import java.awt.\*;

import java.awt.event.\*;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

class Account {

private String accountNumber;

private String accountHolder;

private double balance;

private List<String> transactionHistory = new ArrayList<>();

public Account(String accountNumber, String accountHolder, double balance) {

this.accountNumber = accountNumber;

this.accountHolder = accountHolder;

this.balance = balance;

transactionHistory.add("Account created with initial balance: $" + balance);

}

public void deposit(double amount) {

balance += amount;

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

}

public void withdraw(double amount) throws Exception {

if (amount > balance) {

throw new Exception("Insufficient Balance!");

}

balance -= amount;

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

}

public double getBalance() {

return balance;

}

public void viewTransactionHistory(TextArea textArea) {

textArea.setText(""); // Clear previous text

textArea.append("--- Transaction History for Account: " + accountNumber + " ---\n");

for (String record : transactionHistory) {

textArea.append(record + "\n");

}

textArea.append("--------------------------------------------\n");

}

}

public class BankingAppAWT extends Frame {

private static Map<String, Account> accounts = new HashMap<>();

private TextField accountNumberField, amountField, nameField;

private TextArea transactionHistoryArea;

private Label balanceLabel;

public BankingAppAWT() {

setTitle("Banking App");

setSize(600, 400);

setLayout(new FlowLayout());

setResizable(false);

// Initialize components

accountNumberField = new TextField(15);

amountField = new TextField(10);

nameField = new TextField(15);

transactionHistoryArea = new TextArea(10, 40);

transactionHistoryArea.setEditable(false);

balanceLabel = new Label("Balance: $0.0");

Button createButton = new Button("Create Account");

Button depositButton = new Button("Deposit");

Button withdrawButton = new Button("Withdraw");

Button checkBalanceButton = new Button("Check Balance");

Button viewHistoryButton = new Button("View Transaction History");

// Add components to the frame

add(new Label("Account Number:"));

add(accountNumberField);

add(new Label("Account Holder Name:"));

add(nameField);

add(new Label("Amount:"));

add(amountField);

add(createButton);

add(depositButton);

add(withdrawButton);

add(checkBalanceButton);

add(viewHistoryButton);

add(balanceLabel);

add(transactionHistoryArea);

// Add button actions

createButton.addActionListener(e -> createAccount());

depositButton.addActionListener(e -> deposit());

withdrawButton.addActionListener(e -> withdraw());

checkBalanceButton.addActionListener(e -> checkBalance());

viewHistoryButton.addActionListener(e -> viewTransactionHistory());

// Some sample accounts for testing

accounts.put("1001", new Account("1001", "John Doe", 500.0));

accounts.put("1002", new Account("1002", "Jane Smith", 1000.0));

// Window close event

addWindowListener(new WindowAdapter() {

public void windowClosing(WindowEvent we) {

dispose();

}

});

setVisible(true);

}

private void createAccount() {

String accountNumber = accountNumberField.getText();

String accountHolder = nameField.getText();

try {

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

if (accounts.containsKey(accountNumber)) {

showMessage("Account number already exists!");

return;

}

Account account = new Account(accountNumber, accountHolder, balance);

accounts.put(accountNumber, account);

showMessage("Account created successfully!");

clearFields();

} catch (NumberFormatException e) {

showMessage("Invalid initial balance! Please enter a valid number.");

}

}

private void deposit() {

String accountNumber = accountNumberField.getText();

Account account = accounts.get(accountNumber);

if (account != null) {

try {

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

if (depositAmount <= 0) {

showMessage("Deposit amount must be greater than zero!");

return;

}

account.deposit(depositAmount);

showMessage("Deposit successful!");

balanceLabel.setText("Balance: $" + account.getBalance());

} catch (NumberFormatException e) {

showMessage("Invalid deposit amount! Please enter a valid number.");

}

} else {

showMessage("Account not found!");

}

}

private void withdraw() {

String accountNumber = accountNumberField.getText();

Account account = accounts.get(accountNumber);

if (account != null) {

try {

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

if (withdrawalAmount <= 0) {

showMessage("Withdrawal amount must be greater than zero!");

return;

}

account.withdraw(withdrawalAmount);

showMessage("Withdrawal successful!");

balanceLabel.setText("Balance: $" + account.getBalance());

} catch (Exception e) {

showMessage("Error: " + e.getMessage());

}

} else {

showMessage("Account not found!");

}

}

private void checkBalance() {

String accountNumber = accountNumberField.getText();

Account account = accounts.get(accountNumber);

if (account != null) {

balanceLabel.setText("Balance: $" + account.getBalance());

} else {

showMessage("Account not found!");

}

}

private void viewTransactionHistory() {

String accountNumber = accountNumberField.getText();

Account account = accounts.get(accountNumber);

if (account != null) {

account.viewTransactionHistory(transactionHistoryArea);

} else {

showMessage("Account not found!");

}

}

private void showMessage(String message) {

Dialog dialog = new Dialog(this, "Message", true);

dialog.setLayout(new FlowLayout());

dialog.add(new Label(message));

Button okButton = new Button("OK");

okButton.addActionListener(e -> dialog.dispose());

dialog.add(okButton);

dialog.setSize(300, 150);

dialog.setVisible(true);

}

private void clearFields() {

accountNumberField.setText("");

nameField.setText("");

amountField.setText("");

}

public static void main(String[] args) {

new BankingAppAWT();

}

}