# Rajalakshmi Engineering College

Name: MRIDHULA DEVI M
Email: 240701337@rajalakshmi.edu.in
Roll no:
Phone: 9840329629
Branch: REC
Department: CSE - Section 8
Batch: 2028
Degree: B.E - CSE

Scan to verify results

## 2024_28_III_OOPS Using Java Lab

### REC_2028_OOPS using Java_Week 6_CY

Attempt : 1
Total Mark : 40
Marks Obtained : 40

### Section 1 : Coding

1. Problem Statement

Mary is managing a business and wants to analyze its profitability. She operates both a regular business model and a seasonal business model. To assess profitability, she uses a program that calculates and compares the profit margins for both models based on revenue and cost.

The program defines:

BusinessUtility class with a method calculateMargin(double revenue, double cost).SeasonalBusinessUtility (inherits from BusinessUtility) and overrides calculateMargin(double revenue, double cost), adding a seasonal adjustment of 10% to the base margin.ProfitabilityChecker class with a method checkProfitability(double regularMargin), which prints "Business is profitable." if the regular margin is 10% or more, otherwise prints "Business is not profitable.".

Mary inputs revenue and cost, and the program compute and display the regular and seasonal margins using:

Margin = ((Revenue − Cost) / Revenue) × 100

Seasonal Margin = Margin + 10

***Input Format***

The first line of input consists of a double value r, representing the revenue.

The second line consists of a double value c, representing the cost.

***Output Format***

The first line prints a double value, representing the regular profit margin, rounded to two decimal places, in the format: "Regular Margin: X. XX%", where X.XX denotes the calculated regular margin.

The second line prints a double value, representing the seasonal profit margin, rounded to two decimal places, in the format: "Seasonal Margin: X. XX%", where X.XX denotes the calculated seasonal margin.

The third line prints a string, indicating whether the business is profitable or not profitable, based on the regular margin.

If the regular margin is less than 10, print "Business is not profitable.". If it is 10 or greater, print "Business is profitable."

Refer to the sample output for the formatting specifications.

***Sample Test Case***

Input: 1000.0
800.0

Output: Regular Margin: 20.00%
Seasonal Margin: 30.00%
Business is profitable.

***Answer***

```
import java.util.Scanner;
```

```
// You are using Java
class BusinessUtility {
    public double calculateMargin(double revenue, double cost) {
        return ((revenue - cost) / revenue) * 100;
    }
}

class SeasonalBusinessUtility extends BusinessUtility {
    @Override
    public double calculateMargin(double revenue, double cost) {
        double baseMargin = super.calculateMargin(revenue, cost);
        return baseMargin + 10.0;
    }
}

class ProfitabilityChecker {
    public void checkProfitability(double regularMargin) {
        if (regularMargin >= 10.0) {
            System.out.print("Business is profitable.");
        } else {
            System.out.print("Business is not profitable.");
        }
    }
}
class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        double revenue = scanner.nextDouble();
        double cost = scanner.nextDouble();
        BusinessUtility business = new BusinessUtility();
        SeasonalBusinessUtility seasonalBusiness = new
SeasonalBusinessUtility();
        double regularMargin = business.calculateMargin(revenue, cost);
        double seasonalMargin = seasonalBusiness.calculateMargin(revenue,
cost);

        System.out.printf("Regular Margin: %.2f%%\n", regularMargin);
        System.out.printf("Seasonal Margin: %.2f%%\n", seasonalMargin);

        ProfitabilityChecker checker = new ProfitabilityChecker();
        checker.checkProfitability(regularMargin);
        scanner.close();
```

```
    }
}
```

***Status :*** Correct ***Marks : 10/10***

2. Problem Statement

A bank provides two types of deposit schemes: Fixed Deposits (FD) and Recurring Deposits (RD). Customers want to calculate the interest they can earn based on their selected scheme.

Develop a Java program using inheritance to compute the interest for FD and RD. The program should include:

A base class Account with attributes accountHolder and principalAmount, along with a method for interest calculation.A subclass FixedDeposit that calculates interest for FD.A subclass RecurringDeposit that calculates interest for RD.

