```
package Project;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
import java.time.LocalDate;

class Login extends JFrame implements ActionListener {
   JTextField tf1 = new JTextField();
   JTextField tf2 = new JTextField();

   public Login() {
      getContentPane().setBackground(Color.white);
      setLayout(null);

      JLabel l1 = new JLabel("UserName: ");
      l1.setBounds(150, 250, 150, 30);
      tf1.setBounds(250, 250, 250, 30);

      JLabel l2 = new JLabel("Password: ");
      l2.setBounds(150, 300, 150, 30);
      tf2.setBounds(250, 300, 250, 30);

      JButton b1 = new JButton("SignIn Domestic");
      b1.setBounds(300, 400, 150, 30);

      JButton b2 = new JButton("SignIn Commercial");
      b2.setBounds(300, 450, 150, 30);

      JButton b3 = new JButton("Cancel");
      b3.setBounds(300, 500, 150, 30);
```

```
    add(l1);
    add(tf1);
    add(l2);
    add(tf2);
    add(b1);
    add(b2);
    add(b3);

    b1.addActionListener(this);
    b2.addActionListener(this);
    b3.addActionListener(this);

    setSize(800, 900);
    setVisible(true);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}

@Override
public void actionPerformed(ActionEvent e) {
    String action = e.getActionCommand();
    if (action.equals("SignIn Domestic")) {
        openNewPage("DOMESTICCUSTOMERS");
    } else if (action.equals("SignIn Commercial")) {
        openNewPage("COMMERCIALCUSTOMERS");
    } else if (action.equals("Cancel")) {
        dispose();
    }
}

public void openNewPage(String customerType) {
```

```
JFrame newPage = new JFrame(customerType + " Page");
newPage.setSize(600, 600);
newPage.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

String imagePath = "C:/Program Files (x86)/NetBeans 8.2/TNEB1.png";
ImageIcon backgroundImage = new ImageIcon(imagePath);
Image img = backgroundImage.getImage();
Image scaledImg = img.getScaledInstance(600, 600, Image.SCALE_SMOOTH);
backgroundImage = new ImageIcon(scaledImg);
JLabel label = new JLabel(backgroundImage);

label.setBounds(650, 100, 600, 600);
newPage.add(label);

JLabel l3 = new JLabel("Receipt Date:      " + LocalDate.now());
l3.setBounds(50, 100, 200, 30);

JLabel l4 = new JLabel("Old Units: ");
l4.setBounds(50, 150, 150, 30);

JTextField oldUnitsField = new JTextField();
oldUnitsField.setBounds(150, 150, 250, 30);
oldUnitsField.setEditable(false);

JLabel l5 = new JLabel("New Units: ");
l5.setBounds(50, 200, 150, 30);
JTextField newUnitsField = new JTextField();
newUnitsField.setBounds(150, 200, 250, 30);

JLabel l6 = new JLabel("Units Used: ");
l6.setBounds(50, 250, 150, 30);
```

```
        JTextField unitsUsedField = new JTextField();
        unitsUsedField.setBounds(150, 250, 250, 30);
        unitsUsedField.setEditable(false);

        JLabel l7 = new JLabel("Amount To Pay: ");
        l7.setBounds(50, 300, 150, 30);
        JTextField amountField = new JTextField();
        amountField.setBounds(150, 300, 250, 30);
        amountField.setEditable(false);

        JButton calculateButton = new JButton("Calculate");
        calculateButton.setBounds(200, 350, 150, 30);

        label.setLayout(null);
        label.add(l3);
        label.add(l4);
        label.add(oldUnitsField);
        label.add(l5);
        label.add(newUnitsField);
        label.add(l6);
        label.add(unitsUsedField);
        label.add(l7);
        label.add(amountField);
        label.add(calculateButton);

        fetchOldUnits(customerType, oldUnitsField);

        calculateButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                calculateAmount(customerType, newUnitsField, oldUnitsField, unitsUsedField,
amountField);
```

```
        }
    });

    newPage.setVisible(true);
  }

  public void fetchOldUnits(String customerType, JTextField oldUnitsField) {
    try {
      String userName = tf1.getText().trim();
      Class.forName("org.apache.derby.jdbc.ClientDriver");
      Connection con = DriverManager.getConnection("jdbc:derby://localhost:1527/EB", "rohith",
"rohith2002");

      String query = "SELECT OLDUNITS FROM " + customerType + " WHERE NAME = ?";
      PreparedStatement pst = con.prepareStatement(query);
      pst.setString(1, userName);
      ResultSet rs = pst.executeQuery();

      if (rs.next()) {
        int oldUnits = rs.getInt("OLDUNITS");
        oldUnitsField.setText(String.valueOf(oldUnits));
      }

      rs.close();
      pst.close();
      con.close();
    } catch (Exception ex) {
      ex.printStackTrace();
    }
  }
```

```
    public void calculateAmount(String customerType, JTextField newUnitsField, JTextField
oldUnitsField, JTextField unitsUsedField, JTextField amountField) {
        try {
            int oldUnits = Integer.parseInt(oldUnitsField.getText());
            int newUnits = Integer.parseInt(newUnitsField.getText());
            int unitsUsed = newUnits - oldUnits;
            unitsUsedField.setText(String.valueOf(unitsUsed));

            if (unitsUsed < 0) {
                JOptionPane.showMessageDialog(null, "New units must be greater than or equal to old
units.");
                return;
            }

            double totalPrice = 0;

            if (customerType.equals("DOMESTICCUSTOMERS")) {
                totalPrice = (unitsUsed <= 500) ? calculateLessThan500(unitsUsed) :
calculateAbove500(unitsUsed);
            } else if (customerType.equals("COMMERCIALCUSTOMERS")) {
                totalPrice = (unitsUsed <= 500) ? calculateCommercialLessThan500(unitsUsed) :
calculateCommercialAbove500(unitsUsed);
            }

            amountField.setText(String.format("%.2f", totalPrice));

            updateOldUnits(customerType, newUnits);

        } catch (NumberFormatException ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(null, "Please enter valid numbers.");
        } catch (SQLException ex) {
```

```
            ex.printStackTrace();
            JOptionPane.showMessageDialog(null, "Error connecting to the database.");
        }
    }

    public void updateOldUnits(String customerType, int newUnits) throws SQLException {
        String updateQuery = "UPDATE " + customerType + " SET OLDUNITS = ? WHERE NAME = ?";
        Connection con = DriverManager.getConnection("jdbc:derby://localhost:1527/EB", "rohith",
"rohith2002");

        PreparedStatement pstUpdate = con.prepareStatement(updateQuery);
        pstUpdate.setInt(1, newUnits);
        pstUpdate.setString(2, tf1.getText());
        pstUpdate.executeUpdate();

        pstUpdate.close();
        con.close();
    }

    private double calculateLessThan500(int unitsUsed) {
        double total = 0;
        if (unitsUsed <= 100) {
            return 0;
        } else if (unitsUsed <= 200) {
            total = (unitsUsed - 100) * 2.35;
        } else if (unitsUsed <= 400) {
            total = 100 * 2.35 + (unitsUsed - 200) * 4.7;
        } else if (unitsUsed <= 500) {
            total = 100 * 2.35 + 200 * 4.7 + (unitsUsed - 400) * 6.3;
        }
        return total;
```

```
    }

    private double calculateAbove500(int unitsUsed) {
        return 100 * 2.35 + 200 * 4.7 + 100 * 6.3 + (unitsUsed - 500) * 8.0;
    }

    private double calculateCommercialLessThan500(int unitsUsed) {
        double total = 0;
        if (unitsUsed <= 100) {
            total = unitsUsed * 5.5;
        } else if (unitsUsed <= 200) {
            total = 100 * 5.5 + (unitsUsed - 100) * 6.0;
        } else if (unitsUsed <= 500) {
            total = 100 * 5.5 + 100 * 6.0 + (unitsUsed - 200) * 6.5;
        }
        return total;
    }

    private double calculateCommercialAbove500(int unitsUsed) {

        return 100 * 5.5 + 100 * 6.0 + 300 * 6.5 + (unitsUsed - 500) * 7.0;
    }
}

public class Project1 {
    public static void main(String[] args) {
        new Login();
    }
}
```