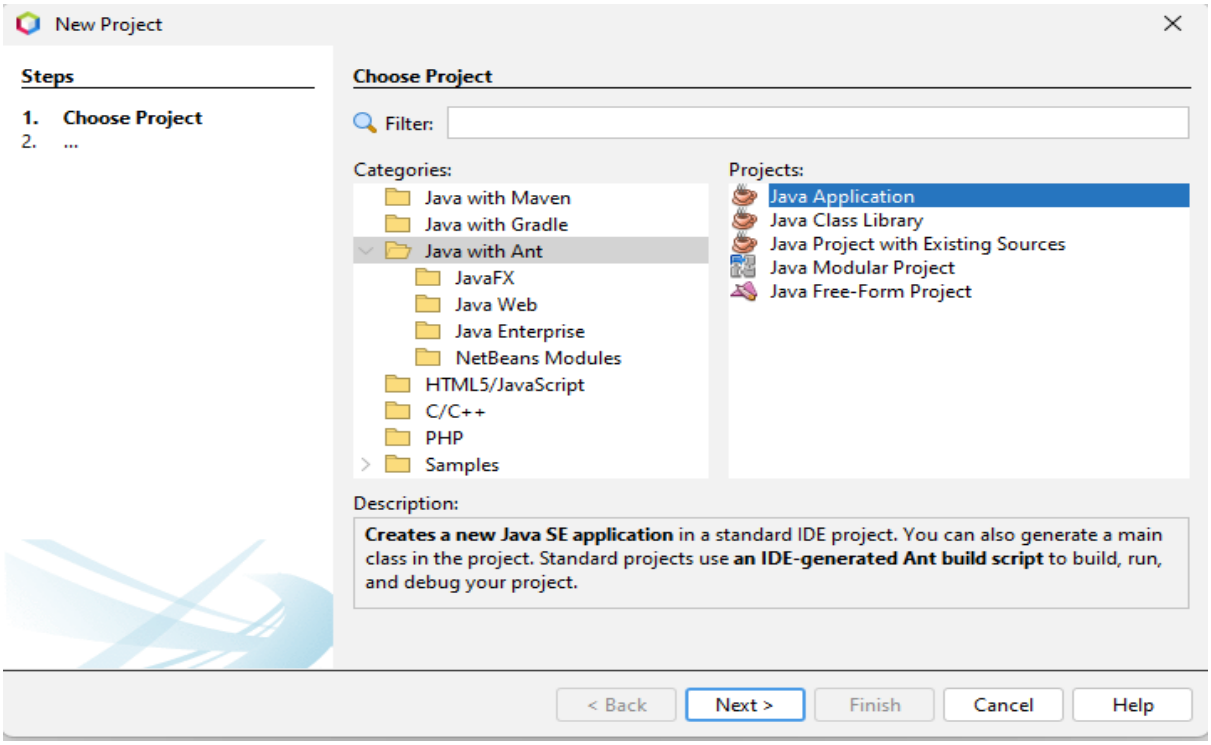
Steps to create the application:

1. First, open Netbeans and click on the File option from the menu bar.

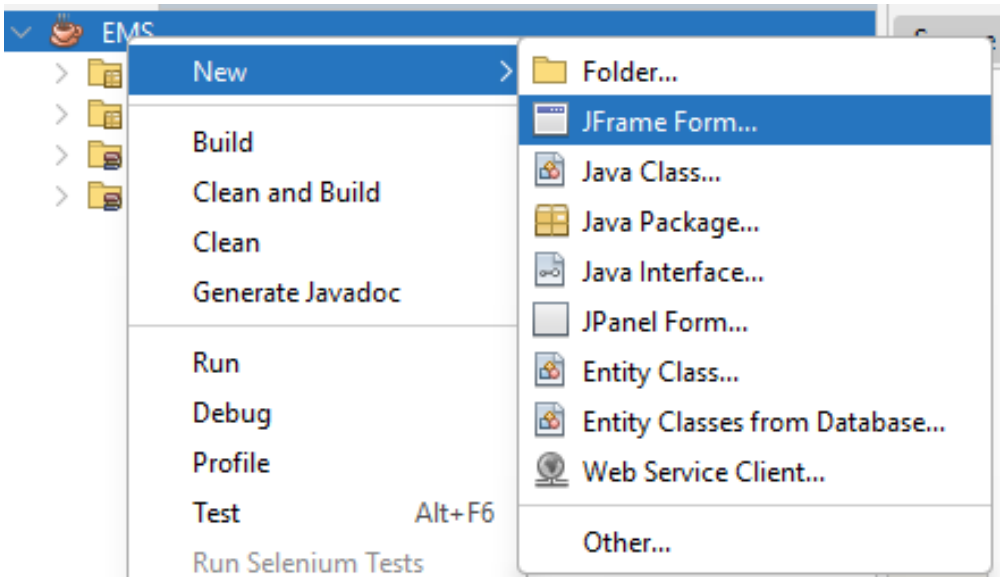


2. Now create a new Java application by clicking on New Project -> Java -> Java Application and give a suitable project name **for example** “ems” and click finish.



