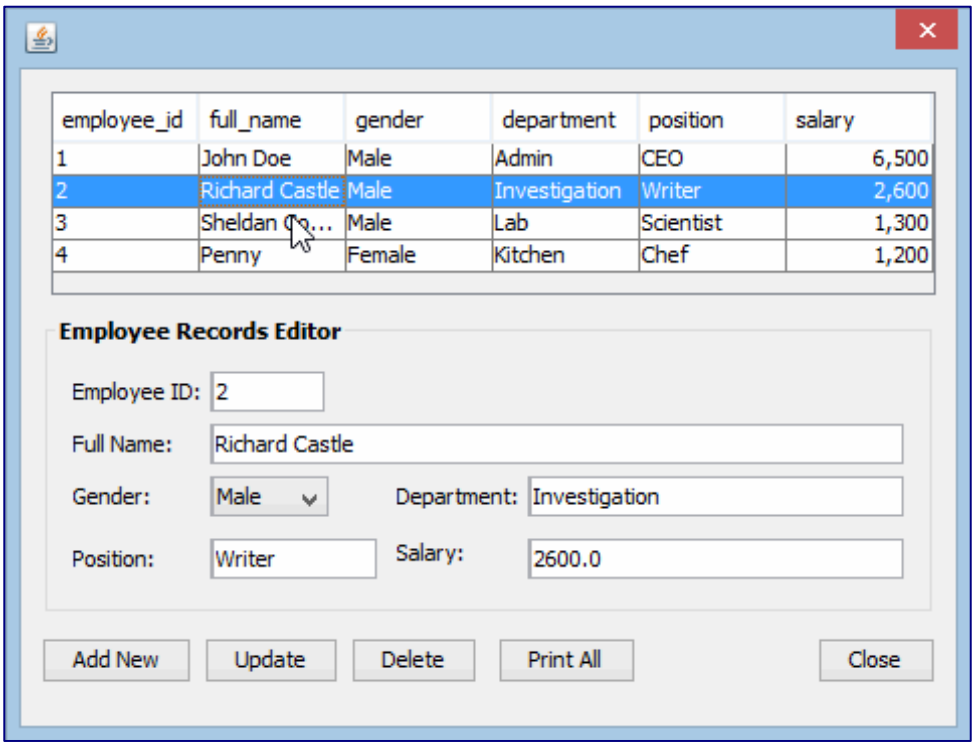
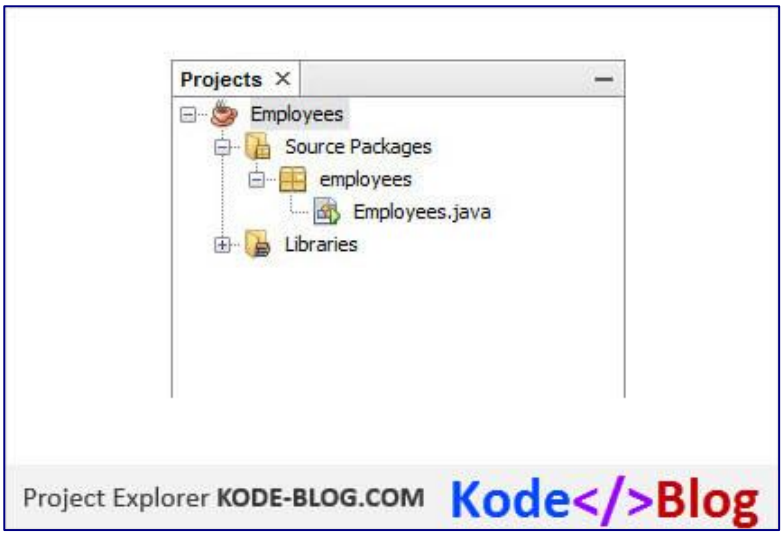
Java Swing JDBC CRUD Example

In this tutorial, we are going to develop a MySQL powered Java Swing GUI Application. We will also create a report using JasperReports. We will work with NetBeans IDE and Jasper Reports plugin. This tutorial assumes you understand the basics of Java and JDBC. If you would like to read a comprehensive guide on JDBC then I recommend you read these free step by step tutorial series on JDBC. By the time that you are done with this tutorial, you will have the following working application.

