import java.util.HashMap;

import java.util.Scanner;

class BankAccount {

private HashMap<String, Double> accounts;

public BankAccount() {

this.accounts = new HashMap<>();

}

public void createAccount(String accountNumber, double initialBalance) {

if (!accounts.containsKey(accountNumber)) {

accounts.put(accountNumber, initialBalance);

System.out.println("Account created successfully.");

} else {

System.out.println("your account is Already found in the bank");

}

}

public double getBalance(String accountNumber) {

return accounts.getOrDefault(accountNumber, 0.0);

}

public void deposit(String accountNumber, double amount) {

if (accounts.containsKey(accountNumber)) {

double currentBalance = accounts.get(accountNumber);

accounts.put(accountNumber, currentBalance + amount);

System.out.println("Deposit successful.");

} else {

System.out.println("Account not found.");

}

}

public void withdraw(String accountNumber, double amount) {

if (accounts.containsKey(accountNumber)) {

double currentBalance = accounts.get(accountNumber);

if (amount <= currentBalance) {

accounts.put(accountNumber, currentBalance - amount);

System.out.println("Withdrawal successfully.");

} else {

System.out.println("Insufficient balance.");

}

} else {

System.out.println("Account not found.");

}

}

}

class BankView {

public void displayAccountDetails(String accountNumber, double balance) {

System.out.println("Account Number: " + accountNumber);

System.out.println("Balance: $" + balance);

}

}

class BankController {

private BankAccount model;

private BankView view;

public BankController(BankAccount model, BankView view) {

this.model = model;

this.view = view;

}

public void createAccount(String accountNumber, double initialBalance) {

model.createAccount(accountNumber, initialBalance);

updateView(accountNumber);

}

public void deposit(String accountNumber, double amount) {

model.deposit(accountNumber, amount);

updateView(accountNumber);

}

public void withdraw(String accountNumber, double amount) {

model.withdraw(accountNumber, amount);

updateView(accountNumber);

}

public void checkBalance(String accountNumber) {

double balance = model.getBalance(accountNumber);

view.displayAccountDetails(accountNumber, balance);

}

public void updateView(String accountNumber) {

view.displayAccountDetails(accountNumber, model.getBalance(accountNumber));

}

}

class Application {

public static void main(String[] args) {

BankAccount bankAccount = new BankAccount();

BankView bankView = new BankView();

BankController bankController = new BankController(bankAccount, bankView);

Scanner scanner = new Scanner(System.in);

int choice;

do {

System.out.println("1. Create Account");

System.out.println("2. Deposit");

System.out.println("3. Withdraw");

System.out.println("4. Check Balance");

System.out.print("Enter your choice: ");

choice = scanner.nextInt();

scanner.nextLine();

switch (choice) {

case 1:

System.out.print("Enter account number: ");

String accountNumber = scanner.nextLine();

System.out.print("Enter initial balance: ");

double initialBalance;

while (true) {

try {

initialBalance = Double.parseDouble(scanner.nextLine());

break;

} catch (NumberFormatException e) {

System.out.println("Invalid input. Please enter a valid numeric value.");

}

}

bankController.createAccount(accountNumber, initialBalance);

break;

case 2:

System.out.print("Enter account number: ");

String depositAccountNumber = scanner.nextLine();

System.out.print("Enter deposit amount: ");

double depositAmount;

while (true) {

try {

depositAmount = Double.parseDouble(scanner.nextLine());

break;

} catch (NumberFormatException e) {

System.out.println("Invalid input. Please enter a valid numeric value.");

}

}

bankController.deposit(depositAccountNumber, depositAmount);

break;

case 3:

System.out.print("Enter account number: ");

String withdrawAccountNumber = scanner.nextLine();

System.out.print("Enter withdrawal amount: ");

double withdrawAmount;

while (true) {

try {

withdrawAmount = Double.parseDouble(scanner.nextLine());

break;

} catch (NumberFormatException e) {

System.out.println("Invalid input. Please enter a valid numeric value.");

}

}

bankController.withdraw(withdrawAccountNumber, withdrawAmount);

break;

case 4:

System.out.print("Enter account number: ");

String checkBalanceAccountNumber = scanner.nextLine();

bankController.checkBalance(checkBalanceAccountNumber);

break;

default:

System.out.println("Invalid choice. Please enter a valid option.");

}

} while (choice != 5);

}

}