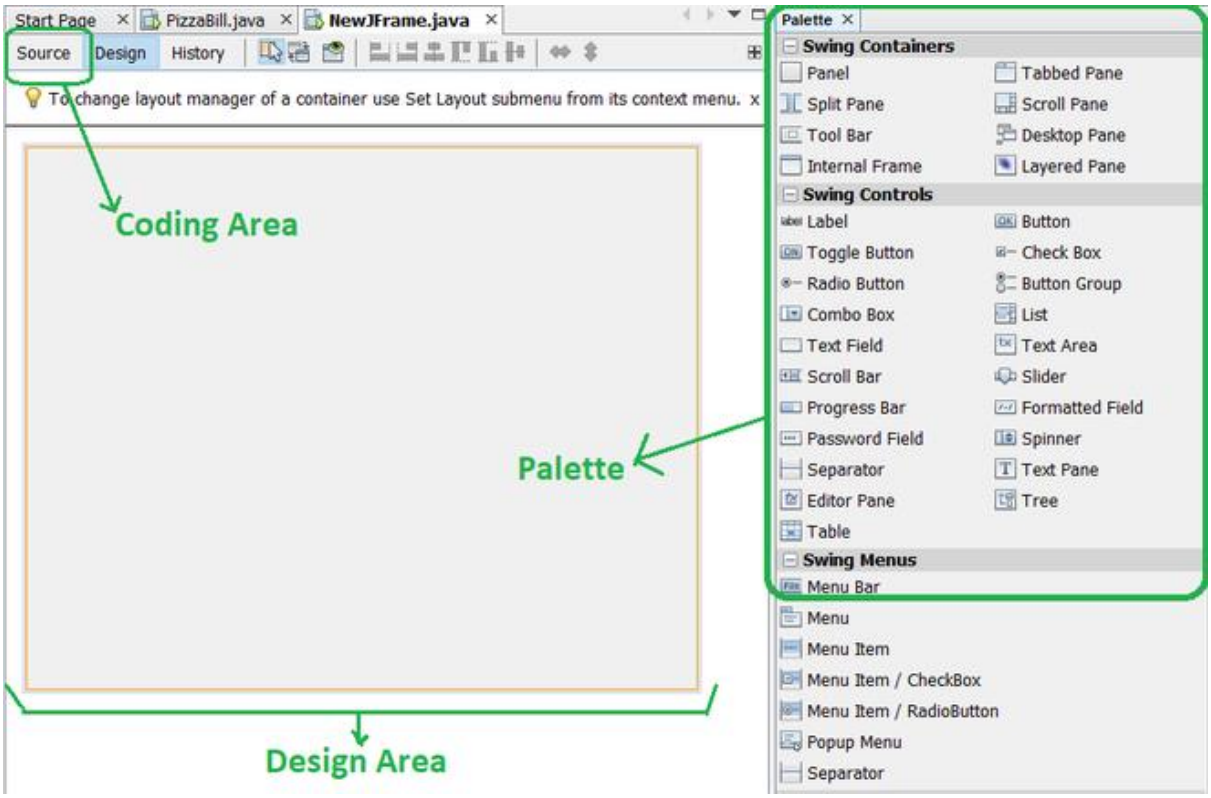


3. Now create a new file by going to the File option again on the menu bar, then New File -> Swing GUI Forms -> JFrame Form, and give a suitable file name **for example** “RegistrationForm” click finish.