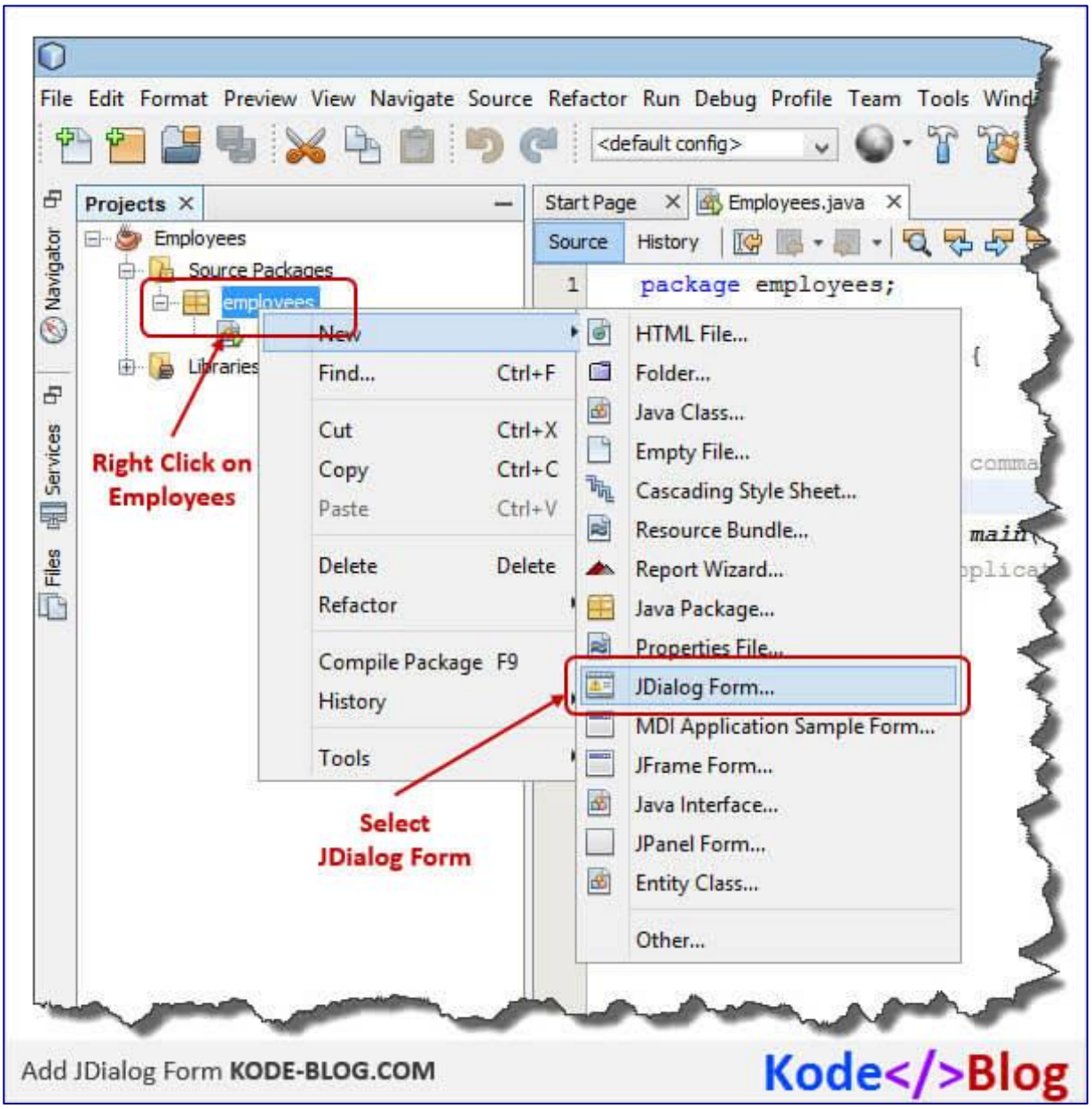


Getting started

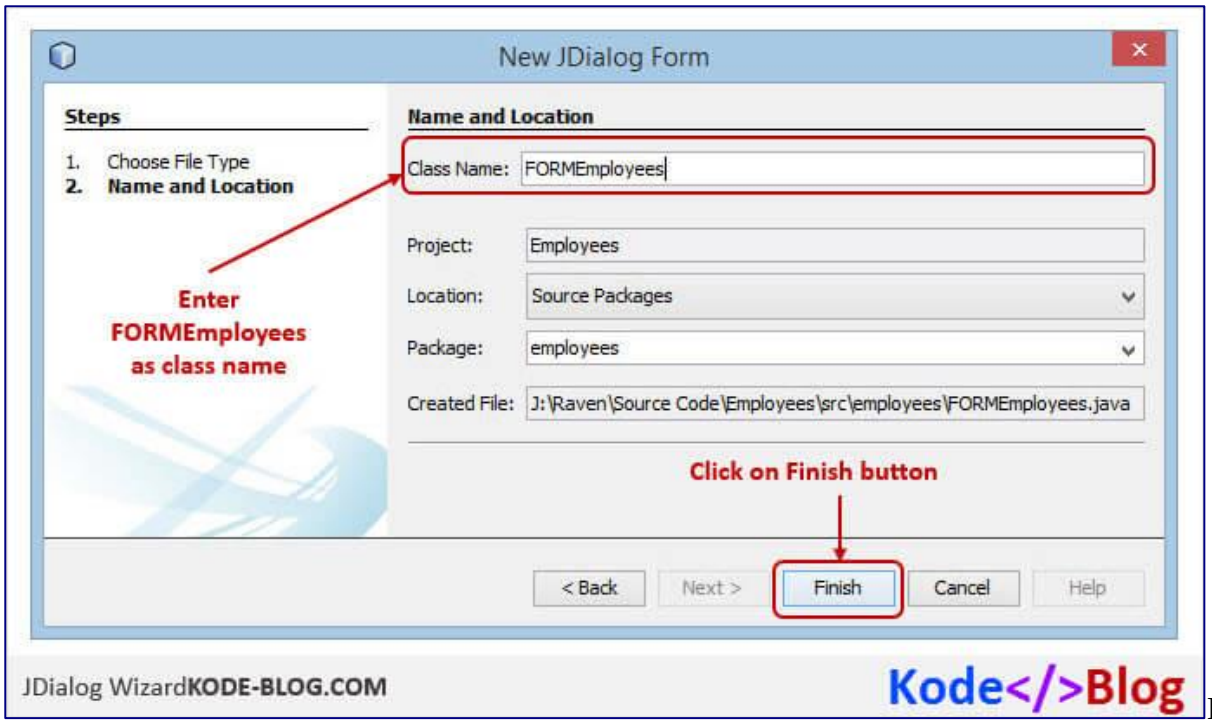
Create a new project in NetBeans IDE. Name the project Employees After the project has successfully been created, it will appear in the project explorer Your project explorer will appear as follows



