ATM Interface:

Source code:

package intern;

import javax.swing.\*;

import java.awt.\*;

import java.util.ArrayList;

import java.util.List;

/\*\*

\* ATMSimulation class simulates an ATM machine with basic functionalities:

\* Balance Inquiry, Cash Withdrawal, Cash Deposit, PIN Change, and Transaction History.

\*/

public class ATM {

private JFrame frame; // Main frame for the application

private JPanel panel; // Main panel to hold all components

private JTextField amountField; // TextField for entering amounts

private JPasswordField pinField; // PasswordField for entering PIN

private JTextArea displayArea; // TextArea to display messages and transaction history

private double balance = 1000.00; // Initial balance

private String pin = "1234"; // Default PIN

private List<String> transactionHistory = new ArrayList<>(); // List to store transaction history

/\*\*

\* Constructor initializes the GUI components and sets up the ATM simulation.

\*/

public ATM() {

// Create the main frame

frame = new JFrame("ATM Machine");

// Create the main panel with GridBagLayout for flexible component placement

panel = new JPanel(new GridBagLayout());

panel.setBackground(Color.WHITE); // Set panel background color to white

GridBagConstraints gbc = new GridBagConstraints();

gbc.fill = GridBagConstraints.HORIZONTAL;

gbc.insets = new Insets(5, 5, 5, 5);

// Create and configure GUI components with blue and white color palette

amountField = new JTextField(15);

amountField.setBackground(new Color(173, 216, 230)); // Light blue background

amountField.setForeground(Color.BLACK);

pinField = new JPasswordField(15);

pinField.setBackground(new Color(173, 216, 230)); // Light blue background

pinField.setForeground(Color.BLACK);

displayArea = new JTextArea(10, 30);

displayArea.setEditable(false);

displayArea.setBackground(new Color(240, 248, 255)); // Alice blue background

displayArea.setForeground(Color.BLACK);

displayArea.setBorder(BorderFactory.createLineBorder(Color.GRAY));

// Create buttons for ATM functions

JButton balanceInquiryButton = new JButton("Balance Inquiry");

JButton cashWithdrawalButton = new JButton("Cash Withdrawal");

JButton cashDepositButton = new JButton("Cash Deposit");

JButton pinChangeButton = new JButton("Change PIN");

JButton transactionHistoryButton = new JButton("Transaction History");

// Set button colors

balanceInquiryButton.setBackground(new Color(70, 130, 180)); // Steel blue background

balanceInquiryButton.setForeground(Color.WHITE);

cashWithdrawalButton.setBackground(new Color(70, 130, 180)); // Steel blue background

cashWithdrawalButton.setForeground(Color.WHITE);

cashDepositButton.setBackground(new Color(70, 130, 180)); // Steel blue background

cashDepositButton.setForeground(Color.WHITE);

pinChangeButton.setBackground(new Color(70, 130, 180)); // Steel blue background

pinChangeButton.setForeground(Color.WHITE);

transactionHistoryButton.setBackground(new Color(70, 130, 180)); // Steel blue background

transactionHistoryButton.setForeground(Color.WHITE);

// Set label colors

JLabel amountLabel = new JLabel("Enter Amount:");

amountLabel.setForeground(new Color(70, 130, 180)); // Steel blue foreground

JLabel pinLabel = new JLabel("Enter PIN:");

pinLabel.setForeground(new Color(70, 130, 180)); // Steel blue foreground

// Add components to the panel with proper constraints

gbc.gridx = 0; gbc.gridy = 0;

panel.add(amountLabel, gbc);

gbc.gridx = 1; gbc.gridy = 0;

panel.add(amountField, gbc);

gbc.gridx = 0; gbc.gridy = 1;

panel.add(pinLabel, gbc);

gbc.gridx = 1; gbc.gridy = 1;

panel.add(pinField, gbc);

gbc.gridx = 0; gbc.gridy = 2;

gbc.gridwidth = 2;

panel.add(balanceInquiryButton, gbc);

gbc.gridx = 0; gbc.gridy = 3;

panel.add(cashWithdrawalButton, gbc);

gbc.gridx = 0; gbc.gridy = 4;

panel.add(cashDepositButton, gbc);

gbc.gridx = 0; gbc.gridy = 5;

panel.add(pinChangeButton, gbc);

gbc.gridx = 0; gbc.gridy = 6;

panel.add(transactionHistoryButton, gbc);

gbc.gridx = 0; gbc.gridy = 7;

gbc.gridwidth = 2;

panel.add(new JScrollPane(displayArea), gbc);

// Add action listeners for buttons using lambda expressions

balanceInquiryButton.addActionListener(e -> balanceInquiry());

cashWithdrawalButton.addActionListener(e -> cashWithdrawal());

cashDepositButton.addActionListener(e -> cashDeposit());

pinChangeButton.addActionListener(e -> changePIN());

transactionHistoryButton.addActionListener(e -> showTransactionHistory());

// Set up the frame

frame.add(panel);

frame.pack();

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setVisible(true);

}

/\*\*

\* Method to handle balance inquiry.

\* Displays the current balance if the PIN is validated.

\*/

private void balanceInquiry() {

if (validatePIN()) {

displayArea.setText("Current Balance: $" + balance);

transactionHistory.add("Balance Inquiry: $" + balance);

} else {

displayArea.setText("Invalid PIN. Try again.");

}

}

/\*\*

\* Method to handle cash withdrawal.

\* Deducts the specified amount from the balance if the PIN is validated and balance is sufficient.

\*/

private void cashWithdrawal() {

if (validatePIN()) {

double amount = Double.parseDouble(amountField.getText());

if (amount > balance) {

displayArea.setText("Insufficient balance.");

} else {

balance -= amount;

displayArea.setText("Withdrawal Successful. New Balance: $" + balance);

transactionHistory.add("Withdrawal: -$" + amount + ", New Balance: $" + balance);

}

} else {

displayArea.setText("Invalid PIN. Try again.");

}

}

/\*\*

\* Method to handle cash deposit.

\* Adds the specified amount to the balance if the PIN is validated.

\*/

private void cashDeposit() {

if (validatePIN()) {

double amount = Double.parseDouble(amountField.getText());

balance += amount;

displayArea.setText("Deposit Successful. New Balance: $" + balance);

transactionHistory.add("Deposit: +$" + amount + ", New Balance: $" + balance);

} else {

displayArea.setText("Invalid PIN. Try again.");

}

}

/\*\*

\* Method to handle PIN change.

\* Changes the current PIN to the new one entered.

\*/

private void changePIN() {

String newPIN = new String(pinField.getPassword());

pin = newPIN;

displayArea.setText("PIN changed successfully.");

transactionHistory.add("PIN Change");

}

/\*\*

\* Method to display transaction history.

\* Shows the list of all transactions if the PIN is validated.

\*/

private void showTransactionHistory() {

if (validatePIN()) {

StringBuilder history = new StringBuilder();

for (String transaction : transactionHistory) {

history.append(transaction).append("\n");

}

displayArea.setText(history.toString());

} else {

displayArea.setText("Invalid PIN. Try again.");

}

}

/\*\*

\* Method to validate the entered PIN.

\* @return true if the entered PIN matches the current PIN, false otherwise.

\*/

private boolean validatePIN() {

return new String(pinField.getPassword()).equals(pin);

}

/\*\*

\* Main method to run the ATM simulation application.

\* @param args Command line arguments

\*/

public static void main(String[] args) {

SwingUtilities.invokeLater(ATM::new);

}

}