**Riphah International University – Lahore Campus**

**Advanced Computer Programming**

**Project Title: Solar Management System**

**Name: Muhammad Ali (38611)**

**Ibraheem (42896)**

**Ahmad Hassan (42908)**

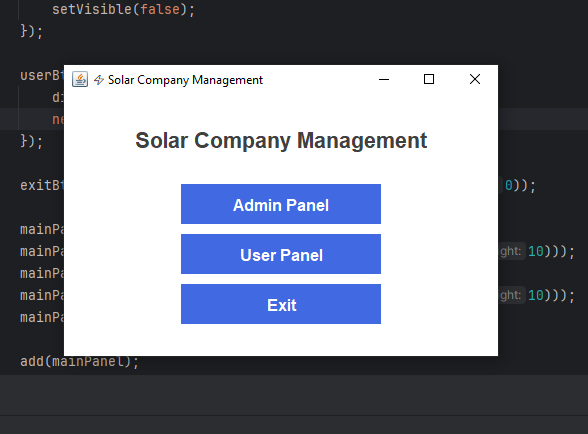
**Section: BSCS-5**

**Submitted To:   
 Prof. Asim Mansha**

**Code:**

import javax.swing.\*;  
import java.awt.\*;  
  
public class MainMenu extends JFrame {  
 public MainMenu() {  
 setTitle("⚡ Solar Company Management");  
 setSize(450, 300);  
 setDefaultCloseOperation(*EXIT\_ON\_CLOSE*);  
 setLocationRelativeTo(null);  
  
 JPanel mainPanel = new JPanel();  
 mainPanel.setBorder(BorderFactory.*createEmptyBorder*(30, 30, 30, 30));  
 mainPanel.setLayout(new BoxLayout(mainPanel, BoxLayout.*Y\_AXIS*));  
 mainPanel.setBackground(Color.*WHITE*);  
  
 JLabel titleLabel = new JLabel("Solar Company Management", SwingConstants.*CENTER*);  
 titleLabel.setFont(new Font("SansSerif", Font.*BOLD*, 22));  
 titleLabel.setAlignmentX(Component.*CENTER\_ALIGNMENT*);  
 titleLabel.setForeground(Color.*DARK\_GRAY*);  
 mainPanel.add(titleLabel);  
 mainPanel.add(Box.*createRigidArea*(new Dimension(0, 30)));  
  
 JButton adminBtn = createStyledButton("Admin Panel");  
 JButton userBtn = createStyledButton("User Panel");  
 JButton exitBtn = createStyledButton("Exit");  
  
 adminBtn.addActionListener(e -> {  
 new AdminLogin(this).setVisible(true);  
 setVisible(false);  
 });  
  
 userBtn.addActionListener(e -> {  
 dispose();  
 new UserPanel(this).setVisible(true);  
 });  
  
 exitBtn.addActionListener(e -> System.*exit*(0));  
  
 mainPanel.add(adminBtn);  
 mainPanel.add(Box.*createRigidArea*(new Dimension(0, 10)));  
 mainPanel.add(userBtn);  
 mainPanel.add(Box.*createRigidArea*(new Dimension(0, 10)));  
 mainPanel.add(exitBtn);  
  
 add(mainPanel);  
 }  
  
 private JButton createStyledButton(String text) {  
 JButton button = new JButton(text);  
 button.setFont(new Font("SansSerif", Font.*BOLD*, 16));  
 button.setBackground(new Color(65, 105, 225));  
 button.setForeground(Color.*WHITE*);  
 button.setFocusPainted(false);  
 button.setOpaque(true);  
 button.setContentAreaFilled(true);  
 button.setBorderPainted(false);  
 button.setAlignmentX(Component.*CENTER\_ALIGNMENT*);  
 button.setMaximumSize(new Dimension(200, 40));  
 return button;  
 }  
  
 public static void main(String[] args) {  
 SwingUtilities.*invokeLater*(() -> new MainMenu().setVisible(true));  
 }  
}

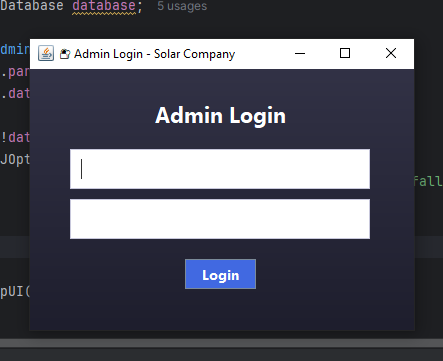
**Output:**



**Admin Login:**

import javax.swing.\*;  
import java.awt.\*;  
import java.sql.\*;  
  
public class AdminLogin extends JFrame {  
 private JFrame parent;  
 private Database database;  
  
 public AdminLogin(JFrame parent) {  
 this.parent = parent;  
 this.database = new Database();  
  
 if (!database.isConnected()) {  
 JOptionPane.*showMessageDialog*(this,  
 "Database connection failed. Using fallback credentials.",  
 "Warning",  
 JOptionPane.*WARNING\_MESSAGE*);  
 }  
  
 setupUI();  
 }  
  
 private void setupUI() {  
 setTitle("🔐 Admin Login - Solar Company");  
 setSize(400, 300);  
 setDefaultCloseOperation(*DISPOSE\_ON\_CLOSE*);  
 setLocationRelativeTo(null);  
  
 JPanel panel = new JPanel() {  
 @Override  
 protected void paintComponent(Graphics g) {  
 super.paintComponent(g);  
 Graphics2D g2d = (Graphics2D) g;  
 Color color1 = new Color(50, 50, 70);  
 Color color2 = new Color(30, 30, 45);  
 GradientPaint gp = new GradientPaint(0, 0, color1, 0, getHeight(), color2);  
 g2d.setPaint(gp);  
 g2d.fillRect(0, 0, getWidth(), getHeight());  
 }  
 };  
  
 panel.setLayout(new BoxLayout(panel, BoxLayout.*Y\_AXIS*));  
 panel.setBorder(BorderFactory.*createEmptyBorder*(30, 40, 30, 40));  
  
 JLabel titleLabel = new JLabel("Admin Login", SwingConstants.*CENTER*);  
 titleLabel.setFont(new Font("Segoe UI", Font.*BOLD*, 22));  
 titleLabel.setForeground(Color.*WHITE*);  
 titleLabel.setAlignmentX(Component.*CENTER\_ALIGNMENT*);  
 panel.add(titleLabel);  
 panel.add(Box.*createRigidArea*(new Dimension(0, 20)));  
  
 JTextField usernameField = new JTextField();  
 JPasswordField passwordField = new JPasswordField();  
 styleField(usernameField, "Username");  
 styleField(passwordField, "Password");  
  
 panel.add(usernameField);  
 panel.add(Box.*createRigidArea*(new Dimension(0, 10)));  
 panel.add(passwordField);  
 panel.add(Box.*createRigidArea*(new Dimension(0, 20)));  
  
