LAB ACTIVITY 4: CONSTRUCT SQL QUERY

Learning Outcomes

This lab sheet encompasses 5 activities. (Activity 3A, 3B, 3C, 3D, 3E)

By the end of this lab, student should be able to:
1. Establish a connection to the specific database using specific driver
2. Write SQL statement to insert delete view and update data

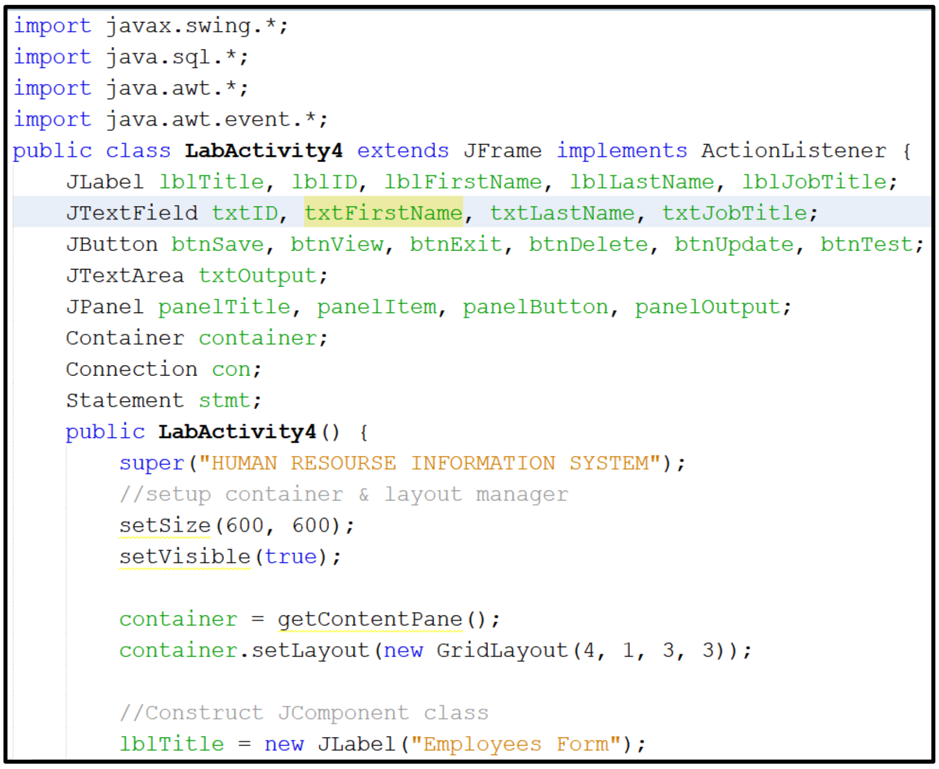
Hardware / software: NetBeans/Eclipse, JDK SDK 6 and above

Activity 3A (CLO1)

Activity Outcome: Establish the connection between database and java application

Students are required to create a new project name with the tittle of LabActivity4

Procedure:
Step 1: Create a java application with the name of LabActivity4.java. Student are required to copy and follow the source code as shown in Figure 3A_1 – Figure3A_5.



```java
import javax.swing.*;
import java.sql.*;
import java.awt.*;
import java.awt.event.*;
public class LabActivity4 extends JFrame implements ActionListener {
    JLabel lblTitle, lblID, lblFirstName, lblLastName, lblJobTitle;
    JTextField txtID, txtFirstName, txtLastName, txtJobTitle;
    JButton btnSave, btnView, btnExit, btnDelete, btnUpdate, btnTest;
    JTextArea txtOutput;
    JPanel panelTitle, panelItem, panelButton, panelOutput;
    Container container;
    Connection con;
    Statement stmt;
    public LabActivity4() {
        super("HUMAN RESOURSE INFORMATION SYSTEM");
        //setup container & layout manager
        setSize(600, 600);
        setVisible(true);

        container = getContentPane();
        container.setLayout(new GridLayout(4, 1, 3, 3));

        //Construct JComponent class
        lblTitle = new JLabel("Employees Form");
```

Figure 3A_1: LabActivity4.java source code

```
//Construct JComponent class
lblTitle = new JLabel("Employees Form");
lblID = new JLabel("ID : ");
lblFirstName = new JLabel("First Name :");
lblLastName = new JLabel("Last Name :");
lblJobTitle = new JLabel("Job Title :");

txtID = new JTextField(15);
txtFirstName = new JTextField(15);
txtLastName = new JTextField(15);
txtJobTitle = new JTextField(15);
btnTest = new JButton("TEST CONNECTION");
btnSave = new JButton("SAVE");
btnView = new JButton("VIEW");
btnExit = new JButton("EXIT");
btnDelete = new JButton("DELETE");
btnUpdate = new JButton("UPDATE");

txtOutput = new JTextArea();
txtOutput.setColumns(50);
txtOutput.setEditable(false);
```

