# Day 4

1)

package day4;

import java.util.\*;

// Custom exception for negative amounts

class NegativeAmountException extends Exception {

    public NegativeAmountException(String message) {

        super(message);

    }

}

// Custom exception for low balance

class LowBalanceException extends Exception {

    public LowBalanceException(String message) {

        super(message);

    }

}

// BankAccount class

class BankAccount {

    private int accNo;

    private String custName;

    private String accType;

    private float balance;

    // Constructor

    public BankAccount(int accNo, String custName, String accType, float balance) throws NegativeAmountException, LowBalanceException {

        this.accNo = accNo;

        this.custName = custName;

        this.accType = accType;

        if (balance < 0) {

            throw new NegativeAmountException("NegativeAmount");

        }

        if ((accType.equalsIgnoreCase("Saving") && balance < 1000) ||

            (accType.equalsIgnoreCase("Current") && balance < 5000)) {

            throw new LowBalanceException("LowBalance");

        }

        this.balance = balance;

    }

    // Deposit method

    public void deposit(float amt) throws NegativeAmountException {

        if (amt < 0) {

            throw new NegativeAmountException("NegativeAmount");

        }

        this.balance += amt;

    }

    // Get balance method

    public float getBalance() throws LowBalanceException {

        if ((accType.equalsIgnoreCase("Saving") && balance < 1000) ||

            (accType.equalsIgnoreCase("Current") && balance < 5000)) {

            throw new LowBalanceException("LowBalance");

        }

        return balance;

    }

}

// Main class

public class BankAccountApp {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        try {

            // Input account details

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

            int accNo = scanner.nextInt();

            scanner.nextLine(); // Consume newline

            System.out.println("Enter Customer Name:");

            String custName = scanner.nextLine();

            System.out.println("Enter Account Type (Saving/Current):");

            String accType = scanner.nextLine();

            System.out.println("Enter Initial Balance:");

            float balance = scanner.nextFloat();

            // Create account

            BankAccount account = new BankAccount(accNo, custName, accType, balance);

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

        } catch (NegativeAmountException | LowBalanceException e) {

            System.out.println(e.getMessage());

        } finally {

            scanner.close();

        }

    }

}

Enter Customer Name:

john current

Enter Account Type (Saving/Current):

current

Enter Initial Balance:

4000

LowBalance

Enter Account Number:

123

Enter Customer Name:

john

Enter Account Type (Saving/Current):

savings

Enter Initial Balance:

-900

NegativeAmount

2)

package day4;

import java.util.Scanner;

// Custom exception for NotEligible case

class NotEligibleException extends Exception {

    public NotEligibleException(String message) {

        super(message);

    }

}

class CricketRating {

    private String playerName;

    private float critic1;

    private float critic2;

    private float critic3;

    private float avgRating;

    private String coins;

    // Constructor to initialize player name and critics' ratings

    public CricketRating(String playerName, float critic1, float critic2) {

        this.playerName = playerName;

        this.critic1 = critic1;

        this.critic2 = critic2;

        this.critic3 = 0.0f; // default value for critic3

        calculateAverageRating(critic1, critic2); // Calculate average

    }

    // Constructor for 3 critics

    public CricketRating(String playerName, float critic1, float critic2, float critic3) {

        this.playerName = playerName;

        this.critic1 = critic1;

        this.critic2 = critic2;

        this.critic3 = critic3;

        calculateAverageRating(critic1, critic2, critic3); // Calculate average

    }

    // Method to calculate average rating based on two critics

    public void calculateAverageRating(float critic1, float critic2) {

        this.avgRating = (critic1 + critic2) / 2;

    }

    // Method to calculate average rating based on three critics

    public void calculateAverageRating(float critic1, float critic2, float critic3) {

        this.avgRating = (critic1 + critic2 + critic3) / 3;

    }

    // Method to calculate the type of coin based on the average rating

    public String calculateCoins() throws NotEligibleException {

        if (avgRating >= 7) {

            coins = "Gold";

        } else if (avgRating >= 5) {

            coins = "Silver";