Right click on employees' package



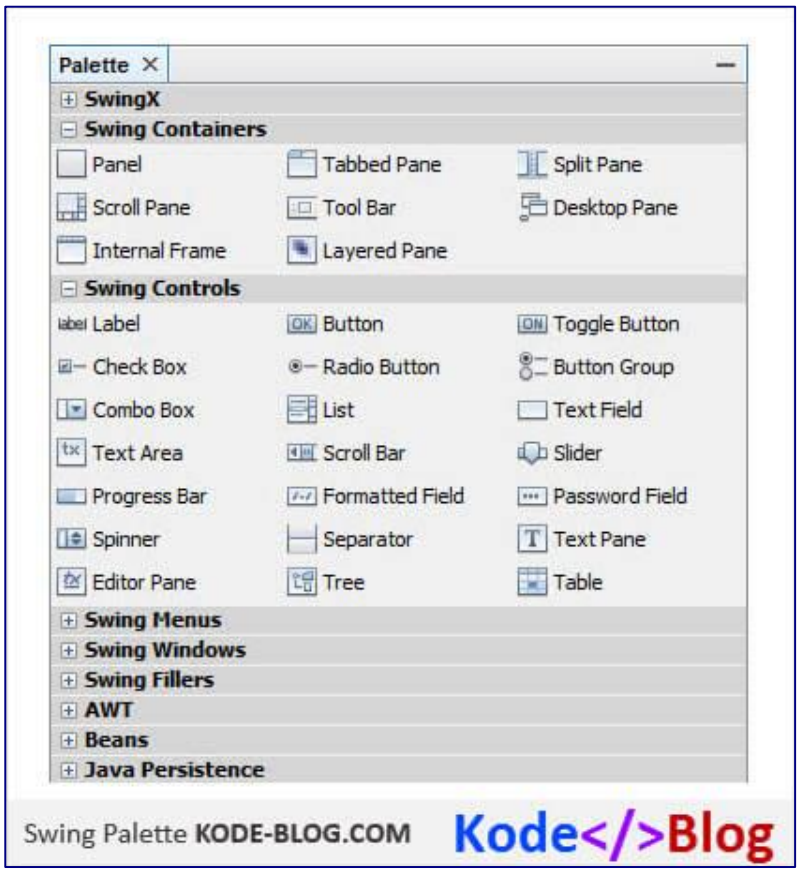
Select JDialog Form You will get the following wizard



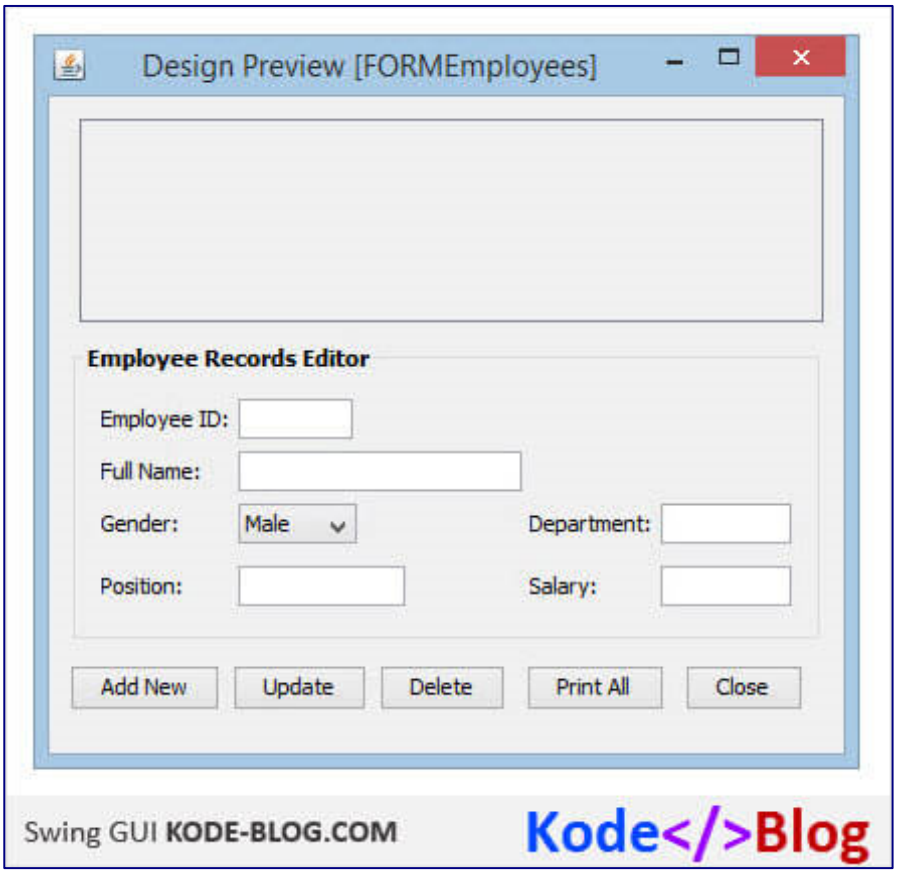
Enter FORMEmployees as class name Click on Finish button Double click on FORMEmployees in the project explorer to open it