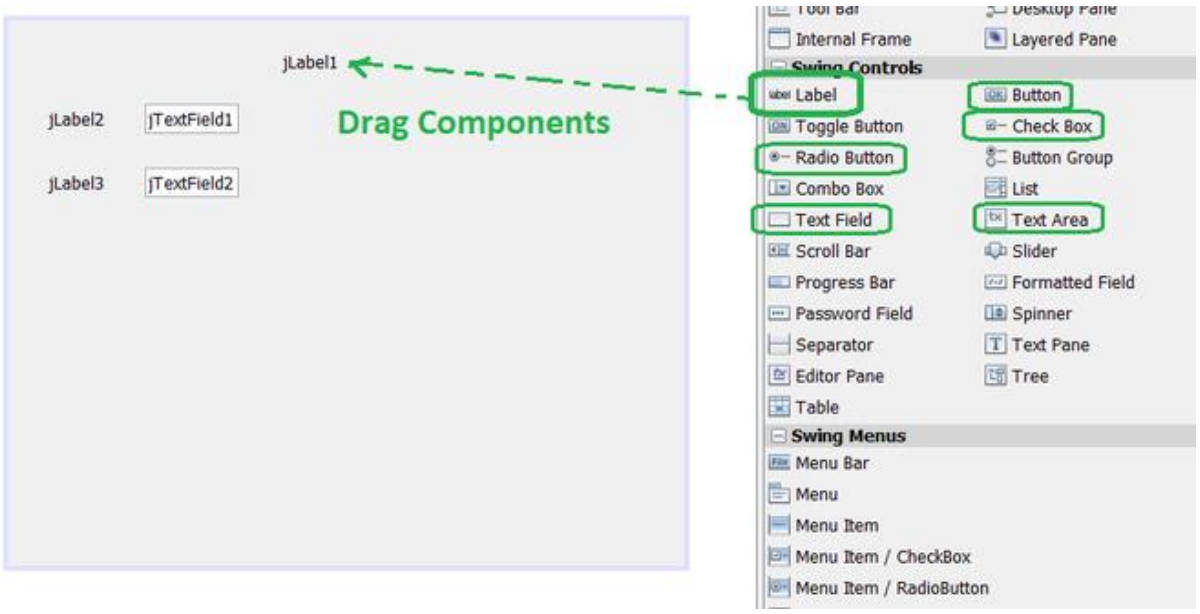


4. After successful file creation, we will now be presented with the following screen. The 3 important parts of this window are:

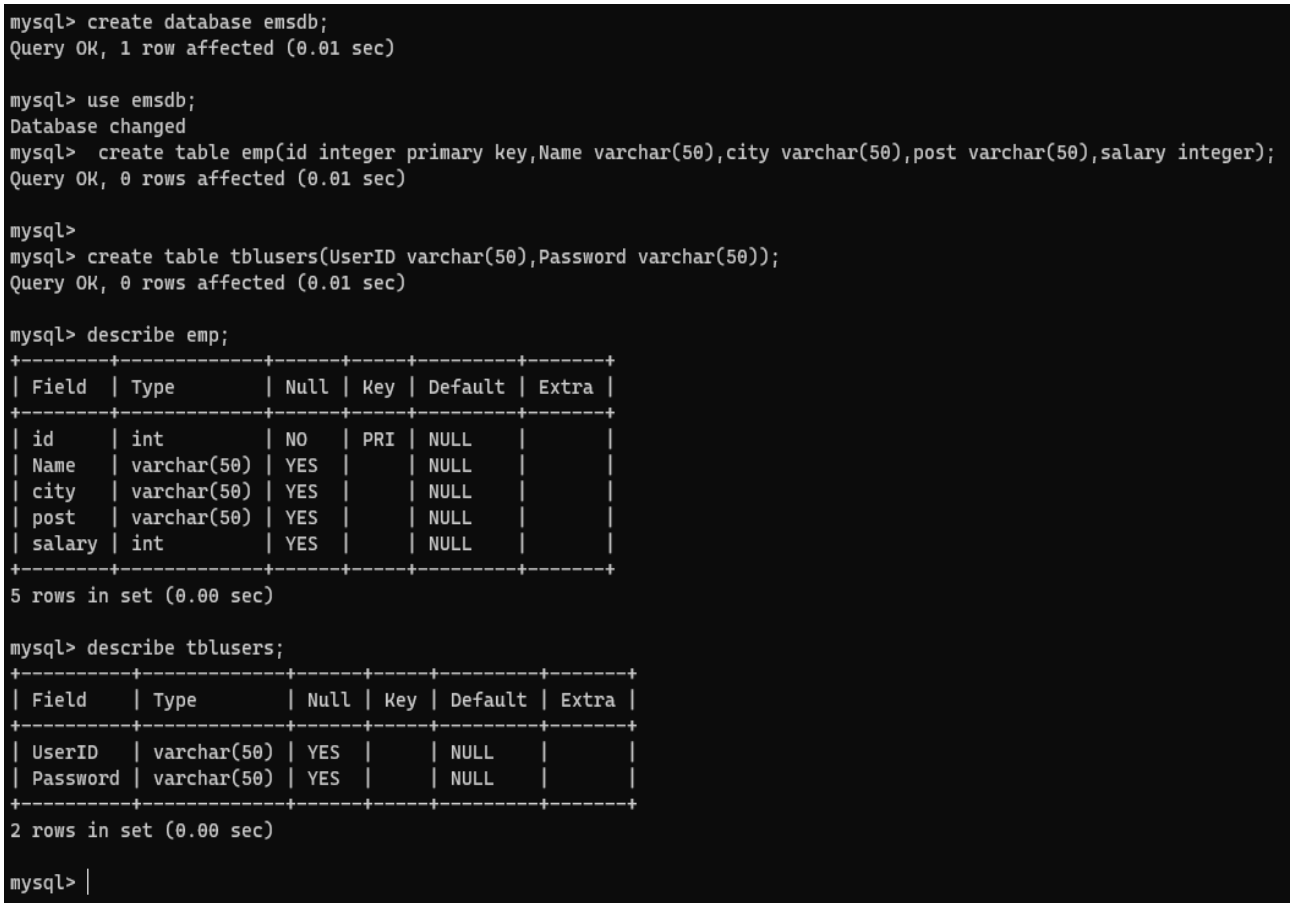
- **Design:** This is the area where we will create the design/template of our application.
- **Source:** This is where the logic code of the program is written.
- **Palette:** This component contains all the widgets which we need to drag and drop on the design area