 JButton loginBtn = new JButton("Login");  
 loginBtn.setAlignmentX(Component.*CENTER\_ALIGNMENT*);  
 loginBtn.setPreferredSize(new Dimension(100, 40));  
 loginBtn.setFont(new Font("Segoe UI", Font.*BOLD*, 14));  
 loginBtn.setFocusPainted(false);  
 loginBtn.setBackground(new Color(65, 105, 225));  
 loginBtn.setForeground(Color.*WHITE*);  
 loginBtn.setCursor(new Cursor(Cursor.*HAND\_CURSOR*));  
  
 loginBtn.addActionListener(e -> {  
 String username = usernameField.getText().trim();  
 String password = new String(passwordField.getPassword()).trim();  
  
 if (validateLogin(username, password)) {  
 JOptionPane.*showMessageDialog*(this, "Login successful!", "Success", JOptionPane.*INFORMATION\_MESSAGE*);  
 dispose();  
 if (parent != null) parent.dispose();  
 new AdminPanel().setVisible(true);  
 } else {  
 JOptionPane.*showMessageDialog*(this, "Invalid credentials!", "Error", JOptionPane.*ERROR\_MESSAGE*);  
 }  
 });  
  
 panel.add(loginBtn);  
 add(panel);  
 }  
  
 private boolean validateLogin(String username, String password) {  
 // Fallback credentials if database is not available  
 /\* if (!database.isConnected()) {  
 return username.equals("admin") && password.equals("admin123");  
 }\*/  
  
 String sql = "SELECT \* FROM adminlogin WHERE username = ? AND password = ?";  
  
 try (PreparedStatement pstmt = database.getConnection().prepareStatement(sql)) {  
 pstmt.setString(1, username);  
 pstmt.setString(2, password);  
  
 try (ResultSet rs = pstmt.executeQuery()) {  
 return rs.next(); // Returns true if a matching record was found  
 }  
 } catch (SQLException ex) {  
 JOptionPane.*showMessageDialog*(this,  
 "Error verifying credentials: " + ex.getMessage(),  
 "Database Error",  
 JOptionPane.*ERROR\_MESSAGE*);  
 return false;  
 }  
 }  
  
 private void styleField(JComponent field, String placeholder) {  
 field.setMaximumSize(new Dimension(300, 40));  
 field.setFont(new Font("Segoe UI", Font.*PLAIN*, 14));  
 field.setBorder(BorderFactory.*createCompoundBorder*(  
 BorderFactory.*createLineBorder*(new Color(200, 200, 220)),  
 BorderFactory.*createEmptyBorder*(5, 10, 5, 10)));  
 field.setForeground(Color.*BLACK*);  
 field.setBackground(Color.*WHITE*);  
 if (field instanceof JTextField) {  
 ((JTextField) field).setToolTipText(placeholder);  
 }  
 }  
  
 @Override  
 public void dispose() {  
 if (database != null) {  
 database.close();  
 }  
 super.dispose();  
 }  
}

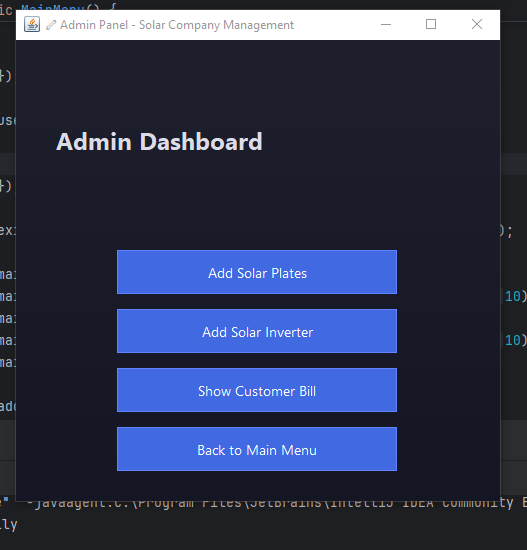
Output:



Admin panel:

import javax.swing.\*;  
import java.awt.\*;  
public class AdminPanel extends JFrame {  
 public AdminPanel() {  
 setTitle("🔧 Admin Panel - Solar Company Management");  
 setSize(500, 500);  
 setDefaultCloseOperation(*DISPOSE\_ON\_CLOSE*);  
 setLocationRelativeTo(null);  
 JPanel mainPanel = new JPanel() {  
 protected void paintComponent(Graphics g) {  
 super.paintComponent(g);  
 Graphics2D g2d = (Graphics2D) g;  
 Color color1 = new Color(30, 33, 45);  
 Color color2 = new Color(20, 23, 35);  
 GradientPaint gp = new GradientPaint(0, 0, color1, 0, getHeight(), color2);  
 g2d.setPaint(gp);  
 g2d.fillRect(0, 0, getWidth(), getHeight());  
 }  
 };  
 mainPanel.setLayout(new BoxLayout(mainPanel, BoxLayout.*Y\_AXIS*));  
 mainPanel.setBorder(BorderFactory.*createEmptyBorder*(30, 40, 30, 40));  
 JPanel headerPanel = new JPanel(new BorderLayout());  
 headerPanel.setOpaque(false);  
 JLabel title = new JLabel("Admin Dashboard");  
 title.setFont(new Font("Segoe UI", Font.*BOLD*, 24));  
 title.setForeground(new Color(220, 220, 230));  
 title.setBorder(BorderFactory.*createEmptyBorder*(0, 0, 20, 0));  
 headerPanel.add(title, BorderLayout.*WEST*);  
 mainPanel.add(headerPanel);  
 mainPanel.add(Box.*createRigidArea*(new Dimension(0, 20)));  
 JButton addPlatesBtn = createProfessionalButton("Add Solar Plates");  
 JButton addInverterBtn = createProfessionalButton("Add Solar Inverter");  
 JButton showBillBtn = createProfessionalButton("Show Customer Bill");  
 JButton backBtn = createProfessionalButton("Back to Main Menu");  
 addPlatesBtn.addActionListener(e -> new SolarPlatesInfo().setVisible(true));  
 addInverterBtn.addActionListener(e -> new SolarInverterInfo().setVisible(true));  
 showBillBtn.addActionListener(e -> new ShowCustomerBill().setVisible(true));  
 backBtn.addActionListener(e -> {  
 dispose();  
 new MainMenu().setVisible(true);  
 });  
 mainPanel.add(addPlatesBtn);  
 mainPanel.add(Box.*createRigidArea*(new Dimension(0, 15)));  
 mainPanel.add(addInverterBtn);  
 mainPanel.add(Box.*createRigidArea*(new Dimension(0, 15)));  
 mainPanel.add(showBillBtn);  
 mainPanel.add(Box.*createRigidArea*(new Dimension(0, 15)));  
 mainPanel.add(backBtn);  
 add(mainPanel);  
 }  
 private JButton createProfessionalButton(String text) {  
 JButton button = new JButton(text) {  
 @Override  
 protected void paintComponent(Graphics g) {  
 Graphics2D g2 = (Graphics2D) g;  
 g2.setRenderingHint(RenderingHints.*KEY\_ANTIALIASING*, RenderingHints.*VALUE\_ANTIALIAS\_ON*);  
 if (getModel().isPressed()) {  
 g2.setColor(new Color(65, 105, 225).darker());  
 } else if (getModel().isRollover()) {  
 g2.setColor(new Color(65, 105, 225).brighter());  
 } else {  
 g2.setColor(new Color(65, 105, 225));  
 }  
 g2.fillRect(0, 0, getWidth(), getHeight());  
  