Creating a Swing GUI in NetBeans IDE

Click on windows menu from the IDE Select IDE tools Click on Palette Alternatively, you can click Ctrl + Shift + 8 You will get the following palette



You can drag and drop controls from the palette onto the form. Create the GUI as shown in the image below



The following table shows the control name and properties for the above GUI

S/N	CONTROL	PROPERTY	VALUE
1	JTable1	model	Delete all default lines from table settings
2	JPanel1	border	Select titled border and enter Employee Records Editor
3	JLabel1	text	Employee ID:
4	JLabel2	text	Full Name:
5	JLabel3	text	Gender:
6	JLabel4	text	Department:
7	JLabel5	text	Position:
8	JLabel6	text	Salary
9	JButton1	text	Add New
10		Variable Name	btnAddNew
11	JButton2	text	Update
		variable	btnUpdate
12	JButton3	text	Delete
		Variable Name	btnDelete
13	JButton4	text	Print All
		variable	btnPrintAll
14	JButton5	text	Close

		variable	btnClose
15	JTextField1	text	blank
		Variable Name	txtEmployeeID
		enabled	false
16	JTextField2	text	blank
		Variable Name	txtFullName
17	JTextField3	text	blank
		Variable Name	txtEmployeeID
18	JCombo	item	Add Male and Female items
		Variable Name	cboGender
19	JTextField4	text	blank
		Variable Name	txtDepartment
20	JTextField5	text	blank
		Variable Name	txtPosition
21	JTextField	text	blank
		Variable Name	txtSalary

MySQL JDBC Drivers

For this tutorial, we will work with MySQL database. Before we proceed, run the following script to create the database that we will work with.

```
CREATE SCHEMA `employees` ;

CREATE TABLE `employees`.`employees` (
  `employee_id` INT NOT NULL AUTO_INCREMENT,
  `full_name` VARCHAR(75) NULL,
  `gender` VARCHAR(45) NULL,
  `department` VARCHAR(45) NULL,
  `position` VARCHAR(45) NULL,
  `salary` DOUBLE NULL,
  PRIMARY KEY (`employee_id`));

INSERT INTO `employees`.`employees` (`full_name`, `gender`, `department`, `position`, `salary`)
VALUES ('John Doe', 'Male', 'Admin', 'CEO', '6500')
,('Richard Castle', 'Male', 'Investigation', 'Writer', '2600')
,('Sheldan Cooper', 'Male', 'Lab', 'Scientist', '1300'),
('Penny', 'Female', 'Kitchen', 'Chef', '1200');
```

Adding MySQL JDBC drivers to the project.

If you are new to JDBC, then read this tutorial JDBC Connection - How to Connect to the Database (With pictures) on how to add a jar file to a project Right click on Libraries in the project explorer

Select Add Jar/Folder Browse to the path of the MySQL JDBC Driver jar file Select it and click on Open button

How to connect to MySQL using JDBC

Create a new class Config.java Add the following code

```
package employees;

public class Config {

    public static final String DATABASE_NAME = "employees";
    public static final String DATABASE_SERVER = "localhost";
    public static final String DATABASE_USER_ID = "root";
    public static final String DATABASE_PASSWORD = "melody";

    public static final String connection_url = "jdbc:mysql://" + DATABASE_SERVER + "/" +
DATABASE_NAME;
}
```

HERE,

⑩ We have created variables for the database name, server name, user id, password and the database JDBC connection string"

Create a new class DBUtilities.java Add the following code

```
package employees;

import java.sql.*;

public class DBUtilities {

    Connection connection = null;
    Statement statement = null;
    ResultSet resultSet = null;

    public DBUtilities() throws SQLException {
        try {
            connection = DriverManager.getConnection(Config.connection_url,
Config.DATABASE_USER_ID, Config.DATABASE_PASSWORD);

        } catch (SQLException ex) {
            System.out.println("The following error has occured: " + ex.getMessage());
        }
    }

    public Connection getConnection() {
        return connection;
    }
```

```
    public void ExecuteSQLStatement(String sql_stmt) {
        try {
            statement = connection.createStatement();

            statement.executeUpdate(sql_stmt);
        } catch (SQLException ex) {
            System.out.println("The following error has occured: " + ex.getMessage());
        }
    }
}
```

HERE,

- ❿ **"DBUtilities()"** initializes the class and establishes a database connection
- ❿ **"ExecuteSQLStatement()"** executes INSERT,UPDATE and DELETE statements.

Populate JTable with Database Data

Create a new class ResultSetTableModel.java Add the following code

```
package employees;

import java.sql.*;
import javax.swing.table.AbstractTableModel;

public class ResultSetTableModel extends AbstractTableModel {

    private Connection connection;
    private final Statement statement;
    private ResultSet resultSet;
    private ResultSetMetaData metaData;
    private int numberOfRows;

    private boolean connectedToDatabase = false;

    private void SetDatabaseURL() throws SQLException {
        try {
            connection = DriverManager.getConnection(Config.connection_url,
Config.DATABASE_USER_ID, Config.DATABASE_PASSWORD);
        } catch (SQLException sex) {
            System.out.println(sex.getMessage());
        }
    }

    public ResultSetTableModel(String query) throws SQLException {

        SetDatabaseURL();
        connection = DriverManager.getConnection(Config.connection_url, Config.DATABASE_USER_ID,
Config.DATABASE_PASSWORD);
```

```
        statement = connection.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_READ_ONLY);

        connectedToDatabase = true;

        if (!connectedToDatabase) {
            throw new IllegalStateException("Not Connected to Database");
        }

        resultSet = statement.executeQuery(query);
        metaData = resultSet.getMetaData();
        resultSet.last();
        numberOfRows = resultSet.getRow();

        fireTableStructureChanged();

    }

    @Override
    public Class getColumnClass(int column) throws IllegalStateException {
        if (!connectedToDatabase) {
            throw new IllegalStateException("Not Connected to Database");
        }

        try {
            String className = metaData.getColumnClassName(column + 1);
            return Class.forName(className);
        } catch (ClassNotFoundException | SQLException ex) {
            System.out.println(ex.getMessage());
        }

        return Object.class;
    }

    @Override
    public int getColumnCount() throws IllegalStateException {
        if (!connectedToDatabase) {
            throw new IllegalStateException("Not Connected to Database");
        }

        try {
            return metaData.getColumnCount();
        } catch (SQLException sex) {
            System.out.println(sex.getMessage());
        }

        return 0;
    }
```

```
@Override
public String getColumnName(int column) throws IllegalStateException {
    if (!connectedToDatabase) {
        throw new IllegalStateException("Not Connected to Database");
    }

    try {
        return metaData.getColumnName(column + 1);
    } catch (SQLException sex) {
        System.out.println(sex.getMessage());
    }

    return "";
}

@Override
public int getRowCount() throws IllegalStateException {
    if (!connectedToDatabase) {
        throw new IllegalStateException("Not Connected to Database");
    }

    return numberOfRows;
}

@Override
public Object getValueAt(int row, int column)
        throws IllegalStateException {
    if (!connectedToDatabase) {
        throw new IllegalStateException("Not Connected to Database");
    }

    try {
        resultSet.absolute(row + 1);
        return resultSet.getObject(column + 1);
    } catch (SQLException sex) {
        System.out.println(sex.getMessage());
    }

    return "";
}

public void disconnectFromDatabase() {
    if (connectedToDatabase) {
        try {
            resultSet.close();
            statement.close();
            connection.close();
```

```
            } catch (SQLException sex) {
                System.out.println(sex.getMessage());
            } finally {
                connectedToDatabase = false;
            }
        }
    }
}
```

HERE,

- ⑩ **"ResultSetTableModel"** is the class constructor that accepts a string parameter. The string parameter is the SELECT SQL statement used to retrieve data. This class extends AbstractTableModel class.
- ⑩ **"The other methods"** are abstract methods of AbstractTableModel class

