Maddison Kiefer

Dr. Schwartz

Advanced Java Programming

10/31/2023

**Project 2-1 Creating a GUI Bank Application**

**Source Code:**

//@author Maddison Kiefer

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

// Creating the BankBalanceApplication class that extends JFrame and implements ActionListener

public class BankBalanceApplication extends JFrame implements ActionListener {

// Declaring the instance variables

private final JTextField balanceField;

private final JButton depositButton;

private final JButton withdrawButton;

private final JButton displayButton;

private final JButton exitButton;

// Initializing the balance to 0

private double accountBalance = 0.0;

// Constructor for the class

public BankBalanceApplication() {

// Sets the title and size for the frame

setTitle("Bank Balance Application");

setSize(500, 200);

// Exits the frame when the 'x' is clicked

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

// Centers the frame on the screen

setLocationRelativeTo(null);

// Creating the panel to hold the components

JPanel panel = new JPanel();

// Creating a text field with the initial balance

balanceField = new JTextField("Balance: $" + accountBalance);

// Makes sure the text field can't be edited

balanceField.setEditable(false);

// Sets the size of the text field

balanceField.setPreferredSize(new Dimension(150, 30));

// Adds the text field to the panel

panel.add(balanceField);

// Creating the needed buttons

depositButton = new JButton("Deposit");

withdrawButton = new JButton("Withdraw");

displayButton = new JButton("Display Balance");

exitButton = new JButton("Exit");

// Adding action listeners to the buttons

depositButton.addActionListener(this);

withdrawButton.addActionListener(this);

displayButton.addActionListener(this);

exitButton.addActionListener(this);

// Adding the buttons to the panel

panel.add(depositButton);

panel.add(withdrawButton);

panel.add(displayButton);

panel.add(exitButton);

// Adding the panel to the frame

add(panel);

// Setting the frame to be visible

setVisible(true);

}

// Implementing the ActionListener

@Override

public void actionPerformed(ActionEvent e) {

// If the deposit button is clicked

if (e.getSource() == depositButton) {

// Asks user for input amount

String depositAmountString = JOptionPane.showInputDialog(this, "Enter deposit amount:");

// Checks if the user hits the cancel button

if (depositAmountString != null) {

try {

// Parsing input as double

double depositAmount = Double.parseDouble(depositAmountString);

// Checks that the amount is positive

if (depositAmount > 0) {

// Adds the amount to the accountBalance

accountBalance += depositAmount;

// Displays the new amount in the text field

balanceField.setText("Balance: $" + accountBalance);

} else {

// Otherwise, a message to enter a valid input will be displayed

JOptionPane.showMessageDialog(this, "Please enter a valid positive amount.");

}

} catch (NumberFormatException ex) {

// Handles inputs that are not numbers

JOptionPane.showMessageDialog(this, "Please enter a valid numeric amount.");

}

}

// If the withdraw button is clicked

} else if (e.getSource() == withdrawButton) {

// Asks user for input amount

String withdrawAmountString = JOptionPane.showInputDialog(this, "Enter withdrawal amount:");

// Checks if the user hits the cancel button

if (withdrawAmountString != null) {

try {

// Parsing input as double

double withdrawAmount = Double.parseDouble(withdrawAmountString);

// Checks that the amount is positive

if (withdrawAmount > 0) {

// Checks if the amount is less than the accountBalance

if (withdrawAmount <= accountBalance) {

// Subtracts input from the balance

accountBalance -= withdrawAmount;

// Displays the new amount in the text field

balanceField.setText("Balance: $" + accountBalance);

} else {

// Displays insufficient funds if the withdraw amount is greater than the balance

JOptionPane.showMessageDialog(this, "Insufficient funds!");

}

} else {

// Handles negative number inputs

JOptionPane.showMessageDialog(this, "Please enter a valid positive amount.");

}

} catch (NumberFormatException ex) {

// Handles inputs that are not numbers

JOptionPane.showMessageDialog(this, "Please enter a valid numeric amount.");

}

}

// If the display button is clicked

} else if (e.getSource() == displayButton) {

// Displays the remaining balance in a dialog box

JOptionPane.showMessageDialog(this, "Remaining Balance: $" + accountBalance);

// If the exit button is clicked

} else if (e.getSource() == exitButton) {

// Displays the remaining balance in a dialog box

JOptionPane.showMessageDialog(this, "Remaining Balance: $" + accountBalance);

// Exits the application

System.exit(0);

}

}

// Main method to start the application

public static void main(String[] args) {

new BankBalanceApplication();

}

}

**Executing the Application:**















