**Implement the following projects which focus on different aspects of Java programming, including:**

Object-Oriented Programming principles

Exception handling

File operations

Data structures

Input validation

User interface design

**Bank Account Manager**

Design a BankAccount class that manages deposits, withdrawals, and maintains balance with proper validation using object oriented programming in Java.

Key points:

- Prevent negative balance

- Track transaction history

- Use proper access modifiers

import java.io.\*;

import java.time.LocalDateTime;

import java.time.format.DateTimeFormatter;

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

// Custom exceptions

class InsufficientBalanceException extends Exception {

public InsufficientBalanceException(String message) {

super(message);

}

}

class InvalidAmountException extends Exception {

public InvalidAmountException(String message) {

super(message);

}

}

// Transaction class to store transaction details

class Transaction implements Serializable {

private final String type;

private final double amount;

private final double balance;

private final LocalDateTime timestamp;

public Transaction(String type, double amount, double balance) {

this.type = type;

this.amount = amount;

this.balance = balance;

this.timestamp = LocalDateTime.now();

}

@Override

public String toString() {

DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss");

return String.format("[%s] %s: $%.2f - Balance: $%.2f",

timestamp.format(formatter), type, amount, balance);

}

}

// BankAccount class with main functionality

class BankAccount implements Serializable {

private static final long serialVersionUID = 1L;

private final String accountNumber;

private final String accountHolder;

private double balance;

private final List<Transaction> transactions;

private final String transactionFilePath;

public BankAccount(String accountNumber, String accountHolder, double initialBalance)

throws InvalidAmountException {

if (initialBalance < 0) {

throw new InvalidAmountException("Initial balance cannot be negative");

}

this.accountNumber = accountNumber;

this.accountHolder = accountHolder;

this.balance = initialBalance;

this.transactions = new ArrayList<>();

this.transactionFilePath = "transactions\_" + accountNumber + ".txt";

// Record initial deposit

addTransaction("INITIAL\_DEPOSIT", initialBalance);

}

// Getter methods

public String getAccountNumber() { return accountNumber; }

public String getAccountHolder() { return accountHolder; }

public double getBalance() { return balance; }

// Deposit method

public void deposit(double amount) throws InvalidAmountException {

if (amount <= 0) {

throw new InvalidAmountException("Deposit amount must be positive");

}

balance += amount;

addTransaction("DEPOSIT", amount);

}

// Withdrawal method

public void withdraw(double amount)

throws InvalidAmountException, InsufficientBalanceException {

if (amount <= 0) {

throw new InvalidAmountException("Withdrawal amount must be positive");

}

if (amount > balance) {

throw new InsufficientBalanceException(

"Insufficient balance. Current balance: $" + String.format("%.2f", balance));

}

balance -= amount;

addTransaction("WITHDRAWAL", amount);

}

// Add transaction to history and save to file

private void addTransaction(String type, double amount) {

Transaction transaction = new Transaction(type, amount, balance);

transactions.add(transaction);

saveTransactionToFile(transaction);

}

// Save transaction to file

private void saveTransactionToFile(Transaction transaction) {

try (FileWriter fw = new FileWriter(transactionFilePath, true);

BufferedWriter bw = new BufferedWriter(fw);

PrintWriter out = new PrintWriter(bw)) {

out.println(transaction.toString());

} catch (IOException e) {

System.err.println("Error saving transaction: " + e.getMessage());

}

}

// Get transaction history

public List<Transaction> getTransactionHistory() {

return new ArrayList<>(transactions);

}

// Print account statement

public void printStatement() {

System.out.println("\n=== Account Statement ===");

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

System.out.println("Account Holder: " + accountHolder);

System.out.println("Current Balance: $" + String.format("%.2f", balance));

System.out.println("\nTransaction History:");

for (Transaction transaction : transactions) {

System.out.println(transaction);

}

System.out.println("=====================");

}

}

// Main class with user interface

public class BankAccountManager {

private static final Scanner scanner = new Scanner(System.in);

public static void main(String[] args) {

try {

// Create new account

System.out.println("=== Create New Bank Account ===");

System.out.print("Enter Account Number: ");

String accountNumber = scanner.nextLine();

System.out.print("Enter Account Holder Name: ");

String accountHolder = scanner.nextLine();

System.out.print("Enter Initial Deposit Amount: $");

double initialDeposit = validateAmount(scanner.nextLine());

BankAccount account = new BankAccount(accountNumber, accountHolder, initialDeposit);

System.out.println("\nAccount created successfully!");

// Main operation loop

while (true) {

displayMenu();

String choice = scanner.nextLine();

switch (choice.toLowerCase()) {

case "1":

handleDeposit(account);

break;

case "2":

handleWithdrawal(account);

break;

case "3":

account.printStatement();

break;

case "4":

System.out.println("Thank you for using our banking services!");

return;

default:

System.out.println("Invalid choice. Please try again.");

}

}

} catch (Exception e) {

System.out.println("Error: " + e.getMessage());

} finally {

scanner.close();

}

}

private static void displayMenu() {

System.out.println("\n=== Bank Account Menu ===");

System.out.println("1. Deposit");

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

System.out.println("3. View Statement");

System.out.println("4. Exit");

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

}

private static void handleDeposit(BankAccount account) {

try {

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

double amount = validateAmount(scanner.nextLine());

account.deposit(amount);

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

} catch (Exception e) {

System.out.println("Error: " + e.getMessage());

}

}

private static void handleWithdrawal(BankAccount account) {

try {

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

double amount = validateAmount(scanner.nextLine());

account.withdraw(amount);

System.out.println("Withdrawal successful!");

} catch (Exception e) {

System.out.println("Error: " + e.getMessage());

}

}

private static double validateAmount(String input) throws InvalidAmountException {

try {

double amount = Double.parseDouble(input);

if (amount <= 0) {

throw new InvalidAmountException("Amount must be positive");

}

return amount;

} catch (NumberFormatException e) {

throw new InvalidAmountException("Invalid amount format");

}

}

}

o/p

=== Create New Bank Account ===

Enter Account Number: 101

Enter Account Holder Name: Ram Kumar

Enter Initial Deposit Amount: $10000

Account created successfully!

=== Bank Account Menu ===

1. Deposit

2. Withdraw

3. View Statement

4. Exit

Enter your choice: 3

=== Account Statement ===

Account Number: 101

Account Holder: Ram Kumar

Current Balance: $10000.00

Transaction History:

[2025-01-11 12:30:39] INITIAL\_DEPOSIT: $10000.00 - Balance: $10000.00

=====================

=== Bank Account Menu ===

1. Deposit

2. Withdraw

3. View Statement

4. Exit

Enter your choice: 2

Enter withdrawal amount: $2000

Withdrawal successful!

=== Bank Account Menu ===

1. Deposit

2. Withdraw

3. View Statement

4. Exit

Enter your choice: 3

=== Account Statement ===

Account Number: 101

Account Holder: Ram Kumar

Current Balance: $8000.00

Transaction History:

[2025-01-11 12:30:39] INITIAL\_DEPOSIT: $10000.00 - Balance: $10000.00

[2025-01-11 12:30:55] WITHDRAWAL: $2000.00 - Balance: $8000.00

=====================