Formulas Used:

Interest for FD: (principal amount * duration in years * rate of interest) / 100

Interest for RD: (maturity amount * duration in months * rate of interest) / (12 * 100), where maturity amount = monthly deposit * duration in months.

***Input Format***

The first line of input consists of the choice (1 for FD, 2 for RD).

If the choice is 1, the following lines consist of account holder (string), principal amount (double), duration in years (int), and rate of interest (double).

If the choice is 2, the following lines consist of account holder (string), monthly deposit (int), duration in months (int), and rate of interest (double).

***Output Format***

The output prints the calculated interest with one decimal place in the following format.

For choice 1: "Interest for FD: <calculated interest >"

For choice 2: "Interest for FD: <calculated interest >"

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: 1
Alice
50000.56
5
6.5
Output: Interest for FD: 16250.2

***Answer***

```
import java.util.Scanner;

// You are using Java

class Account {
    protected String accountHolder;
    protected double principalAmount;

    public Account(String accountHolder, double principalAmount) {
        this.accountHolder = accountHolder;
        this.principalAmount = principalAmount;
    }

    public double calculateInterest() {
        return 0.0;
    }
}

class FixedDeposit extends Account {
    private int durationInYears;
    private double rateOfInterest;

    public FixedDeposit(String accountHolder, double principalAmount, int
durationInYears, double rateOfInterest) {
        super(accountHolder, principalAmount);
```

```
        this.durationInYears = durationInYears;
        this.rateOfInterest = rateOfInterest;
    }

    @Override
    public double calculateInterest() {
        return (principalAmount * durationInYears * rateOfInterest) / 100.0;
    }
}

class RecurringDeposit extends Account {
    private int monthlyDeposit;
    private int durationInMonths;
    private double rateOfInterest;

    public RecurringDeposit(String accountHolder, int monthlyDeposit, int
durationInMonths, double rateOfInterest) {
        super(accountHolder, 0); // principalAmount is not directly used here
        this.monthlyDeposit = monthlyDeposit;
        this.durationInMonths = durationInMonths;
        this.rateOfInterest = rateOfInterest;
    }

    @Override
    public double calculateInterest() {
        double maturityAmount = monthlyDeposit * durationInMonths;
        return (maturityAmount * durationInMonths * rateOfInterest) / (12 * 100.0);
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        int choice = sc.nextInt();

        switch (choice) {
            case 1:
                sc.nextLine();
                String fdName = sc.nextLine();
                double fdPrincipal = sc.nextDouble();
                int fdDuration = sc.nextInt();
                double fdRate = sc.nextDouble();
```

```
            FixedDeposit fd = new FixedDeposit(fdName, fdPrincipal, fdDuration,
fdRate);
            System.out.printf("Interest for FD: %.1f", fd.calculateInterest());
            break;

        case 2:
            sc.nextLine();
            String rdName = sc.nextLine();
            int rdDeposit = sc.nextInt();
            int rdDuration = sc.nextInt();
            double rdRate = sc.nextDouble();

            RecurringDeposit rd = new RecurringDeposit(rdName, rdDeposit,
rdDuration, rdRate);
            System.out.printf("Interest for RD: %.1f", rd.calculateInterest());
            break;

        default:
            System.out.println("Invalid Choice");
    }
  }
}
```

***Status :*** Correct **Marks : 10/10**

3. Problem Statement

Arun wants to calculate the age gap between the grandfather and the son and determine the father's age after 5 years.

Your task is to assist him in developing a program using three classes: GrandFather, Father, and Son, where the GrandFather stores the grandfather's age, the Father extends GrandFather to include the father's age and calculates his age after 5 years, and Son extends Father to include the son's age and calculate the age difference between the grandfather and the son.

***Input Format***

The input consists of three integers representing the ages of the grandfather,

father, and son, one per line.

***Output Format***

The first line of output prints "Grandfather and son's age gap:" followed by an integer representing the age gap between the grandfather and the son, ending with "years".

The second line prints "Father's Age:" followed by an integer representing the father's age after 5 years, ending with "years".

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: 50
30
3
Output: Grandfather and son's age gap: 47 years
Father's Age: 35 years

***Answer***

```
import java.util.Scanner;

// You are using Java
class GrandFather {
    protected int grandfatherAge;

    public void setGrandfatherAge(int age) {
        this.grandfatherAge = age;
    }
}

class Father extends GrandFather {
    protected int fatherAge;

    public void setFatherAge(int age) {
        this.fatherAge = age;
    }

    public int calculateFatherAgeAfter5Years() {
```

```
        return fatherAge + 5;
    }
}

class Son extends Father {
    private int sonAge;

    public void setSonAge(int age) {
        this.sonAge = age;
    }

    public int calculateGrandfatherSonAgeDifference() {
        return grandfatherAge - sonAge;
    }
}
public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Son son = new Son();

        int grandfatherAge = scanner.nextInt();
        son.setGrandfatherAge(grandfatherAge);

        int fatherAge = scanner.nextInt();
        son.setFatherAge(fatherAge);

        int sonAge = scanner.nextInt();
        son.setSonAge(sonAge);

        System.out.println("Grandfather and son's age gap: "+
son.calculateGrandfatherSonAgeDifference() + " years");

        int fatherAgeAfter5Years = son.calculateFatherAgeAfter5Years();
        System.out.println("Father's Age: " + fatherAgeAfter5Years + " years");
    }
}
```

***Status :*** Correct ***Marks : 10/10***

4. Problem Statement

Teena is launching a new airline, Boeing747, and needs to calculate the total revenue generated from ticket sales based on the ticket cost and seat availability. Teena's airline offers two types of seats: regular and premium. The ticket cost and seat availability for both types of seats need to be considered for revenue calculation.

To help with this, Teena wants to implement a system using multilevel inheritance with three classes:

Airline: This class will have the ticket cost as an attribute and defines the method setCost(double cost) and double getCost().Indigo: This class will extend Airline and add the seat availability attribute and defines the method getSeatAvailability() and setSeatAvailability(int seatAvailability) .Boeing747: This class will extend Indigo and include a method calculateTotalRevenue() based on the ticket cost and seat availability .

Teena needs to calculate the total revenue using the formula:

Total Revenue = ticket cost * seat availability

Help Teena implement this system for calculating the revenue of her airline.

***Input Format***

The first line of input consists of a double value, representing the flight's ticket cost.

The second line consists of an integer, representing seat availability.

***Output Format***

The first line of output prints "Ticket Cost: Rs. " followed by a double value representing the ticket cost rounded to one decimal place.

The second line of output prints "Seat Availability: X seats" where X is an integer value representing the seat availability.

The third line of output prints "Total Revenue: Rs. " followed by a double value representing the total revenue rounded to one decimal place.

Refer to the sample output for the exact text and format.

***Sample Test Case***

Input: 1000.0
100

Output: Ticket Cost: Rs. 1000.0
Seat Availability: 100 seats
Total Revenue: Rs. 100000.0

***Answer***

```
import java.util.Scanner;

// You are using Java
class Airline {
    private double ticketCost;

    public void setCost(double cost) {
        this.ticketCost = cost;
    }

    public double getCost() {
        return ticketCost;
    }
}

class Indigo extends Airline {
    private int seatAvailability;

    public void setSeatAvailability(int seatAvailability) {
        this.seatAvailability = seatAvailability;
    }

    public int getSeatAvailability() {
        return seatAvailability;
    }
}

class Boeing747 extends Indigo {
    public double calculateTotalRevenue() {
        return getCost() * getSeatAvailability();
    }
```

```
}
public class Main {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      Boeing747 plane = new Boeing747();

      double ticketCost = scanner.nextDouble();
      plane.setCost(ticketCost);
      int seatAvailability = scanner.nextInt();
      plane.setSeatAvailability(seatAvailability);

      System.out.printf("Ticket Cost: Rs. %.1f\n", plane.getCost());
      System.out.println("Seat Availability: " + plane.getSeatAvailability() + "
seats");
      System.out.printf("Total Revenue: Rs. %.1f\n",
plane.calculateTotalRevenue());
   }
}
```

***Status :*** Correct ***Marks : 10/10***