 g2.setColor(new Color(100, 130, 255));  
 g2.drawRect(0, 0, getWidth()-1, getHeight()-1);  
 super.paintComponent(g);  
 }  
 };  
  
 button.setFont(new Font("Segoe UI", Font.*PLAIN*, 14));  
 button.setForeground(Color.*WHITE*);  
 button.setContentAreaFilled(false);  
 button.setBorder(BorderFactory.*createEmptyBorder*(12, 25, 12, 25));  
 button.setFocusPainted(false);  
 button.setAlignmentX(Component.*CENTER\_ALIGNMENT*);  
 button.setMaximumSize(new Dimension(280, 45));  
 button.setCursor(new Cursor(Cursor.*HAND\_CURSOR*));  
  
 return button;  
 }  
  
  
}

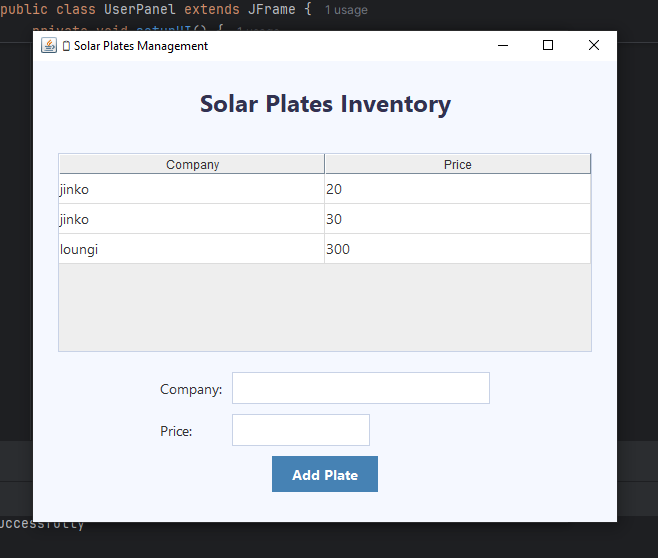
Output:



Solarpanels.java:

import javax.swing.\*;  
import javax.swing.table.DefaultTableModel;  
import java.awt.\*;  
import java.sql.\*;  
  
public class SolarPlatesInfo extends JFrame {  
 private DefaultTableModel tableModel;  
 private Database database;  
  
 public SolarPlatesInfo() {  
 database = new Database();  
 if (!database.isConnected()) {  
 JOptionPane.*showMessageDialog*(this,  
 "Failed to connect to database. Some features may not work.",  
 "Database Error",  
 JOptionPane.*ERROR\_MESSAGE*);  
 }  
  
 setupUI();  
 }  
  
 private void setupUI() {  
 setTitle("🔋 Solar Plates Management");  
 setSize(600, 500);  
 setLocationRelativeTo(null);  
 setDefaultCloseOperation(*DISPOSE\_ON\_CLOSE*);  
  
 JPanel mainPanel = new JPanel(new BorderLayout(15, 15));  
 mainPanel.setBackground(new Color(245, 248, 255));  
 mainPanel.setBorder(BorderFactory.*createEmptyBorder*(25, 25, 25, 25));  
  
 JLabel title = new JLabel("Solar Plates Inventory", SwingConstants.*CENTER*);  
 title.setFont(new Font("Segoe UI", Font.*BOLD*, 24));  
 title.setForeground(new Color(50, 50, 80));  
 title.setBorder(BorderFactory.*createEmptyBorder*(0, 0, 20, 0));  
 mainPanel.add(title, BorderLayout.*NORTH*);  
  
 String[] columns = {"Company", "Price"};  
 tableModel = new DefaultTableModel(columns, 0) {  
 @Override  
 public boolean isCellEditable(int row, int column) {  
 return false;  
 }  
 };  
  
 JTable table = new JTable(tableModel);  
 table.setFont(new Font("Segoe UI", Font.*PLAIN*, 14));  
 table.setRowHeight(30);  
 table.setSelectionBackground(new Color(200, 220, 255));  
 table.setSelectionForeground(Color.*BLACK*);  
 table.setGridColor(new Color(220, 220, 220));  
  
 JScrollPane scrollPane = new JScrollPane(table);  
 scrollPane.setBorder(BorderFactory.*createLineBorder*(new Color(200, 210, 230)));  
 mainPanel.add(scrollPane, BorderLayout.*CENTER*);  
  
 JPanel inputPanel = new JPanel(new GridBagLayout());  
 inputPanel.setBackground(new Color(245, 248, 255));  
 GridBagConstraints gbc = new GridBagConstraints();  
 gbc.insets = new Insets(5, 5, 5, 5);  
 gbc.anchor = GridBagConstraints.*WEST*;  
  
 gbc.gridx = 0;  
 gbc.gridy = 0;  
 JLabel companyLabel = new JLabel("Company:");  
 companyLabel.setFont(new Font("Segoe UI", Font.*PLAIN*, 14));  
 inputPanel.add(companyLabel, gbc);  
  
 gbc.gridx = 1;  
 JTextField nameField = new JTextField(20);  
 nameField.setFont(new Font("Segoe UI", Font.*PLAIN*, 14));  
 nameField.setBorder(BorderFactory.*createCompoundBorder*(  
 BorderFactory.*createLineBorder*(new Color(200, 210, 230)),  
 BorderFactory.*createEmptyBorder*(5, 8, 5, 8)  
 ));  
 inputPanel.add(nameField, gbc);  
  
 gbc.gridx = 0;  
 gbc.gridy = 1;  
 JLabel priceLabel = new JLabel("Price:");  
 priceLabel.setFont(new Font("Segoe UI", Font.*PLAIN*, 14));  
 inputPanel.add(priceLabel, gbc);  
  
 gbc.gridx = 1;  
 JTextField priceField = new JTextField(10);  
 priceField.setFont(new Font("Segoe UI", Font.*PLAIN*, 14));  
 priceField.setBorder(BorderFactory.*createCompoundBorder*(  
 BorderFactory.*createLineBorder*(new Color(200, 210, 230)),  
 BorderFactory.*createEmptyBorder*(5, 8, 5, 8)  
 ));  
 inputPanel.add(priceField, gbc);  
  
 gbc.gridx = 0;  
 gbc.gridy = 2;  
 gbc.gridwidth = 2;  
 gbc.anchor = GridBagConstraints.*CENTER*;  
  