5. Now from the palette situated at the right-hand side of the window, start dragging the toolkit widgets.



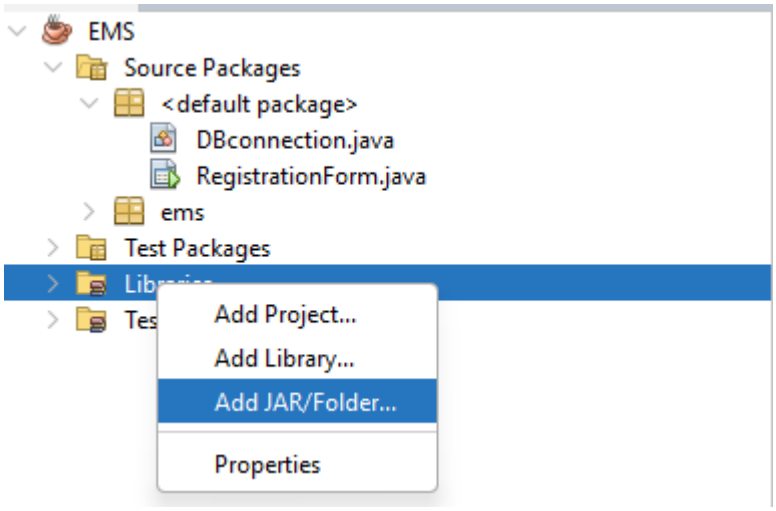
6. Now let us create the database named “emsdb” to store the data. Open MySQL command client, enter password, and type in the following commands to create a new database, new table, and defining the attributes.



```
mysql> create database emsdb;
Query OK, 1 row affected (0.01 sec)

mysql> use emsdb;
Database changed
mysql>  create table emp(id integer primary key,Name varchar(50),city varchar(50),post varchar(50),salary integer);
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> create table tblusers(UserID varchar(50),Password varchar(50));
Query OK, 0 rows affected (0.01 sec)

mysql> describe emp;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| id     | int         | NO   | PRI | NULL    |       |
| Name   | varchar(50) | YES  |     | NULL    |       |
| city   | varchar(50) | YES  |     | NULL    |       |
| post   | varchar(50) | YES  |     | NULL    |       |
| salary | int         | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)

mysql> describe tblusers;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| UserID   | varchar(50) | YES  |     | NULL    |       |
| Password | varchar(50) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql>
```