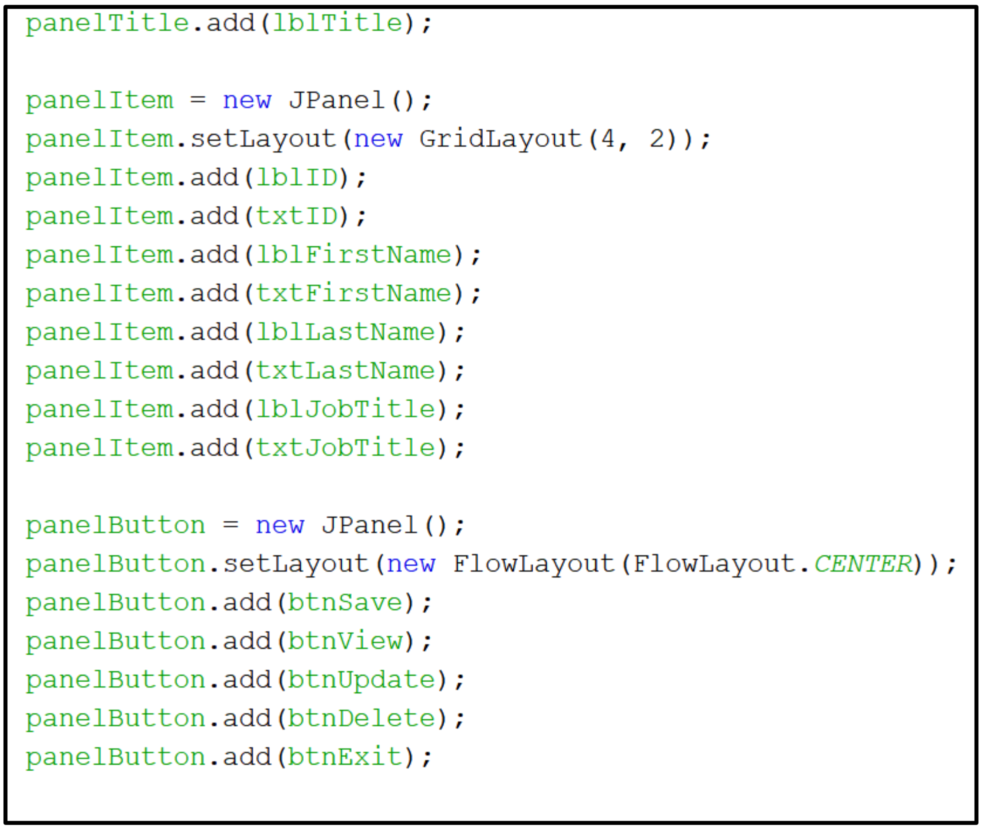
Figure 3A_2: LabActivity4.java source code

```
txtOutput.setEditable(false);

//add GUI components into panel
panelTitle = new JPanel();
panelTitle.setLayout(new FlowLayout(FlowLayout.CENTER));
panelTitle.add(lblTitle);

panelItem = new JPanel();
panelItem.setLayout(new GridLayout(4, 2));
panelItem.add(lblID);
panelItem.add(txtID);
panelItem.add(lblFirstName);
panelItem.add(txtFirstName);
panelItem.add(lblLastName);
panelItem.add(txtLastName);
panelItem.add(lblJobTitle);
panelItem.add(txtJobTitle);
```

Figure 3A_3: LabActivity4.java source code

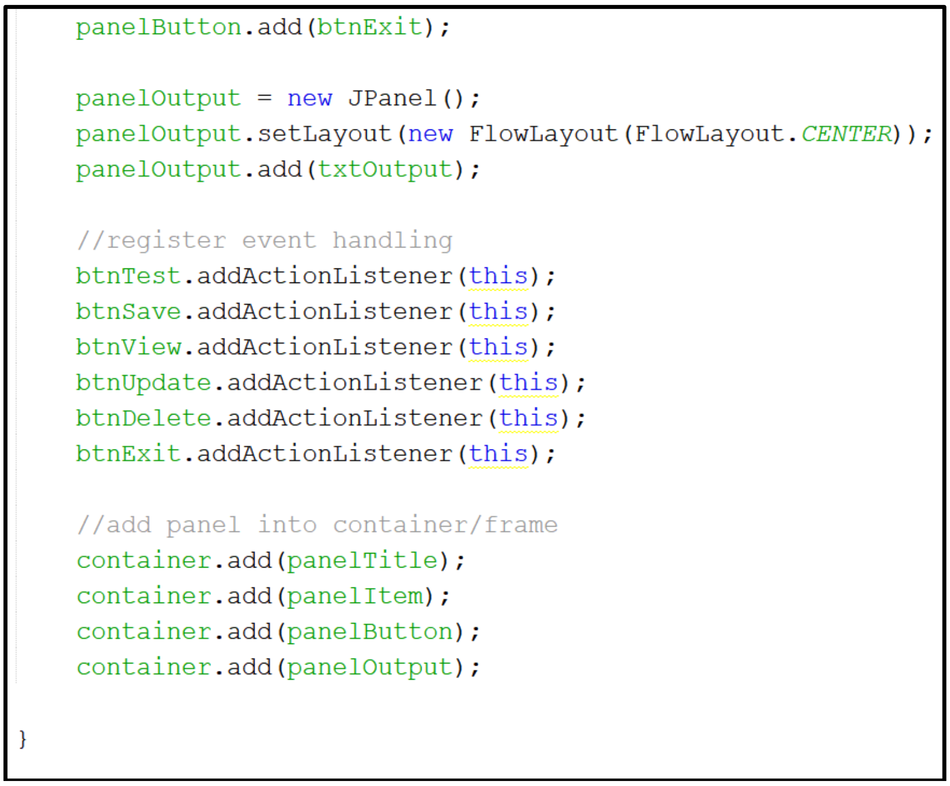


```java
panelTitle.add(lblTitle);

panelItem = new JPanel();
panelItem.setLayout(new GridLayout(4, 2));
panelItem.add(lblID);
panelItem.add(txtID);
panelItem.add(lblFirstName);
panelItem.add(txtFirstName);
panelItem.add(lblLastName);
panelItem.add(txtLastName);
panelItem.add(lblJobTitle);
panelItem.add(txtJobTitle);

panelButton = new JPanel();
panelButton.setLayout(new FlowLayout(FlowLayout.CENTER));
panelButton.add(btnSave);
panelButton.add(btnView);
panelButton.add(btnUpdate);
panelButton.add(btnDelete);
panelButton.add(btnExit);
```