 JButton addButton = new JButton("Add Plate");  
 addButton.setFont(new Font("Segoe UI", Font.*BOLD*, 14));  
 addButton.setBackground(new Color(70, 130, 180));  
 addButton.setForeground(Color.*WHITE*);  
 addButton.setFocusPainted(false);  
 addButton.setBorder(BorderFactory.*createEmptyBorder*(8, 20, 8, 20));  
  
 addButton.addMouseListener(new java.awt.event.MouseAdapter() {  
 public void mouseEntered(java.awt.event.MouseEvent evt) {  
 addButton.setBackground(new Color(90, 150, 200));  
 }  
 public void mouseExited(java.awt.event.MouseEvent evt) {  
 addButton.setBackground(new Color(70, 130, 180));  
 }  
 });  
  
 addButton.addActionListener(e -> {  
 String name = nameField.getText().trim();  
 String price = priceField.getText().trim();  
  
 if (!name.isEmpty() && !price.isEmpty()) {  
 if (addSolarPlateToDatabase(name, price)) {  
 addSolarPlateToTable(name, price);  
 nameField.setText("");  
 priceField.setText("");  
 }  
 } else {  
 JOptionPane.*showMessageDialog*(this,  
 "Please fill in both fields.",  
 "Input Error",  
 JOptionPane.*WARNING\_MESSAGE*);  
 }  
 });  
  
 inputPanel.add(addButton, gbc);  
 mainPanel.add(inputPanel, BorderLayout.*SOUTH*);  
 add(mainPanel);  
 loadExistingPlates();  
 }  
  
 private boolean addSolarPlateToDatabase(String name, String price) {  
 if (!database.isConnected()) {  
 JOptionPane.*showMessageDialog*(this,  
 "Not connected to database. Data will not be saved.",  
 "Database Error",  
 JOptionPane.*WARNING\_MESSAGE*);  
 return true;   
 }  
  
 String sql = "INSERT INTO plates (Company, Price) VALUES (?, ?)";  
  
 try (PreparedStatement pstmt = database.getConnection().prepareStatement(sql)) {  
 pstmt.setString(1, name);  
 pstmt.setString(2, price);  
 pstmt.executeUpdate();  
 return true;  
 } catch (SQLException ex) {  
 JOptionPane.*showMessageDialog*(this,  
 "Error saving to database: " + ex.getMessage(),  
 "Database Error",  
 JOptionPane.*ERROR\_MESSAGE*);  
 return false;  
 }  
 }  
  
 private void addSolarPlateToTable(String name, String price) {  
 tableModel.addRow(new Object[]{name, price});  
 }  
  
 private void loadExistingPlates() {  
 if (!database.isConnected()) return;  
  
 String sql = "SELECT Company, Price FROM plates";  
  
 try (Statement stmt = database.getConnection().createStatement();  
 ResultSet rs = stmt.executeQuery(sql)) {  
  
 while (rs.next()) {  
 String name = rs.getString("Company");  
 String price = rs.getString("Price");  
 tableModel.addRow(new Object[]{name, price});  
 }  
 } catch (SQLException ex) {  
 JOptionPane.*showMessageDialog*(this,  
 "Error loading existing plates: " + ex.getMessage(),  
 "Database Error",  
 JOptionPane.*ERROR\_MESSAGE*);  
 }  
 }  
  
 @Override  
 public void dispose() {  
 if (database != null) {  
 database.close();  
 }  
 super.dispose();  
 }  
}

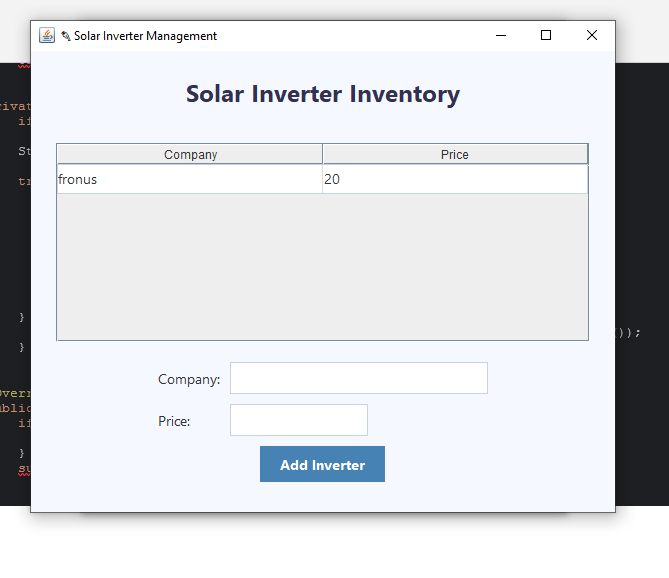
Output:



Solar inverter info:

import javax.swing.\*;  
import javax.swing.table.DefaultTableModel;  
import java.awt.\*;  
import java.sql.\*;  
  
public class SolarInverterInfo extends JFrame {  
 private final DefaultTableModel tableModel;  
 private final Database database;  
 private final JTextField nameField;  
 private final JTextField priceField;  
  
 public SolarInverterInfo() {  
 database = new Database();  
 verifyDatabaseConnection();  
 nameField = createTextField(20);  
 priceField = createTextField(10);  
 tableModel = createTableModel();  
  
 setupUI();  
 loadExistingInverters();  
 }  
  
 private void verifyDatabaseConnection() {  
 if (!database.isConnected()) {  
 JOptionPane.*showMessageDialog*(this,  
 "Failed to connect to database. Some features may not work.",  
 "Database Error",  
 JOptionPane.*ERROR\_MESSAGE*);  
 }  
 }  
  
 private void setupUI() {  
 configureMainFrame();  
 JPanel mainPanel = createMainPanel();  
 add(mainPanel);  
 }  
  
 private void configureMainFrame() {  
 setTitle("🔌 Solar Inverter Management");  
 setSize(600, 500);  
 setLocationRelativeTo(null);  
 setDefaultCloseOperation(*DISPOSE\_ON\_CLOSE*);  
 }  
  
 private JPanel createMainPanel() {  
 JPanel panel = new JPanel(new BorderLayout(15, 15));  
 panel.setBackground(new Color(245, 248, 255));  
 panel.setBorder(BorderFactory.*createEmptyBorder*(25, 25, 25, 25));  
  
 panel.add(createTitleLabel(), BorderLayout.*NORTH*);  
 panel.add(createTableScrollPane(), BorderLayout.*CENTER*);  
 panel.add(createInputPanel(), BorderLayout.*SOUTH*);  
  
 return panel;  
 }  
  
 private JLabel createTitleLabel() {  
 JLabel title = new JLabel("Solar Inverter Inventory", SwingConstants.*CENTER*);  
 title.setFont(new Font("Segoe UI", Font.*BOLD*, 24));  
 title.setForeground(new Color(50, 50, 80));  
 title.setBorder(BorderFactory.*createEmptyBorder*(0, 0, 20, 0));  
 return title;  
 }  
  
 private JScrollPane createTableScrollPane() {  
 JTable table = new JTable(tableModel);  
 styleTable(table);  
 return new JScrollPane(table);  
 }  
  
 private void styleTable(JTable table) {  
 table.setFont(new Font("Segoe UI", Font.*PLAIN*, 14));  
 table.setRowHeight(30);  
 table.setSelectionBackground(new Color(200, 220, 255));  
 table.setSelectionForeground(Color.*BLACK*);  
 table.setGridColor(new Color(220, 220, 220));  
 table.setBorder(BorderFactory.*createLineBorder*(new Color(200, 210, 230)));  
 }  
  
 private DefaultTableModel createTableModel() {  
 String[] columns = {"Company", "Price"};  
 return new DefaultTableModel(columns, 0) {  
 @Override  
 public boolean isCellEditable(int row, int column) {  
 return false;  
 }  
 };  
 }  
  
 private JPanel createInputPanel() {  
 JPanel panel = new JPanel(new GridBagLayout());  
 panel.setBackground(new Color(245, 248, 255));  
  