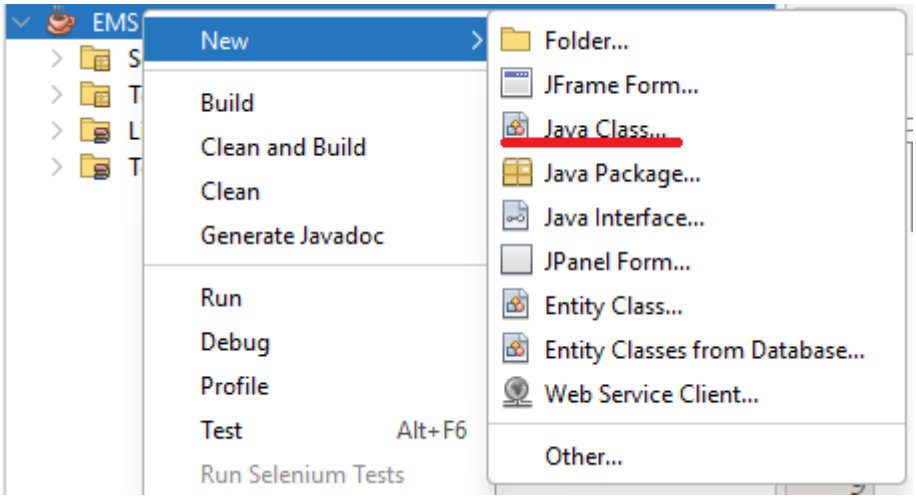
Going back to NetBeans, we need to follow these steps for database connectivity:

7. We need to import libraries that are needed to set up a connection with the database and retrieve data which is done by – *DriverManager Class*, *Connection Class*, and *Statement Class.* *Now go to* ***Projects*** *toolbar and go to your application's Libraries. Right-click and select* ***Add Jar/Library*** *and browse the Library classpath noted down previously.*

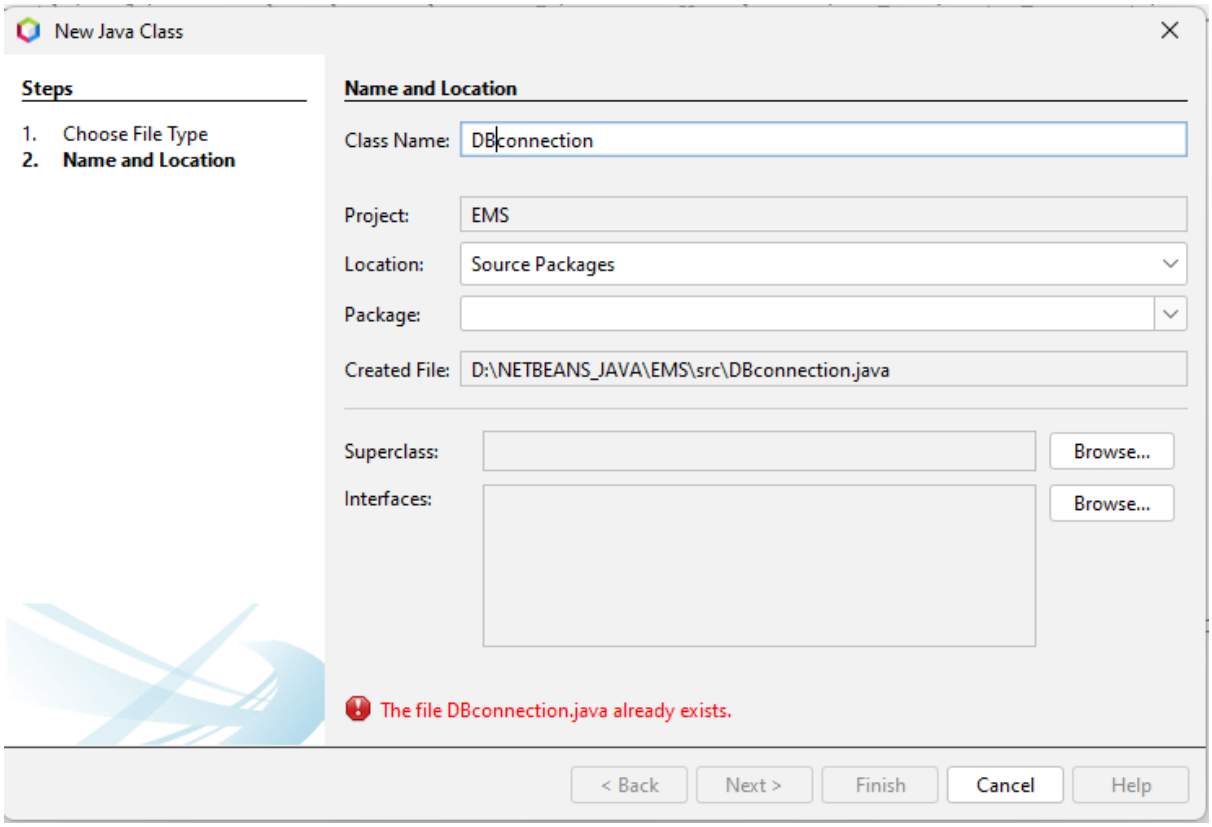


8. Now we have to connect the database to the actual framework so to do that

1- we have to create a java class into our “ems” project



2- then we have to name the java class “DBconnection.java” and have to type the code.