Figure 3A_4: LabActivity4.java source code



```java
    panelButton.add(btnExit);

    panelOutput = new JPanel();
    panelOutput.setLayout(new FlowLayout(FlowLayout.CENTER));
    panelOutput.add(txtOutput);

    //register event handling
    btnTest.addActionListener(this);
    btnSave.addActionListener(this);
    btnView.addActionListener(this);
    btnUpdate.addActionListener(this);
    btnDelete.addActionListener(this);
    btnExit.addActionListener(this);

    //add panel into container/frame
    container.add(panelTitle);
    container.add(panelItem);
    container.add(panelButton);
    container.add(panelOutput);

}
```

Figure 3A_5: LabActivity4.java source code

The LabActivity4.java output should be as shown in Figure3A_6.

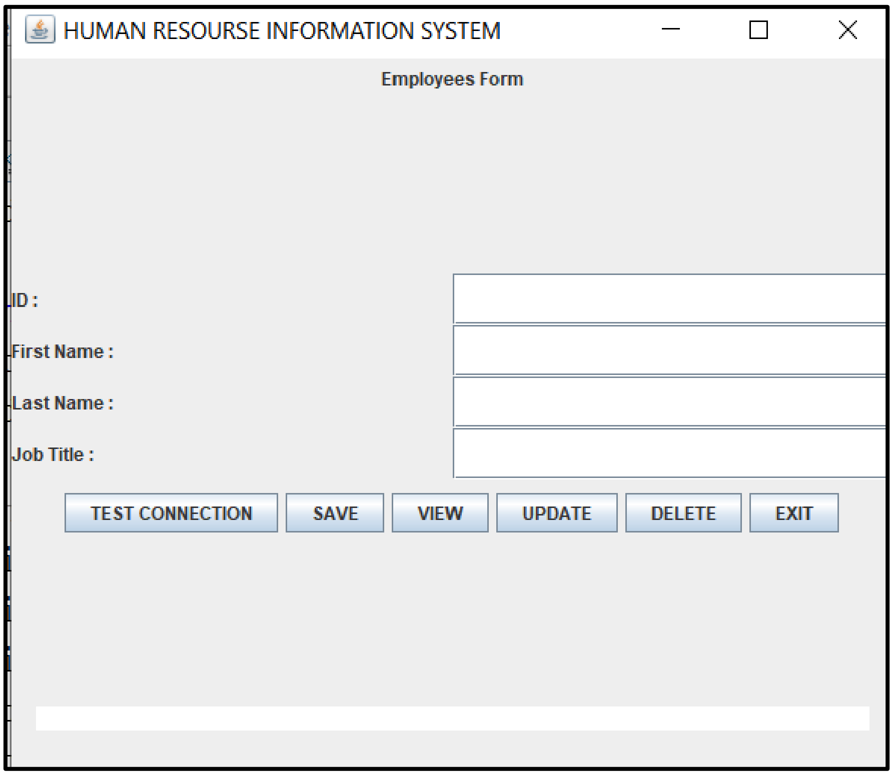
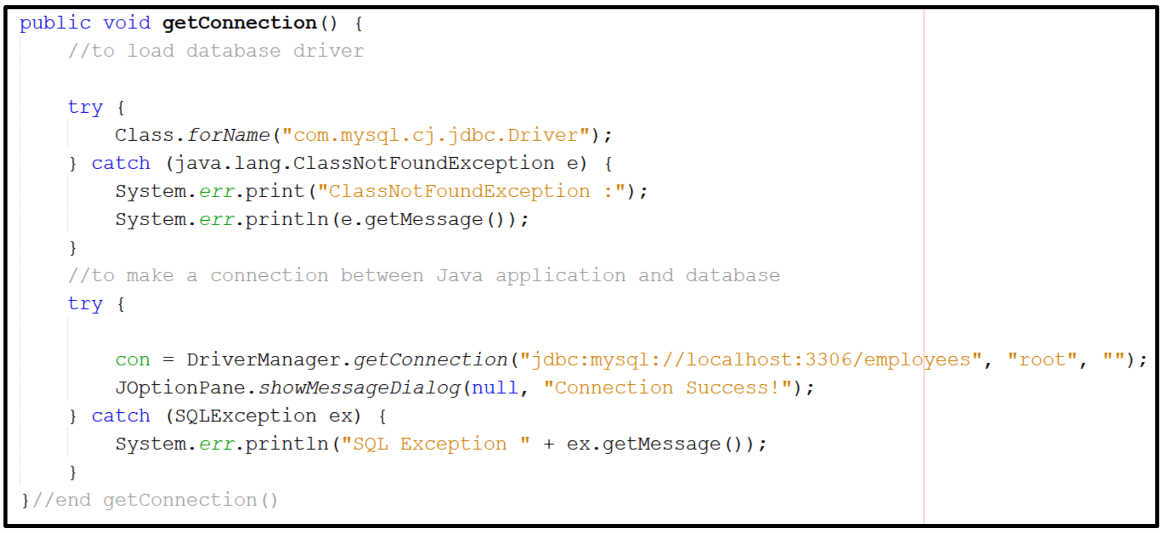