        } else if (avgRating >= 2) {

            coins = "Copper";

        } else {

            throw new NotEligibleException("NotEligible");

        }

        return coins;

    }

    // Method to display all the information

    public void display() {

        try {

            String coin = calculateCoins();

            System.out.printf("%s %.2f %s\n", playerName, avgRating, coin);

        } catch (NotEligibleException e) {

            System.out.println(e.getMessage());

        }

    }

}

public class Tester {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        // Read input values

        String playerName = scanner.next();

        float critic1 = scanner.nextFloat();

        float critic2 = scanner.nextFloat();

        float critic3 = 0.0f;

        // Check if there is a third critic's score

        if (scanner.hasNextFloat()) {

            critic3 = scanner.nextFloat();

        }

        // Create a CricketRating object based on the input

        CricketRating cricketRating;

        if (critic3 == 0.0f) {

            cricketRating = new CricketRating(playerName, critic1, critic2);

        } else {

            cricketRating = new CricketRating(playerName, critic1, critic2, critic3);

        }

        // Display the result

        cricketRating.display();

        scanner.close();

    }

}

john

9.3

9.67

8.7

john 9.22 Gold

henry

1.5

1.5

1.5

NotEligible

3)

package day4;

import java.util.Scanner;

// User-defined exception for CatheyBank application validation errors

class CatheyBankException extends Exception {

    public CatheyBankException(String message) {

        super(message);

    }

}

// Exception for Invalid Applicant Name

class InvalidNameException extends CatheyBankException {

    public InvalidNameException(String message) {

        super(message);

    }

}

// Exception for Invalid Post

class InvalidPostException extends CatheyBankException {

    public InvalidPostException(String message) {

        super(message);

    }

}

// Exception for Invalid Age

class InvalidAgeException extends CatheyBankException {

    public InvalidAgeException(String message) {

        super(message);

    }

}

class Applicant {

    private String applicantName;

    private String postApplied;

    private int applicantAge;

    // Constructor to initialize the applicant details

    public Applicant(String applicantName, String postApplied, int applicantAge) {

        this.applicantName = applicantName;

        this.postApplied = postApplied;

        this.applicantAge = applicantAge;

    }

    // Getters for the fields

    public String getApplicantName() {

        return applicantName;

    }

    public String getPostApplied() {

        return postApplied;

    }

    public int getApplicantAge() {

        return applicantAge;

    }

}

class Validator {

    // Method to validate the applicant details

    public void validate(Applicant applicant) {

        try {

            if (!isValidApplicantName(applicant.getApplicantName())) {

                throw new InvalidNameException("Invalid Applicant Name");

            }

            if (!isValidPost(applicant.getPostApplied())) {

                throw new InvalidPostException("Invalid Post");

            }

            if (!isValidAge(applicant.getApplicantAge())) {

                throw new InvalidAgeException("Invalid Age");

            }

            // If all validations are successful

            System.out.println("All details are valid");

        } catch (CatheyBankException e) {

            System.out.println(e.getMessage());

        }

    }

    // Method to validate the applicant name

    public boolean isValidApplicantName(String name) {

        return name != null && !name.trim().isEmpty();

    }

    // Method to validate the post applied for

    public boolean isValidPost(String post) {

        return post.equals("Probationary Officer") || post.equals("Assistant") || post.equals("Special Cadre Officer");

    }

    // Method to validate the age of the applicant

    public boolean isValidAge(int age) {

        return age > 18 && age <= 30;

    }

}

public class CatheyBank {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        // Taking user input for applicant details

        String applicantName = scanner.nextLine();

        String postApplied = scanner.nextLine();

        int applicantAge = scanner.nextInt();

        // Create an Applicant object

        Applicant applicant = new Applicant(applicantName, postApplied, applicantAge);

        // Create a Validator object to validate the details

        Validator validator = new Validator();

        // Validate the applicant details

        validator.validate(applicant);

        scanner.close();

    }

}

mary

Assistant

34

Invalid Age

mary

clerk

27

Invalid Post

Joseph

Probationary Officer

30

All details are valid