 GridBagConstraints gbc = new GridBagConstraints();  
 gbc.insets = new Insets(5, 5, 5, 5);  
 gbc.anchor = GridBagConstraints.*WEST*;  
  
 addInputComponents(panel, gbc);  
 panel.add(createAddButton(), gbc);  
  
 return panel;  
 }  
  
 private void addInputComponents(JPanel panel, GridBagConstraints gbc) {  
 gbc.gridx = 0;  
 gbc.gridy = 0;  
 panel.add(createLabel("Company:"), gbc);  
  
 gbc.gridx = 1;  
 panel.add(nameField, gbc);  
  
 gbc.gridx = 0;  
 gbc.gridy = 1;  
 panel.add(createLabel("Price:"), gbc);  
  
 gbc.gridx = 1;  
 panel.add(priceField, gbc);  
  
 gbc.gridx = 0;  
 gbc.gridy = 2;  
 gbc.gridwidth = 2;  
 gbc.anchor = GridBagConstraints.*CENTER*;  
 }  
  
 private JLabel createLabel(String text) {  
 JLabel label = new JLabel(text);  
 label.setFont(new Font("Segoe UI", Font.*PLAIN*, 14));  
 return label;  
 }  
  
 private JTextField createTextField(int columns) {  
 JTextField field = new JTextField(columns);  
 field.setFont(new Font("Segoe UI", Font.*PLAIN*, 14));  
 field.setBorder(BorderFactory.*createCompoundBorder*(  
 BorderFactory.*createLineBorder*(new Color(200, 210, 230)),  
 BorderFactory.*createEmptyBorder*(5, 8, 5, 8)  
 ));  
 return field;  
 }  
  
 private JButton createAddButton() {  
 JButton button = new JButton("Add Inverter");  
 button.setFont(new Font("Segoe UI", Font.*BOLD*, 14));  
 button.setBackground(new Color(70, 130, 180));  
 button.setForeground(Color.*WHITE*);  
 button.setFocusPainted(false);  
 button.setBorder(BorderFactory.*createEmptyBorder*(8, 20, 8, 20));  
  
 button.addMouseListener(new java.awt.event.MouseAdapter() {  
 public void mouseEntered(java.awt.event.MouseEvent evt) {  
 button.setBackground(new Color(90, 150, 200));  
 }  
 public void mouseExited(java.awt.event.MouseEvent evt) {  
 button.setBackground(new Color(70, 130, 180));  
 }  
 });  
  
 button.addActionListener(e -> handleAddInverter());  
 return button;  
 }  
  
 private void handleAddInverter() {  
 String name = nameField.getText().trim();  
 String price = priceField.getText().trim();  
  
 if (validateInput(name, price)) {  
 if (addInverterToDatabase(name, price)) {  
 addInverterToTable(name, price);  
 clearInputFields();  
 }  
 }  
 }  
  
 private boolean validateInput(String name, String price) {  
 if (name.isEmpty() || price.isEmpty()) {  
 JOptionPane.*showMessageDialog*(this,  
 "Please enter both company and price.",  
 "Input Error",  
 JOptionPane.*WARNING\_MESSAGE*);  
 return false;  
 }  
 return true;  
 }  
  
 private void clearInputFields() {  
 nameField.setText("");  
 priceField.setText("");  
 }  
  
 private boolean addInverterToDatabase(String name, String price) {  
 if (!database.isConnected()) {  
 showDatabaseWarning();  
 return true;  
 }  
  
 String sql = "INSERT INTO inverter (Company, Price) VALUES (?, ?)";  
  
 try (PreparedStatement pstmt = database.getConnection().prepareStatement(sql)) {  
 pstmt.setString(1, name);  
 pstmt.setString(2, price);  
 pstmt.executeUpdate();  
 return true;  
 } catch (SQLException ex) {  
 showDatabaseError("Error saving to database: " + ex.getMessage());  
 return false;  
 }  
 }  
  
 private void showDatabaseWarning() {  
 JOptionPane.*showMessageDialog*(this,  
 "Not connected to database. Data will not be saved.",  
 "Database Error",  
 JOptionPane.*WARNING\_MESSAGE*);  
 }  
  
 private void showDatabaseError(String message) {  
 JOptionPane.*showMessageDialog*(this,  
 message,  
 "Database Error",  
 JOptionPane.*ERROR\_MESSAGE*);  
 }  
  
 private void addInverterToTable(String name, String price) {  
 tableModel.addRow(new Object[]{name, price});  
 }  
  
 private void loadExistingInverters() {  
 if (!database.isConnected()) return;  
  
 String sql = "SELECT Company, Price FROM inverter";  
  
 try (Statement stmt = database.getConnection().createStatement();  
 ResultSet rs = stmt.executeQuery(sql)) {  
  
 while (rs.next()) {  
 tableModel.addRow(new Object[]{  
 rs.getString("Company"),  
 rs.getString("Price")  
 });  
 }  
 } catch (SQLException ex) {  
 showDatabaseError("Error loading existing inverters: " + ex.getMessage());  
 }  
 }  
  
 @Override  
 public void dispose() {  
 if (database != null) {  
 database.close();  
 }  
 super.dispose();  
 }  
}

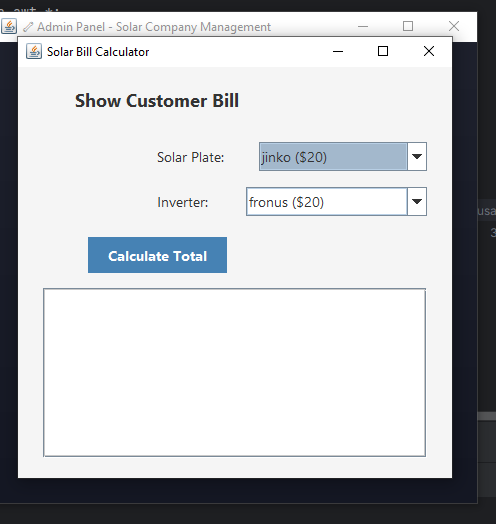
Output:



Show customer bill.java:

import javax.swing.\*;  
import java.awt.\*;  
import java.sql.\*;  
import java.util.HashMap;  
  
public class ShowCustomerBill extends JFrame {  
 private JComboBox<String> plateComboBox, inverterComboBox;  
 private JTextArea billArea;  
 private JButton calculateButton;  
  
 private HashMap<String, Integer> platePrices = new HashMap<>();  
 private HashMap<String, Integer> inverterPrices = new HashMap<>();  
  
 public ShowCustomerBill() {  
 setTitle("Solar Bill Calculator");  
 setSize(450, 450);  
 setLocationRelativeTo(null);  
 setDefaultCloseOperation(JFrame.*DISPOSE\_ON\_CLOSE*);  
  
 JPanel mainPanel = new JPanel();  
 mainPanel.setLayout(new BoxLayout(mainPanel, BoxLayout.*Y\_AXIS*));  
 mainPanel.setBorder(BorderFactory.*createEmptyBorder*(20, 25, 20, 25));  
 mainPanel.setBackground(new Color(245, 245, 245));  
  
 JLabel titleLabel = new JLabel("Show Customer Bill");  
 titleLabel.setFont(new Font("Segoe UI", Font.*BOLD*, 18));  
 titleLabel.setAlignmentX(Component.*CENTER\_ALIGNMENT*);  
 titleLabel.setBorder(BorderFactory.*createEmptyBorder*(0, 0, 15, 0));  
  
 plateComboBox = new JComboBox<>();  
 inverterComboBox = new JComboBox<>();  
  
 JPanel platePanel = createProductPanel("Solar Plate:", plateComboBox);  
 JPanel inverterPanel = createProductPanel("Inverter:", inverterComboBox);  
  
 billArea = new JTextArea();  
 billArea.setEditable(false);  
 billArea.setFont(new Font("Segoe UI", Font.*PLAIN*, 14));  
 billArea.setBackground(Color.*WHITE*);  
 billArea.setBorder(BorderFactory.*createCompoundBorder*(  
 BorderFactory.*createLineBorder*(new Color(200, 200, 200)),  
 BorderFactory.*createEmptyBorder*(8, 8, 8, 8)));  
 JScrollPane scrollPane = new JScrollPane(billArea);  
 scrollPane.setPreferredSize(new Dimension(350, 150));  
  
 calculateButton = new JButton("Calculate Total");  
 styleButton(calculateButton, new Color(70, 130, 180));  
 calculateButton.addActionListener(e -> calculateAndSaveBill());  
  
 mainPanel.add(titleLabel);  
 mainPanel.add(Box.*createRigidArea*(new Dimension(0, 15)));  
 mainPanel.add(platePanel);  
 mainPanel.add(Box.*createRigidArea*(new Dimension(0, 15)));  
 mainPanel.add(inverterPanel);  
 mainPanel.add(Box.*createRigidArea*(new Dimension(0, 20)));  
 mainPanel.add(calculateButton);  
 mainPanel.add(Box.*createRigidArea*(new Dimension(0, 15)));  
 mainPanel.add(scrollPane);  
  
 add(mainPanel);  
  
 loadProductsFromDatabase(); // Auto-load on startup  
 }  
  
 private void loadProductsFromDatabase() {  
 platePrices.clear();  
 inverterPrices.clear();  
 plateComboBox.removeAllItems();  
 inverterComboBox.removeAllItems();  
  
 loadProductData("plates", plateComboBox, platePrices);  
 loadProductData("inverter", inverterComboBox, inverterPrices);  
  
 if (plateComboBox.getItemCount() > 0 && inverterComboBox.getItemCount() > 0) {  
 billArea.setText("");  
 } else {  
 billArea.setText("Error loading product data.");  
 }  
 }  
  
 private void loadProductData(String tableName, JComboBox<String> comboBox, HashMap<String, Integer> priceMap) {  
 Database db = new Database();  
 try {  
 ResultSet rs = db.executeQuery("SELECT Company, Price FROM " + tableName);  
 while (rs.next()) {  
 String name = rs.getString("Company");  
 int price = rs.getInt("Price");  
 String displayName = name + " ($" + price + ")";  
 comboBox.addItem(displayName);  
 priceMap.put(displayName, price);  
 }  
 } catch (SQLException e) {  
 JOptionPane.*showMessageDialog*(this,  
 "Error loading data from table '" + tableName + "': " + e.getMessage(),  
 "Database Error",  
 JOptionPane.*ERROR\_MESSAGE*);  
 } finally {  
 db.close();  
 }  
 }  
  
 private void calculateAndSaveBill() {  
 String selectedPlate = (String) plateComboBox.getSelectedItem();  
 String selectedInverter = (String) inverterComboBox.getSelectedItem();  
  
 if (selectedPlate == null || selectedInverter == null) {  
 billArea.setText("Please select both a plate and an inverter.");  
 return;  
 }  
  
 int platePrice = platePrices.getOrDefault(selectedPlate, 0);  
 int inverterPrice = inverterPrices.getOrDefault(selectedInverter, 0);  
 int total = platePrice + inverterPrice;  
 String customerIdStr = JOptionPane.*showInputDialog*(this, "Enter Customer ID:");  
 if (customerIdStr == null || customerIdStr.trim().isEmpty()) return;  
  
 String customerName = JOptionPane.*showInputDialog*(this, "Enter Customer Name:");  
 if (customerName == null || customerName.trim().isEmpty()) return;  
  
 try {  
 int customerId = Integer.*parseInt*(customerIdStr.trim());  
  
 // Save to database  
 Database db = new Database();  
 String query = "INSERT INTO savebill (Name, id, plate, plateprice, inverter, inverterprice, total) VALUES (?, ?, ?, ?, ?, ?, ?)";  
 PreparedStatement pst = db.getConnection().prepareStatement(query);  
 pst.setString(1, customerName);  
 pst.setInt(2, customerId);  
 pst.setString(3, selectedPlate);  
 pst.setInt(4, platePrice);  
 pst.setString(5, selectedInverter);  
 pst.setInt(6, inverterPrice);  
 pst.setInt(7, total);  
 pst.executeUpdate();  
 db.close();  
  
 billArea.setText("Selected items:\n" +  
 "Plate: " + selectedPlate + "\n" +  
 "Inverter: " + selectedInverter + "\n\n" +  
 "Customer ID: " + customerId + "\n" +  
 "Customer Name: " + customerName + "\n" +  
 "Total Bill: $" + total + "\n\n" +  
 "✅ Bill saved to database!");  
  