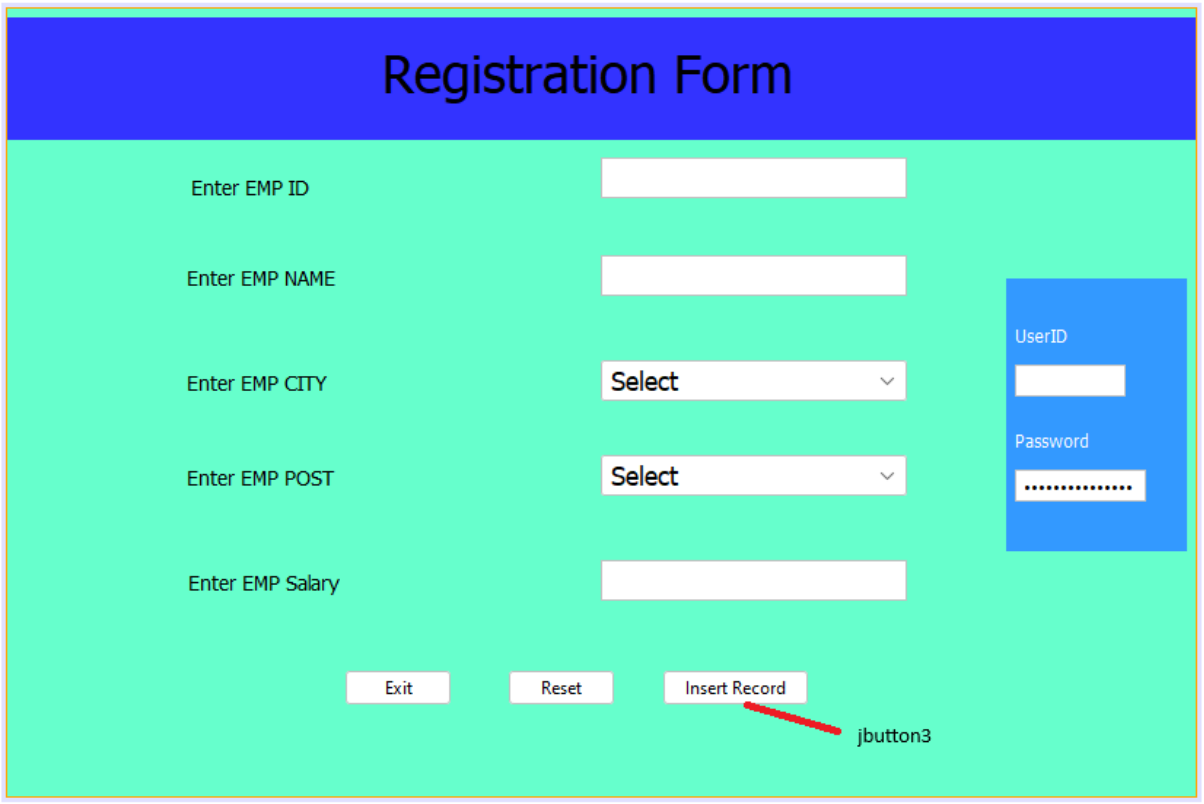


3-Then we can start typing the code in “DBconnection” class

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

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 *
 * @author sagar
 */
public class DBconnection {
    public static Connection MyConnection()
    {
      try
      {
         Class.forName(className: "com.mysql.cj.jdbc.Driver");
         Connection con =DriverManager.getConnection(url:"jdbc:mysql://localhost:3306/emsdb",user: "root",password: "Root");
         return con;

      }
      catch(Exception e)
      {
          System.out.println("Database Error"+e.getMessage());
          return null;
      }

    }

}
```

After that we can come back to “RegistrationForm” and start design the framework.

9.After designing the frame Now we have to type the code, double-click on jButton3 (Insert), you will be directed to the source tab.