Figure 3A_6: LabActivity4.java output

Step 2: Create a getConnection() method to establish the connection between the java application and database. Figure 3A_7 shows the source code to establish the database connection.



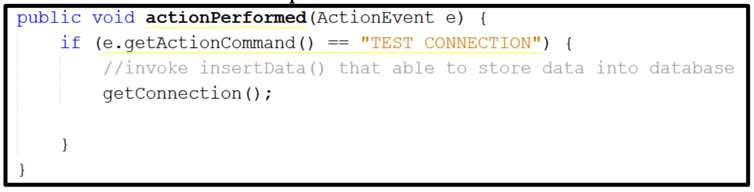
```java
public void getConnection() {
    //to load database driver

    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
    } catch (java.lang.ClassNotFoundException e) {
        System.err.print("ClassNotFoundException :");
        System.err.println(e.getMessage());
    }
    //to make a connection between Java application and database
    try {

        con = DriverManager.getConnection("jdbc:mysql://localhost:3306/employees", "root", "");
        JOptionPane.showMessageDialog(null, "Connection Success!");
    } catch (SQLException ex) {
        System.err.println("SQL Exception " + ex.getMessage());
    }
}//end getConnection()
```

Figure 3A_7: getConnection source code

Step 4: Create actionPerformed method to handle specific event



```java
public void actionPerformed(ActionEvent e) {
    if (e.getActionCommand() == "TEST CONNECTION") {
        //invoke insertData() that able to store data into database
        getConnection();

    }
}
```

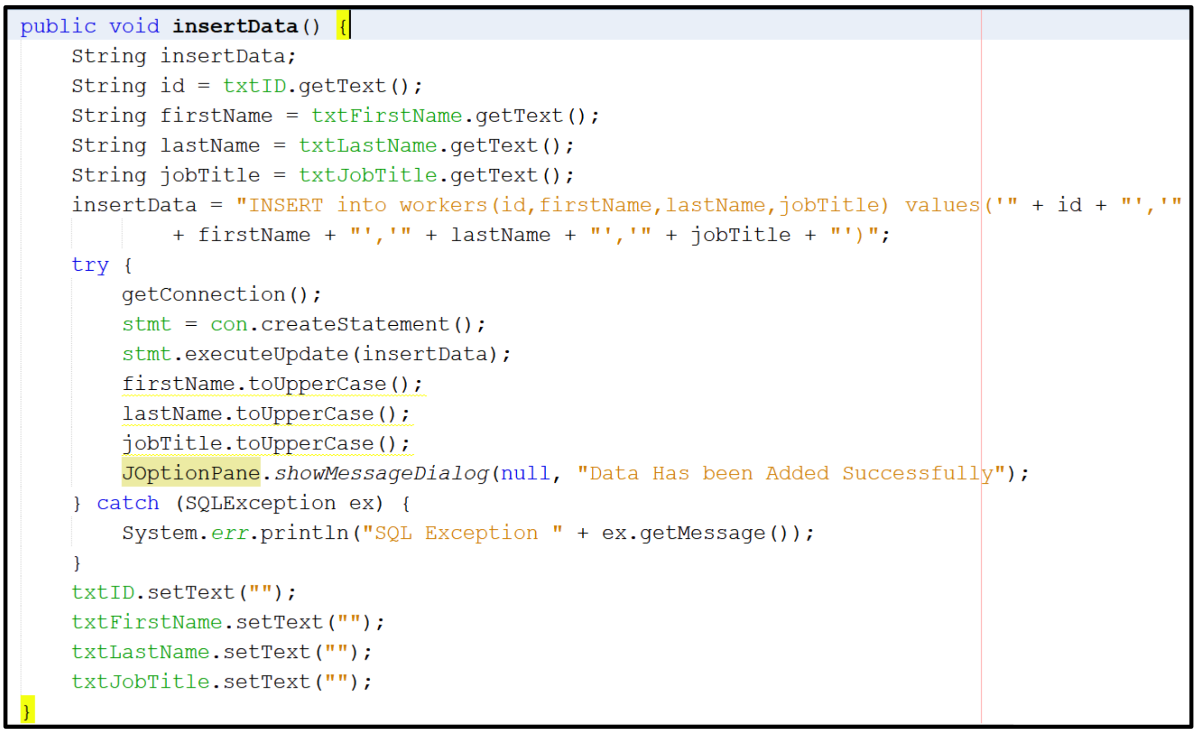
Figure 3A_7: actionPerformed source code