 } catch (NumberFormatException e) {  
 JOptionPane.*showMessageDialog*(this, "Invalid ID. Please enter a valid number.");  
 } catch (SQLException e) {  
 JOptionPane.*showMessageDialog*(this, "Error saving bill: " + e.getMessage(), "Database Error", JOptionPane.*ERROR\_MESSAGE*);  
 }  
 }  
 private JPanel createProductPanel(String labelText, JComboBox<String> comboBox) {  
 JPanel panel = new JPanel();  
 panel.setLayout(new BoxLayout(panel, BoxLayout.*X\_AXIS*));  
 panel.setBackground(new Color(245, 245, 245));  
 panel.setAlignmentX(Component.*LEFT\_ALIGNMENT*);  
 JLabel label = new JLabel(labelText);  
 label.setFont(new Font("Segoe UI", Font.*PLAIN*, 14));  
 label.setPreferredSize(new Dimension(100, 30));  
 comboBox.setFont(new Font("Segoe UI", Font.*PLAIN*, 14));  
 comboBox.setMaximumSize(new Dimension(250, 30));  
 comboBox.setBackground(Color.*white*);  
 panel.add(label);  
 panel.add(Box.*createRigidArea*(new Dimension(15, 0)));  
 panel.add(comboBox);  
  
 return panel;  
 }  
  
 private void styleButton(JButton button, Color bgColor) {  
 button.setFont(new Font("Segoe UI", Font.*BOLD*, 14));  
 button.setBackground(bgColor);  
 button.setForeground(Color.*WHITE*);  
 button.setFocusPainted(false);  
 button.setBorder(BorderFactory.*createEmptyBorder*(8, 20, 8, 20));  
 button.setAlignmentX(Component.*CENTER\_ALIGNMENT*);  
 button.setCursor(new Cursor(Cursor.*HAND\_CURSOR*));  
 }  
}

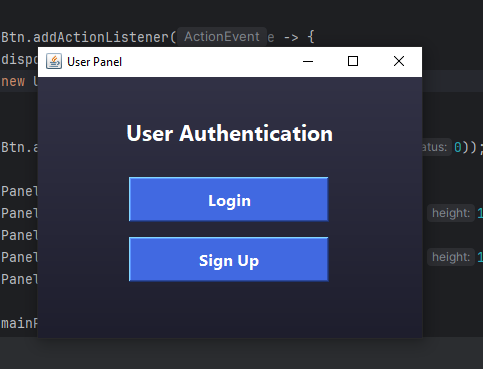
Output:

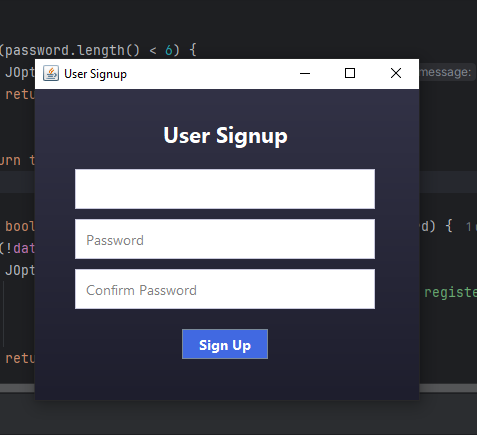


User signup:

import javax.swing.\*;  
import java.awt.\*;  
import java.sql.\*;  
  
public class UserSignup extends JFrame {  
 private JFrame parent;  
 private Database database;  
  
 public UserSignup(JFrame parent) {  
 this.parent = parent;  
 this.database = new Database();  
  
 if (!database.isConnected()) {  
 JOptionPane.*showMessageDialog*(this,  
 "Database connection failed. Please try again later.",  
 "Warning",  
 JOptionPane.*WARNING\_MESSAGE*);  
 }  
  
 setupUI();  
 }  
  
 private void setupUI() {  
 setTitle("User Signup");  
 setSize(400, 350);  
 setDefaultCloseOperation(*DISPOSE\_ON\_CLOSE*);  
 setLocationRelativeTo(null);  
  
 JPanel panel = new JPanel() {  
 @Override  
 protected void paintComponent(Graphics g) {  
 super.paintComponent(g);  
 Graphics2D g2d = (Graphics2D) g;  
 Color color1 = new Color(50, 50, 70);  
 Color color2 = new Color(30, 30, 45);  
 GradientPaint gp = new GradientPaint(0, 0, color1, 0, getHeight(), color2);  
 g2d.setPaint(gp);  
 g2d.fillRect(0, 0, getWidth(), getHeight());  
 }  
 };  
  
 panel.setLayout(new BoxLayout(panel, BoxLayout.*Y\_AXIS*));  
 panel.setBorder(BorderFactory.*createEmptyBorder*(30, 40, 30, 40));  
  
 JLabel titleLabel = new JLabel("User Signup", SwingConstants.*CENTER*);  
 titleLabel.setFont(new Font("Segoe UI", Font.*BOLD*, 22));  
 titleLabel.setForeground(Color.*WHITE*);  
 titleLabel.setAlignmentX(Component.*CENTER\_ALIGNMENT*);  
 panel.add(titleLabel);  
 panel.add(Box.*createRigidArea*(new Dimension(0, 20)));  
  
 // Create text fields with placeholder text  
 JTextField usernameField = createTextFieldWithPlaceholder("Username");  
 JPasswordField passwordField = createPasswordFieldWithPlaceholder("Password");  
 JPasswordField confirmPasswordField = createPasswordFieldWithPlaceholder("Confirm Password");  
  
 panel.add(usernameField);  
 panel.add(Box.*createRigidArea*(new Dimension(0, 10)));  
 panel.add(passwordField);  
 panel.add(Box.*createRigidArea*(new Dimension(0, 10)));  
 panel.add(confirmPasswordField);  
 panel.add(Box.*createRigidArea*(new Dimension(0, 20)));  
  
 JButton signupBtn = new JButton("Sign Up");  
 signupBtn.setAlignmentX(Component.*CENTER\_ALIGNMENT*);  
 signupBtn.setPreferredSize(new Dimension(100, 40));  
 signupBtn.setFont(new Font("Segoe UI", Font.*BOLD*, 14));  
 signupBtn.setFocusPainted(false);  
 signupBtn.setBackground(new Color(65, 105, 225));  
 signupBtn.setForeground(Color.*WHITE*);  
 signupBtn.setCursor(new Cursor(Cursor.*HAND\_CURSOR*));  
  
 signupBtn.addActionListener(e -> {  
 String username = usernameField.getText().trim();  
 String password = new String(passwordField.getPassword()).trim();  
 String confirmPassword = new String(confirmPasswordField.getPassword()).trim();  
  
 if (validateSignup(username, password, confirmPassword)) {  
 if (registerUser(username, password)) {  
 JOptionPane.*showMessageDialog*(this, "Registration successful!", "Success", JOptionPane.*INFORMATION\_MESSAGE*);  
 dispose();  
 if (parent != null) parent.dispose();  
 }  
 }  
 });  
  