```java

import java.sql.Connection;
import java.sql.PreparedStatement;
import javax.swing.JOptionPane;

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 *
 * @author sagar
 */
public class RegistrationForm extends javax.swing.JFrame {

    /**
     * Creates new form RegistrationForm
     */
    Connection con;
    PreparedStatement pstmt;
    public RegistrationForm() {
        con= DBconnection.MyConnection();
        initComponents();
    }


    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    Generated Code

    private void teidActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void combocityActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void tnameActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void tsalActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }
```

Here type in the following code:

```
    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
        // insert record button coding
        String id,name,city,post,uid,pass;
        int sal;
        id=teid.getText();
        name=tname.getText();
        city=(String)combocity.getSelectedItem();
        post=(String)combopost.getSelectedItem();
        sal=Integer.parseInt(s: tsal.getText());
        uid=userid.getText();
        pass=Password.getText();
       try
        {
            String q1="insert into emp values('"+id+"','"+name+"','"+city+"','"+post+"','"+sal+"')";
            pstmt=con.prepareStatement(sql:q1);
            int j=pstmt.executeUpdate();
            if(j>0)
            {
                JOptionPane.showMessageDialog(parentComponent: this,message: "Record Inserted Successfully");
            }
            else
            {
             JOptionPane.showMessageDialog(parentComponent: this,message: "Database Error");
            }

        }
```

```
        catch(Exception e)
        {
            System.out.println("Reg error "+e.getMessage());
        }

    }

    /**
     * @param args the command line arguments
     */
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        Look and feel setting code (optional)

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new RegistrationForm().setVisible(b: true);
            }
        });
    }
```