Step 5: Save, compile and the Java application. Observe the output.

Activity 3B (CLO1)

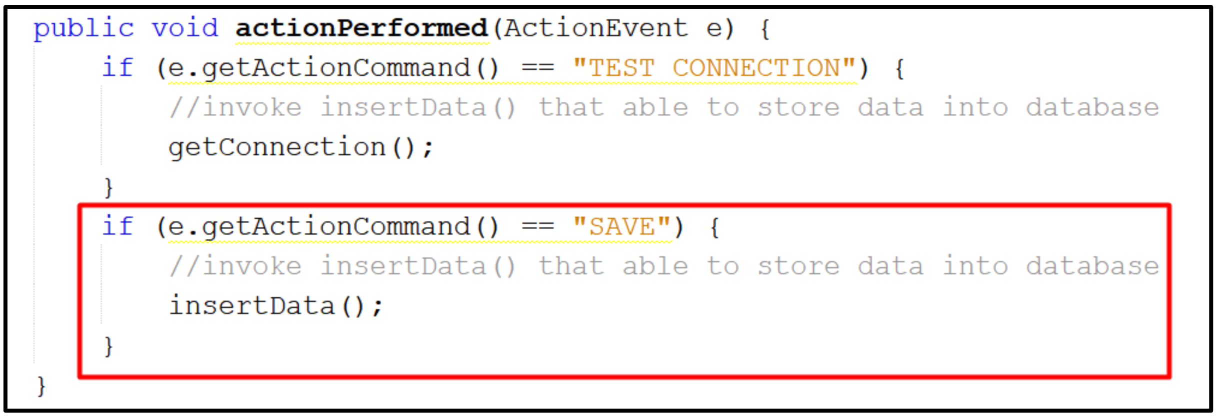
Activity Outcome: Create Insert query to add data to the database.

Step 1: Create a insertData() method to insert data to specific database. Figure 3B_1 shows the source code to read data from user input then save the data to the database. Then, modify the actionPerformed method and add the code as shown in Figure 3B_2. Save, compile, run and observe the output.



```java
public void insertData() {
    String insertData;
    String id = txtID.getText();
    String firstName = txtFirstName.getText();
    String lastName = txtLastName.getText();
    String jobTitle = txtJobTitle.getText();
    insertData = "INSERT into workers(id,firstName,lastName,jobTitle) values('" + id + "','"
            + firstName + "','" + lastName + "','" + jobTitle + "')";
    try {
        getConnection();
        stmt = con.createStatement();
        stmt.executeUpdate(insertData);
        firstName.toUpperCase();
        lastName.toUpperCase();
        jobTitle.toUpperCase();
        JOptionPane.showMessageDialog(null, "Data Has been Added Successfully");
    } catch (SQLException ex) {
        System.err.println("SQL Exception " + ex.getMessage());
    }
    txtID.setText("");
    txtFirstName.setText("");
    txtLastName.setText("");
    txtJobTitle.setText("");
}
```