 panel.add(signupBtn);  
 add(panel);  
 }  
  
 private JTextField createTextFieldWithPlaceholder(String placeholder) {  
 JTextField textField = new JTextField();  
 textField.setMaximumSize(new Dimension(300, 40));  
 textField.setFont(new Font("Segoe UI", Font.*PLAIN*, 14));  
 textField.setBorder(BorderFactory.*createCompoundBorder*(  
 BorderFactory.*createLineBorder*(new Color(200, 200, 220)),  
 BorderFactory.*createEmptyBorder*(5, 10, 5, 10)));  
 textField.setForeground(Color.*GRAY*);  
 textField.setText(placeholder);  
 textField.addFocusListener(new java.awt.event.FocusAdapter() {  
 public void focusGained(java.awt.event.FocusEvent evt) {  
 if (textField.getText().equals(placeholder)) {  
 textField.setText("");  
 textField.setForeground(Color.*BLACK*);  
 }  
 }  
 public void focusLost(java.awt.event.FocusEvent evt) {  
 if (textField.getText().isEmpty()) {  
 textField.setForeground(Color.*GRAY*);  
 textField.setText(placeholder);  
 }  
 }  
 });  
 return textField;  
 }  
  
 private JPasswordField createPasswordFieldWithPlaceholder(String placeholder) {  
 JPasswordField passwordField = new JPasswordField();  
 passwordField.setMaximumSize(new Dimension(300, 40));  
 passwordField.setFont(new Font("Segoe UI", Font.*PLAIN*, 14));  
 passwordField.setBorder(BorderFactory.*createCompoundBorder*(  
 BorderFactory.*createLineBorder*(new Color(200, 200, 220)),  
 BorderFactory.*createEmptyBorder*(5, 10, 5, 10)));  
 passwordField.setForeground(Color.*GRAY*);  
 passwordField.setEchoChar((char)0); // Show plain text for placeholder  
 passwordField.setText(placeholder);  
 passwordField.addFocusListener(new java.awt.event.FocusAdapter() {  
 public void focusGained(java.awt.event.FocusEvent evt) {  
 if (String.*valueOf*(passwordField.getPassword()).equals(placeholder)) {  
 passwordField.setText("");  
 passwordField.setForeground(Color.*BLACK*);  
 passwordField.setEchoChar('•'); // Show password characters  
 }  
 }  
 public void focusLost(java.awt.event.FocusEvent evt) {  
 if (passwordField.getPassword().length == 0) {  
 passwordField.setForeground(Color.*GRAY*);  
 passwordField.setEchoChar((char)0);  
 passwordField.setText(placeholder);  
 }  
 }  
 });  
 return passwordField;  
 }  
  
 // Rest of the methods remain the same (validateSignup, registerUser, dispose)  
 private boolean validateSignup(String username, String password, String confirmPassword) {  
 if (username.isEmpty() || username.equals("Username") || password.isEmpty() || password.equals("Password")) {  
 JOptionPane.*showMessageDialog*(this, "Username and password cannot be empty!", "Error", JOptionPane.*ERROR\_MESSAGE*);  
 return false;  
 }  
  
 if (!password.equals(confirmPassword)) {  
 JOptionPane.*showMessageDialog*(this, "Passwords do not match!", "Error", JOptionPane.*ERROR\_MESSAGE*);  
 return false;  
 }  
  
 if (password.length() < 6) {  
 JOptionPane.*showMessageDialog*(this, "Password must be at least 6 characters!", "Error", JOptionPane.*ERROR\_MESSAGE*);  
 return false;  
 }  
  
 return true;  
 }  
  
 private boolean registerUser(String username, String password) {  
 if (!database.isConnected()) {  
 JOptionPane.*showMessageDialog*(this,  
 "Database connection failed. Cannot register user.",  
 "Error",  
 JOptionPane.*ERROR\_MESSAGE*);  
 return false;  
 }  
  
 String checkSql = "SELECT \* FROM usersignup WHERE username = ?";  
 String insertSql = "INSERT INTO usersignup (username, password) VALUES (?, ?)";  
  
 try {  
 try (PreparedStatement checkStmt = database.getConnection().prepareStatement(checkSql)) {  
 checkStmt.setString(1, username);  
 try (ResultSet rs = checkStmt.executeQuery()) {  
 if (rs.next()) {  
 JOptionPane.*showMessageDialog*(this,  
 "Username already exists!",  
 "Error",  
 JOptionPane.*ERROR\_MESSAGE*);  
 return false;  
 }  
 }  
 }  
  
 try (PreparedStatement insertStmt = database.getConnection().prepareStatement(insertSql)) {  
 insertStmt.setString(1, username);  
 insertStmt.setString(2, password);  
 int affectedRows = insertStmt.executeUpdate();  
 return affectedRows > 0;  
 }  
 } catch (SQLException ex) {  
 JOptionPane.*showMessageDialog*(this,  
 "Error during registration: " + ex.getMessage(),  
 "Database Error",  
 JOptionPane.*ERROR\_MESSAGE*);  
 return false;  
 }  
 }  
  
 @Override  
 public void dispose() {  
 if (database != null) {  
 database.close();  
 }  
 super.dispose();  
 }  
}

Output:





Database:

import java.sql.\*;  
import javax.swing.\*;  
  
public class Database {  
 private static final String *URL* = "jdbc:mysql://localhost:3306/solar";  
 private static final String *USERNAME* = "root";  
 private static final String *PASSWORD* = "";  
  
 private Connection connection;  
 private boolean isConnected = false;  
  
 public Database() {  
 initializeConnection();  
 }  
  
 private void initializeConnection() {  
 try {  
 Class.*forName*("com.mysql.cj.jdbc.Driver");  
  
  
 connection = DriverManager.*getConnection*(*URL*, *USERNAME*, *PASSWORD*);  
 isConnected = true;  
  
  
 if (connection.isValid(2)) {  
 System.*out*.println("Database connection established successfully");  
 }  
 } catch (ClassNotFoundException e) {  
 showError("JDBC Driver not found: " + e.getMessage());  
 isConnected = false;  
 } catch (SQLException e) {  
 showError("Connection failed: " + e.getMessage());  
 isConnected = false;  
 }  
 }  
  
 public boolean isConnected() {  
 return isConnected;  
 }  
  
 public Connection getConnection() {  
 if (!isConnected) {  
 showError("No active database connection");  
 return null;  
 }  
 return connection;  
 }  
  
 public void close() {  
 try {  
 if (connection != null && !connection.isClosed()) {  
 connection.close();  
 isConnected = false;  
 System.*out*.println("Database connection closed");  
 }  
 } catch (SQLException e) {  
 showError("Error closing connection: " + e.getMessage());  
 }  
 }  
  
 private void showError(String message) {  
 JOptionPane.*showMessageDialog*(null,  
 message,  
 "Database Error",  
 JOptionPane.*ERROR\_MESSAGE*);  
 }  
  
 public ResultSet executeQuery(String sql) throws SQLException {  
 if (!isConnected) throw new SQLException("No database connection");  
 Statement stmt = connection.createStatement();  
 return stmt.executeQuery(sql);  
 }  
  
  
 public int executeUpdate(String sql) throws SQLException {  
 if (!isConnected) throw new SQLException("No database connection");  
 Statement stmt = connection.createStatement();  
 return stmt.executeUpdate(sql);  
 }  
}

Output:

