Rajalakshmi Engineering College

Name: Jathin Chowdary CH
Email: 240701209@rajalakshmi.edu.in
Roll no: 240701209
Phone: 8939767767
Branch: REC
Department: CSE - Section 10
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

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;


class GrandFather {
   private int grandfatherAge;

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

   public int getGrandfatherAge() {
      return grandfatherAge;
   }
}

class Father extends GrandFather {
   private int fatherAge;

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

```
    public int getFatherAge() {
        return fatherAge;
    }

    public int calculateFatherAgeAfter5Years() {
        return getFatherAge() + 5;
    }
}

class Son extends Father {
    private int sonAge;

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

    public int getSonAge() {
        return sonAge;
    }

    public int calculateGrandfatherSonAgeDifference() {
        int ageDifferencGrandfatherSon = getGrandfatherAge() - getSonAge();
        return ageDifferencGrandfatherSon;
    }
}
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***

2. 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;


class Airline {
   private double cost;
   public void setCost(double cost) {
      this.cost = cost;
   }
   public double getCost() {
      return cost;
   }
}
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***

3. 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;

class Account {
   String accountHolder;
   double principalAmount;
```

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

    double calculateInterest() {
        return 0;
    }
}


class FixedDeposit extends Account {
    int durationInYears;
    double rateOfInterest;
    public FixedDeposit(String accountHolder, double principalAmount, int
durationInYears, double rateOfInterest) {
        super(accountHolder, principalAmount);
        this.durationInYears = durationInYears;
        this.rateOfInterest = rateOfInterest;
    }


    double calculateInterest() {
        return (principalAmount * durationInYears * rateOfInterest) / 100;
    }
}

class RecurringDeposit extends Account {
    int durationInMonths;
    double rateOfInterest;
    int monthlyDeposit;
    public RecurringDeposit(String accountHolder, int monthlyDeposit, int
durationInMonths, double rateOfInterest) {
        super(accountHolder, 0);
        this.monthlyDeposit = monthlyDeposit;
        this.durationInMonths = durationInMonths;
        this.rateOfInterest = rateOfInterest;
    }
    double calculateInterest() {
        double maturityAmount = monthlyDeposit * durationInMonths;
        return (maturityAmount * durationInMonths * rateOfInterest) / (12 * 100);
    }
}
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***

4. Problem Statement

A painter needs to determine the cost to paint different shapes based on their surface area. The program should be designed to handle the area of a sphere and calculate the total painting cost using the following formulas:

Area of sphere: Area = 4 * pi * r^2 where pi = 3.14Total painting cost: Cost = cost per square meter * area of sphere

The program will consist of three classes:

Shape class: This class should set the shape type and radius.Area class: This class should extend Shape to calculate the area.Cost class: This class should extend Area to calculate the total painting cost.

Input Format

The input consists of a string representing the shape type, a double value representing the radius, and another double value representing the cost per square meter on each line.

Output Format

For a valid shape type of "Sphere":

- The first line prints: "Area of Sphere is: <calculated_area>" rounded to two decimal places.
- The second line prints: "Cost to paint the shape is: <total_painting_cost>" rounded to two decimal places.

For any other shape types, print: "Invalid type".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: Sphere
3.4
5.8
Output: Area of Sphere is: 145.19
Cost to paint the shape is: 842.12

Answer

```
import java.util.Scanner;


class Shape {
    String shapeName = "";
    double radius;
    void setShape(String x, Scanner scanner) {
        shapeName = x;
        if (x.equals("Sphere")) {
            radius = scanner.nextDouble();
        } else {
            System.out.println("Invalid type");
            return;
        }
    }
}

class Area extends Shape {
    double area;
    void calculateArea() {
        if (shapeName.equals("Sphere")) {
            area = 4 * 3.14 * radius * radius;
            System.out.printf("Area of Sphere is: %.2f%n", area);
        }
    }
}

class Cost extends Area {
    double costPerMtrSquare;

    void setCost(double x) {
        costPerMtrSquare = x;
    }
    void calculateCost() {
        double totalCost = costPerMtrSquare * area;
        if (area > 0) {
            System.out.printf("Cost to paint the shape is: %.2f%n", totalCost);
        }
    }
}
```

```
public class Main {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    String s = scanner.next();
    Cost shape = new Cost();
    shape.setShape(s, scanner);
    double costToPaint = scanner.nextDouble();
    shape.calculateArea();
    shape.setCost(costToPaint);
    shape.calculateCost();
  }
}
```

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