Figure 3B_1: insertData source code



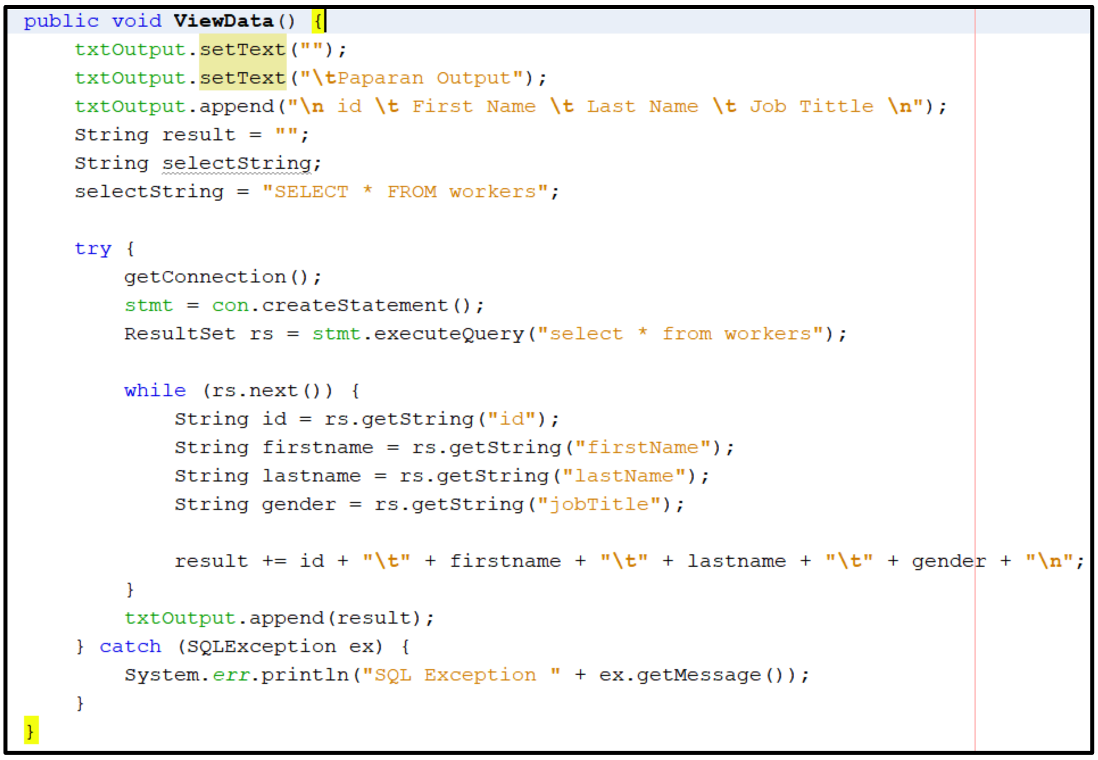
```java
public void actionPerformed(ActionEvent e) {
    if (e.getActionCommand() == "TEST CONNECTION") {
        //invoke insertData() that able to store data into database
        getConnection();
    }
    if (e.getActionCommand() == "SAVE") {
        //invoke insertData() that able to store data into database
        insertData();
    }
}
```

Figure 3B_2: actionPerformed source code

Activity 3C (CLO1)

Activity Outcome: Create View query to view data to the database.

Step 1: Create a ViewData() method to view data gathered from the database. Figure 3C_1 shows the source code to select all data from specific table in database in display using appropriate method. Then, modify the actionPerformed method and add the code as shown in Figure 3C_2. Save, compile, run and observe the output.



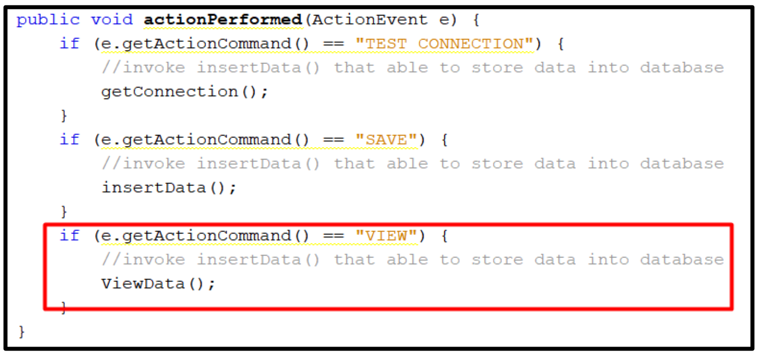
```
public void ViewData() {
    txtOutput.setText("");
    txtOutput.setText("\tPaparan Output");
    txtOutput.append("\n id \t First Name \t Last Name \t Job Tittle \n");
    String result = "";
    String selectString;
    selectString = "SELECT * FROM workers";

    try {
        getConnection();
        stmt = con.createStatement();
        ResultSet rs = stmt.executeQuery("select * from workers");

        while (rs.next()) {
            String id = rs.getString("id");
            String firstname = rs.getString("firstName");
            String lastname = rs.getString("lastName");
            String gender = rs.getString("jobTitle");

            result += id + "\t" + firstname + "\t" + lastname + "\t" + gender + "\n";
        }
        txtOutput.append(result);
    } catch (SQLException ex) {
        System.err.println("SQL Exception " + ex.getMessage());
    }
}
```

Figure 3C_1: ViewData source code



```
public void actionPerformed(ActionEvent e) {
    if (e.getActionCommand() == "TEST CONNECTION") {
        //invoke insertData() that able to store data into database
        getConnection();
    }
    if (e.getActionCommand() == "SAVE") {
        //invoke insertData() that able to store data into database
        insertData();
    }
    if (e.getActionCommand() == "VIEW") {
        //invoke insertData() that able to store data into database
        ViewData();
    }
}
```

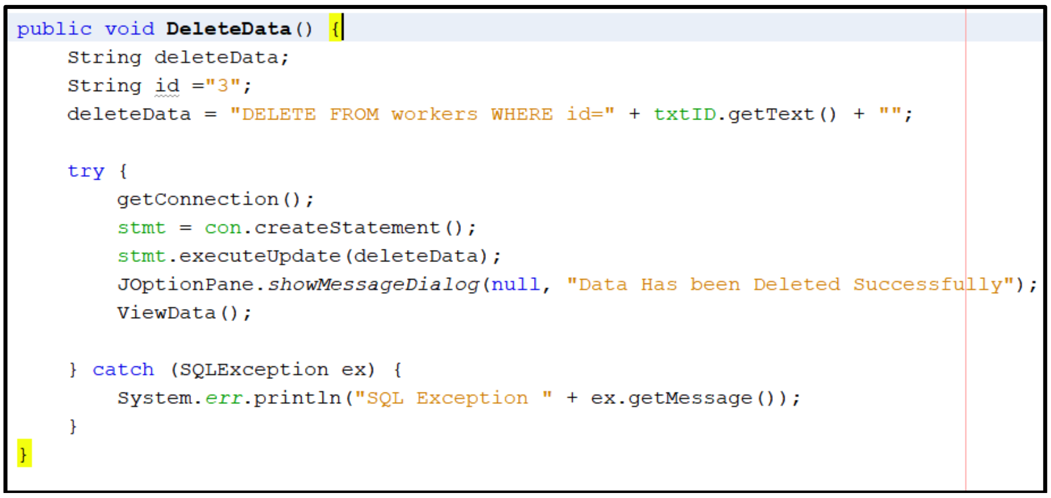
Figure 3C_2: actionPerformed source code

Activity 3D (CLO1)

individual task

Activity Outcome: Create delete query to delete specific data from the database.

Step 1: Create a DeleteData() method to delete specific data from the database. Figure 3D_1 shows the source code to delete specific data from specific table in database. Then, modify the actionPerformed method and add the code as shown in Figure 3D_2. Save, compile, run and observe the output.

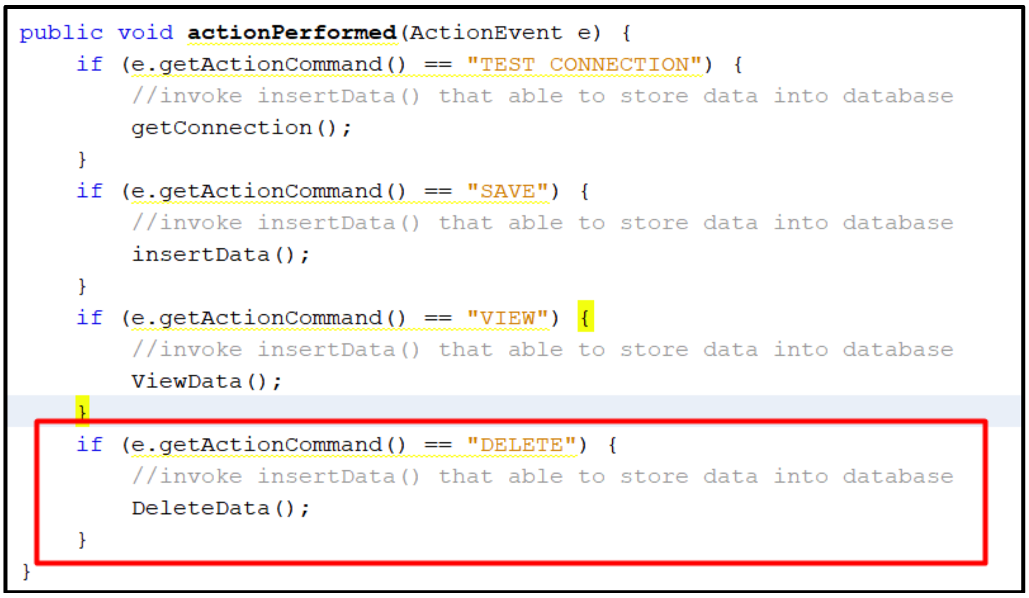


```java
public void DeleteData() {
    String deleteData;
    String id ="3";
    deleteData = "DELETE FROM workers WHERE id=" + txtID.getText() + "";

    try {
        getConnection();
        stmt = con.createStatement();
        stmt.executeUpdate(deleteData);
        JOptionPane.showMessageDialog(null, "Data Has been Deleted Successfully");
        ViewData();

    } catch (SQLException ex) {
        System.err.println("SQL Exception " + ex.getMessage());
    }
}
```

