Rajalakshmi Engineering College

Name: KAVYA SRIRAM
Email: 241901045@rajalakshmi.edu.in
Roll no: 241901045
Phone: 8939657782
Branch: REC
Department: CSE (CS) - Section 1
Batch: 2028
Degree: B.E - CSE (CS)

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

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 {
   protected String accountHolder;

   public Account(String accountHolder) {
      this.accountHolder = accountHolder;
```

```
    }

    public double calculateInterest() {
        return 0.0;
    }
}

class FixedDeposit extends Account {
    private double principalAmount;
    private int durationYears;
    private double rateOfInterest;

    public FixedDeposit(String accountHolder, double principalAmount, int
durationYears, double rateOfInterest) {
        super(accountHolder);
        this.principalAmount = principalAmount;
        this.durationYears = durationYears;
        this.rateOfInterest = rateOfInterest;
    }

    @Override
    public double calculateInterest() {
        return (principalAmount * durationYears * rateOfInterest) / 100;
    }
}

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

    public RecurringDeposit(String accountHolder, int monthlyDeposit, int
durationMonths, double rateOfInterest) {
        super(accountHolder);
        this.monthlyDeposit = monthlyDeposit;
        this.durationMonths = durationMonths;
        this.rateOfInterest = rateOfInterest;
    }

    @Override
    public double calculateInterest() {
        double maturityAmount = monthlyDeposit * durationMonths;
        return (maturityAmount * durationMonths * 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***

2. Problem Statement

Bob has been tasked with creating a program using CircleUtils class to calculate and display the circumference and area of the circle.

The program should allow Bob to input the radius of a circle as both an integer and a double and compute both the circumference and area of the circle using separate overloaded methods:

calculateCircumference- To calculate the circumference using the formula 2 * 3.14 * radiuscalculateArea- To calculate the area 3.14 * radius * radius

Write a program to help Bob.

Input Format

The first line of input consists of an integer m, representing the radius of the circle as a whole number.

The second line consists of a double value n, representing the radius of the circle as a decimal number.

Output Format

The first line of output displays two space-separated double values, rounded to two decimal places, representing the circumference of the circle with the integer radius and the double radius, respectively.

The second line displays two space-separated double values, rounded to two decimal places, representing the area of the circle with the integer radius and the double radius, respectively.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
3.50

Output: 31.40 21.98
78.50 38.47

Answer

```
import java.util.Scanner;
class CircleUtils {

   public static double calculateCircumference(int radius) {
      return 2 * 3.14 * radius;
   }

   public static double calculateCircumference(double radius) {
      return 2 * 3.14 * radius;
   }

   public static double calculateArea(int radius) {
      return 3.14 * radius * radius;
   }

   public static double calculateArea(double radius) {
      return 3.14 * radius * radius;
   }
}
class Main {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);

      int radiusInt = scanner.nextInt();
      double radiusDouble = scanner.nextDouble();

      CircleUtils circleUtils = new CircleUtils();

      double circumferenceInt = circleUtils.calculateCircumference(radiusInt);
      double circumferenceDouble =
circleUtils.calculateCircumference(radiusDouble);
      double areaInt = circleUtils.calculateArea(radiusInt);
      double areaDouble = circleUtils.calculateArea(radiusDouble);

      System.out.format("%.2f %.2f\n", circumferenceInt, circumferenceDouble);
      System.out.format("%.2f %.2f", areaInt, areaDouble);

      scanner.close();
   }
}
```

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

3. Problem Statement

Teena's retail store has implemented a Loyalty Points System to reward customers based on their spending. The program calculates and displays the loyalty points based on whether the customer is a regular or a premium customer.

For regular customers (class Customer), the loyalty points are calculated as:

Loyalty points = amount spent / 10

For premium customers (class PremiumCustomer, which inherits from Customer), the loyalty points are calculated as:

Loyalty points = 2 * (amount spent / 10)

The program should use method overriding for premium customers to calculate their loyalty points. The method that needs to be overridden is calculateLoyaltyPoints in the Customer class.

Input Format

The first line of input consists of an integer representing the amount spent by the customer.

The second line consists of a string representing the premium customer status:

- "yes" if the customer is a premium customer.
- "no" if the customer is not a premium customer.

Output Format

The output should display the loyalty points earned based on the amount spent and the customer type.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 50
yes

Output: 10

Answer

```
import java.util.Scanner;

class Customer {
   public Customer() {
   }

   public int calculateLoyaltyPoints(int amountSpent) {
      return amountSpent / 10;
   }
}

class PremiumCustomer extends Customer {
   public PremiumCustomer() {
   }

   @Override
   public int calculateLoyaltyPoints(int amountSpent) {
      return 2 * (amountSpent / 10);
   }
}

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

      int amountSpent = scanner.nextInt();

      String isPremium = scanner.next().toLowerCase();

      Customer customer;

      if (isPremium.equals("yes")) {
         customer = new PremiumCustomer();
      } else {
         customer = new Customer();
      }
```

```
        int loyaltyPoints = customer.calculateLoyaltyPoints(amountSpent);

        System.out.println(loyaltyPoints);
    }
}
```

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

4. 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;

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;
   }
}
```

```
class ProfitabilityChecker {
   public void checkProfitability(double regularMargin) {
      if (regularMargin >= 10) {
         System.out.println("Business is profitable.");
      } else {
         System.out.println("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***