How to add, update and delete records using JDBC

Add the following code to FORMEmployees

```
    boolean addRecord = false;

    private void clearInputBoxes() {
        txtEmployeeId.setText("");
        txtFullName.setText("");
        cboGender.setSelectedItem("");
        txtDepartment.setText("");
        txtPosition.setText("");
        txtSalary.setText("");
    }

    private void addNew() throws SQLException {
        String sql_stmt = "INSERT INTO employees (full_name,gender,department,position,salary)";
        sql_stmt += " VALUES ('" + txtFullName.getText() + "','" +
cboGender.getSelectedItem().toString() + "','" + txtDepartment.getText() + "','" +
txtPosition.getText() + "','" + txtSalary.getText() + "')";

        DBUtilities dbUtilities = new DBUtilities();

        dbUtilities.ExecuteSQLStatement(sql_stmt);
    }

    private void updateRecord() throws SQLException {
        String sql_stmt = "UPDATE employees SET full_name = '" + txtFullName.getText() + "'";
        sql_stmt += ",gender = '" + cboGender.getSelectedItem().toString() + "'";
        sql_stmt += ",department = '" + txtDepartment.getText() + "'";
        sql_stmt += ",position = '" + txtPosition.getText() + "'";
        sql_stmt += ",salary = '" + txtSalary.getText() + "'";
        sql_stmt += " WHERE employee_id = '" + txtEmployeeId.getText() + "'";
```

```
        DBUtilities dbUtilities = new DBUtilities();

        dbUtilities.ExecuteSQLStatement(sql_stmt);
    }

    private void deleteRecord() throws SQLException {
        String sql_stmt = "DELETE FROM employees WHERE employee_id = '" +
txtEmployeeId.getText() + "'";

        DBUtilities dbUtilities = new DBUtilities();

        dbUtilities.ExecuteSQLStatement(sql_stmt);
    }

    private void loadRecords() throws SQLException {

        String sql_stmt = "SELECT * FROM employees;";

        ResultSetTableModel tableModel = new ResultSetTableModel(sql_stmt);

        jTable1.setModel(tableModel);

        jTable1.getSelectionModel().addListSelectionListener((ListSelectionEvent event) -> {
            try {
                if (jTable1.getSelectedRow() >= 0) {
                    Object employee_id = jTable1.getValueAt(jTable1.getSelectedRow(), 0);
                    Object full_name = jTable1.getValueAt(jTable1.getSelectedRow(), 1);
                    Object gender = jTable1.getValueAt(jTable1.getSelectedRow(), 2);
                    Object department = jTable1.getValueAt(jTable1.getSelectedRow(), 3);
                    Object position = jTable1.getValueAt(jTable1.getSelectedRow(), 4);
                    Object salary = jTable1.getValueAt(jTable1.getSelectedRow(), 5);

                    txtEmployeeId.setText(employee_id.toString());
                    txtFullName.setText(full_name.toString());
                    cboGender.setSelectedItem(gender.toString());
                    txtDepartment.setText(department.toString());
                    txtPosition.setText(position.toString());
                    txtSalary.setText(salary.toString());
                }
            } catch (Exception ex) {
                System.out.println(ex.getMessage());
            }
        });

        DefaultTableCellRenderer rightRenderer = new DefaultTableCellRenderer();
        rightRenderer.setHorizontalAlignment(SwingConstants.LEFT);
        jTable1.getColumnModel().getColumn(0).setCellRenderer(rightRenderer);
    }
```

Add the following code to btnAddNewActionPerformed event

```
addRecord = true;

clearInputBoxes();

txtFullName.requestFocus();
```

Add the following code to btnUpdateActionPerformed event

```
        int dialogResult = JOptionPane.showConfirmDialog(null, "Are you sure you want to update this record?", "Confirm Update Record?", JOptionPane.YES_NO_OPTION);

        if (dialogResult == JOptionPane.YES_OPTION) {
            try {
                if (addRecord == true) {
                    addNew();
                } else {
                    updateRecord();
                }

                addRecord = false;

                loadRecords();
            } catch (SQLException ex) {
                System.out.println(ex.getMessage());
            }
        }
```