Figure 3D_1: DeleteData source code



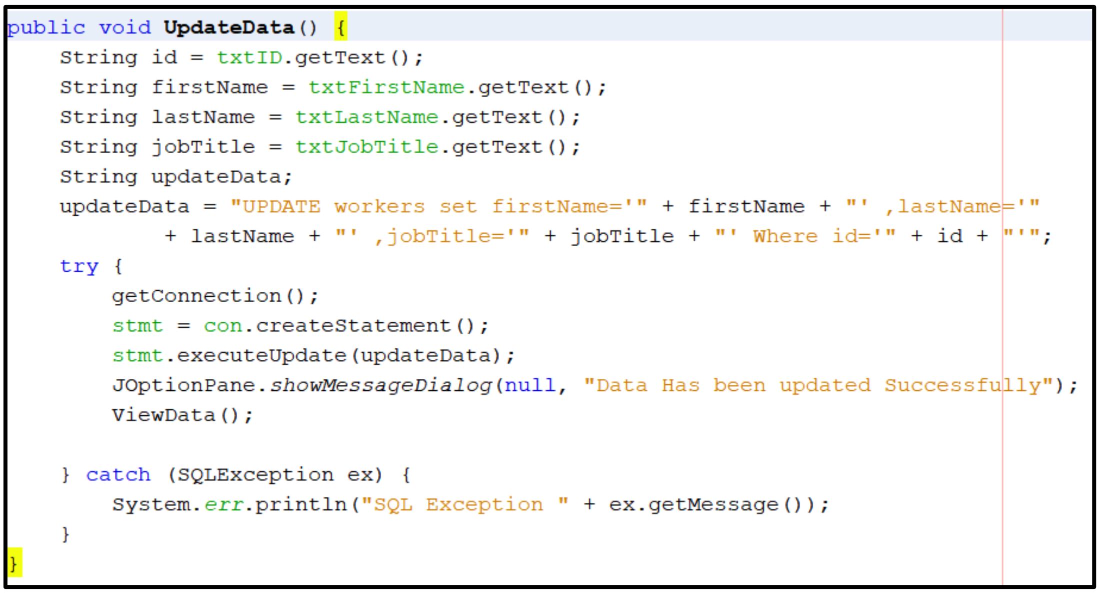
```java
public void actionPerformed(ActionEvent e) {
    if (e.getActionCommand() == "TEST CONNECTION") {
        //invoke insertData() that able to store data into database
        getConnection();
    }
    if (e.getActionCommand() == "SAVE") {
        //invoke insertData() that able to store data into database
        insertData();
    }
    if (e.getActionCommand() == "VIEW") {
        //invoke insertData() that able to store data into database
        ViewData();
    }
    if (e.getActionCommand() == "DELETE") {
        //invoke insertData() that able to store data into database
        DeleteData();
    }
}
```

Figure 3D_2: actionPerformed source code

Activity 3E (CLO1)

Activity Outcome: Create update query to update specific data from the database.

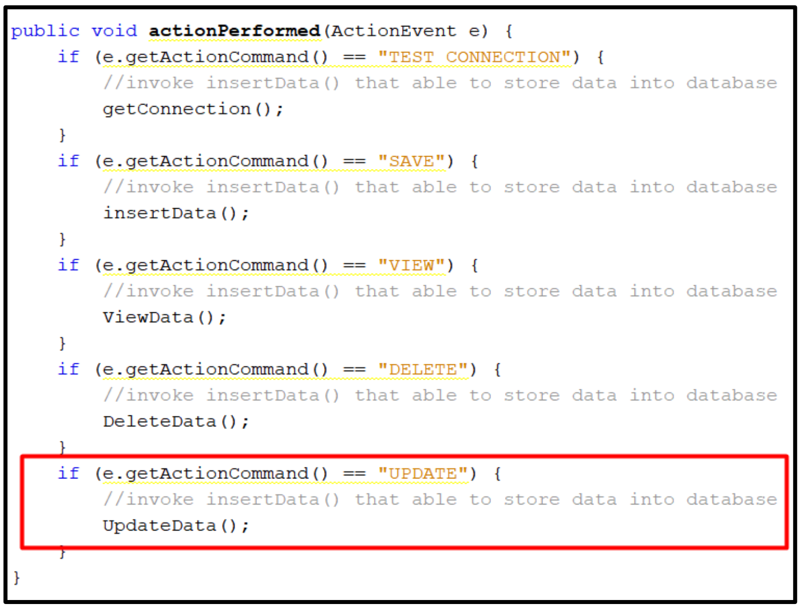
Step 1: Create a UpdateData() method to update specific data from the database. Figure 3E_1 shows the source code to update specific data from specific table in database. Then, modify the actionPerformed method and add the code as shown in Figure 3E_2. Save, compile, run and observe the output.



```
public void UpdateData() {
    String id = txtID.getText();
    String firstName = txtFirstName.getText();
    String lastName = txtLastName.getText();
    String jobTitle = txtJobTitle.getText();
    String updateData;
    updateData = "UPDATE workers set firstName='" + firstName + "' ,lastName='"
            + lastName + "' ,jobTitle='" + jobTitle + "' Where id='" + id + "'";
    try {
        getConnection();
        stmt = con.createStatement();
        stmt.executeUpdate(updateData);
        JOptionPane.showMessageDialog(null, "Data Has been updated Successfully");
        ViewData();

    } catch (SQLException ex) {
        System.err.println("SQL Exception " + ex.getMessage());
    }
}
```

Figure 3E_1: DeleteData source code



```
public void actionPerformed(ActionEvent e) {
    if (e.getActionCommand() == "TEST CONNECTION") {
        //invoke insertData() that able to store data into database
        getConnection();
    }
    if (e.getActionCommand() == "SAVE") {
        //invoke insertData() that able to store data into database
        insertData();
    }
    if (e.getActionCommand() == "VIEW") {
        //invoke insertData() that able to store data into database
        ViewData();
    }
    if (e.getActionCommand() == "DELETE") {
        //invoke insertData() that able to store data into database
        DeleteData();
    }
    if (e.getActionCommand() == "UPDATE") {
        //invoke insertData() that able to store data into database
        UpdateData();
    }
}
```

Figure 3E_2: actionPerformed source code