```
391         // Variables declaration - do not modify
392         private javax.swing.JPasswordField Password;
393         private javax.swing.JComboBox<String> combocity;
394         private javax.swing.JComboBox<String> combopost;
395         private javax.swing.JButton jButton1;
396         private javax.swing.JButton jButton2;
397         private javax.swing.JButton jButton3;
398         private javax.swing.JLabel jLabel1;
399         private javax.swing.JLabel jLabel2;
400         private javax.swing.JLabel jLabel3;
401         private javax.swing.JLabel jLabel4;
402         private javax.swing.JLabel jLabel5;
403         private javax.swing.JLabel jLabel6;
404         private javax.swing.JLabel jLabel7;
405         private javax.swing.JLabel jLabel8;
406         private javax.swing.JPanel jPanel1;
407         private javax.swing.JPanel jPanel2;
408         private javax.swing.JPanel jPanel3;
409         private javax.swing.JTextField teid;
410         private javax.swing.JTextField tname;
411         private javax.swing.JTextField tsal;
412         private javax.swing.JTextField userid;
413         // End of variables declaration
414     }
415
```

In the above code, the following things need to be kept in mind which are:

1. **Connection Class:** It acts as a connection session between the Java program and specific database application. It is through which we send SQL queries to the database.
2. **Statement Class:** A Statement is an interface that represents a SQL statement.
3. **ResultSet:** When you execute Statement objects, and they generate ResultSet objects, which is a table of data representing a database result set. A

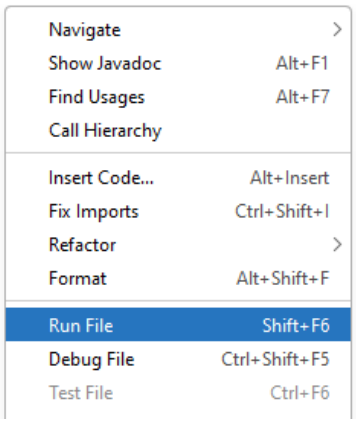
Connection object is needed to create a Statement object.

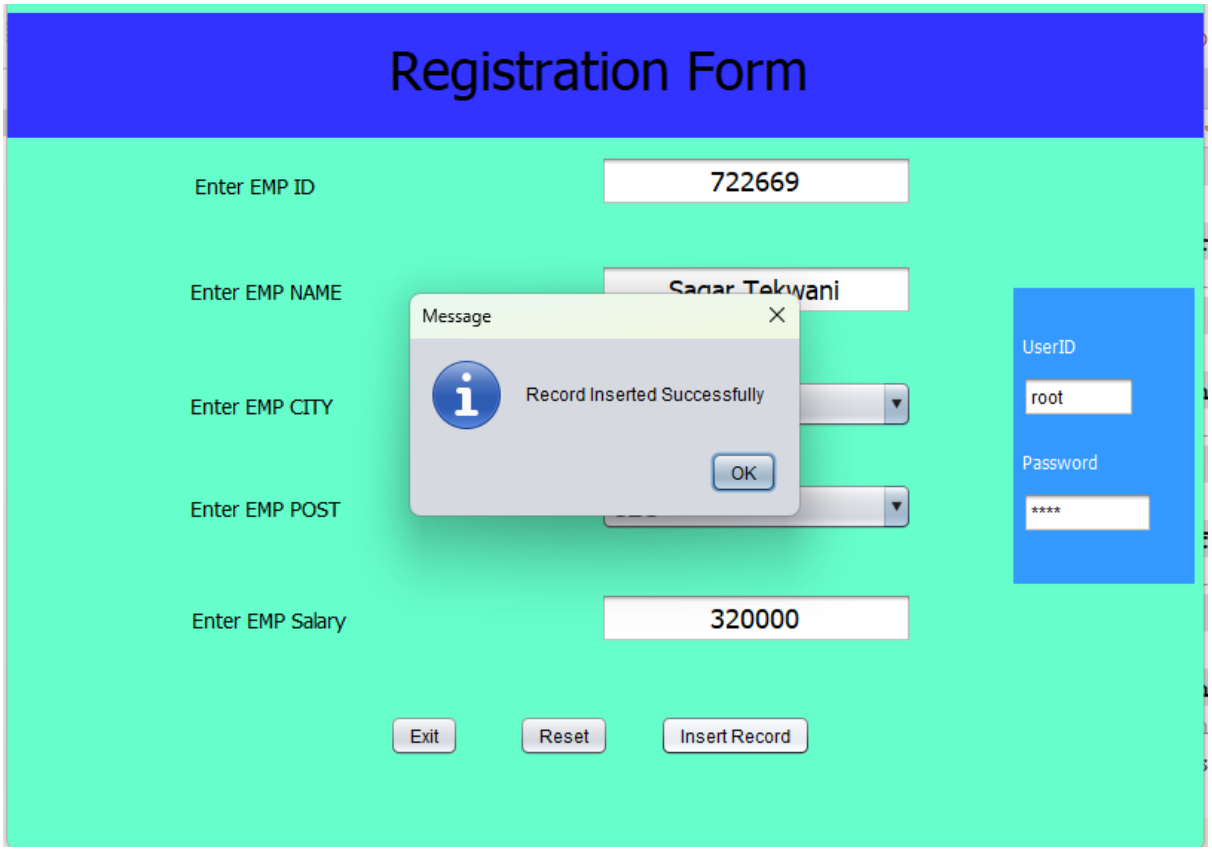
4. **JDBC Driver Registration:** To open a connection to the database from Java application, the JDBC driver should be registered with the Device Manager. Hence we use **forName()** of language package. Here *com.mysql.jdbc.Driver* is the driver name for MySQL.
5. **.getConnection():** It is used to establish a physical connection to the database by specifying the *database name(student) , username(root) and password(root).* This creates a connection object.
6. **Query Execution:** *createStatement()* creates a statement type object that holds SQL queries. Then *executeQuery/executeUpdate* method executes the SQL statement. Here it is "INSERT INTO RECORD VALUES….".
7. **Data Extraction:** The above method creates a resultset object that contains the resultant data. *(Now see the code below)* rs is the variable that stores the resultant dataset and hence we use a *.get<Type>()* method to obtain data.
8. **while(rs.next()):** *(See code below)* Since we need data containing multiple rows, we use a loop to access them. The *next()* method moves the cursor forward by one row.

9. **Closing open databases:** We close all open databases to clean up the environment. Thus we use- *rs.close() , stmt.close() and con.close()* methods.

10. After code is typed, right-click anywhere on the screen and select the Run File option from the drop-down menu. The final output is shown below. Input necessary details and the application is ready!

```
String id,name,city,post,uid,pass;
int sal;
id=teid.getText();
name=tname.getText();
city=(String)combocity.getSelectedItem();
post=(String)combopost.getSelectedItem();
sal=Integer.parseInt(s: tsal.getText());
uid=userid.getText();
pass=Password.getText();
try
 {
```






```
mysql> select * from emp;
+--------+--------------+---------+------+--------+
| id     | Name         | city    | post | salary |
+--------+--------------+---------+------+--------+
| 722669 | Sagar Tekwani | Gurgoan | CEO  | 320000 |
+--------+--------------+---------+------+--------+
1 row in set (0.00 sec)

mysql>
```

We have successfully created a software that allows us to store, delete the data and also we can view the data at any time we want.