Add the following code to btnDeleteActionPerformed event

```
        int dialogResult = JOptionPane.showConfirmDialog(null, "Are you sure you want to delete this record?", "Confirm Delete Record?", JOptionPane.YES_NO_OPTION);

        if (dialogResult == JOptionPane.YES_OPTION) {
            try {
                deleteRecord();

                loadRecords();
            } catch (SQLException ex) {
                System.out.println(ex.getMessage());
            }
        }
```

Add the following code to btnCloseActionPerformed event

```
System.exit(1);
```

Java cross platform Swing Look and Feel

We want our application to have the look and feel of the platform that the program is running on. Open Employees.Java class Add the following code

```
        try {
            // Set System L&F
            UIManager.setLookAndFeel(
                    UIManager.getSystemLookAndFeelClassName());
        } catch (UnsupportedLookAndFeelException | ClassNotFoundException |
InstantiationException | IllegalAccessException ex) {
            System.out.println(ex.getMessage());
        }
```

JDialog open in center

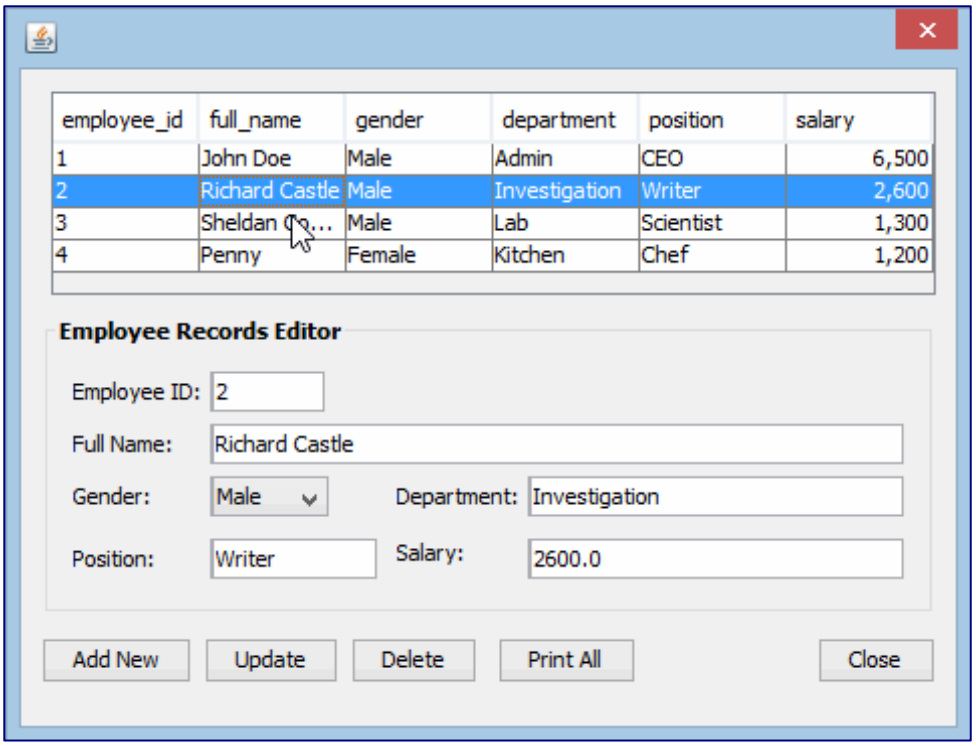
Just below the code for the look and feel, add the following code

```
        FORMEmployees sForm = new FORMEmployees(null, false);
        sForm.setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE);
        sForm.pack();
        sForm.setLocationRelativeTo(null);

        sForm.setVisible(true);
```

Testing the project

Run the project. You will get the following results.



You should also test add new, update and delete existing records.

Summary

In this tutorial exercise, we have created a Java Swing GUI with JDBC CRUD functionality. We also worked with a JTable and learnt how to populate a JTable with data from the database.