Rajalakshmi Engineering College

Name: Jovitha Sheethal B
Email: 241001099@rajalakshmi.edu.in
Roll no: 241001099
Phone: 6385334710
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 5_MCQ

Attempt : 1
Total Mark : 15
Marks Obtained : 13

Section 1 : MCQ

1. What will be the output of the following code?

```
class Box {
   int length = 5;
   int width = 4;

   int area() {
      return length * width;
   }

   public static void main(String[] args) {
      Box b = new Box();
      System.out.println("Area = " + b.area());
   }
}
```

Answer

Area = 20

Status : Correct ***Marks : 1/1***

2. What will be the output of the following code?

```
class A {
   int x = 50;
}

public class Main {
   public static void main(String[] args) {
      A obj1 = new A();
      A obj2 = obj1;
      obj2.x = 100;
      System.out.println(obj1.x);
   }
}
```

Answer

100

Status : Correct ***Marks : 1/1***

3. What will be the output of the following code?

```
class Person {
   String name;
   void setName(String n) {
      name = n;
   }
   void printName() {
      System.out.println(name);
   }
}

class Test {
  public static void main(String[] args) {
    Person p = new Person();
    p.printName();
  }
}
```

Answer

null

Status : Correct ***Marks : 1/1***

4. What will be the output of the following code?

```
class MathUtils {
  int add(int x) {
    return x + x;
  }
}

public class Main {
  public static void main(String[] args) {
    MathUtils m = new MathUtils();
    System.out.println(m.add(5));
  }
}
```

Answer

10

Status : Correct ***Marks : 1/1***

5. What will be the output of the following code?

```
class A {
  int p = 5;
  int q = 2;
}

class Main {
  public static void main(String[] args) {
      A obj = new A();
      System.out.println(obj.p + obj.q);
  }
}
```

Answer

7

Status : Correct ***Marks : 1/1***

6. What will be the output of the following code?

```
class A {
    int val = 20;
}

public class Main {
    public static void main(String[] args) {
        A obj1 = new A();
        A obj2 = obj1;
        obj2.val += 5;
        System.out.println(obj1.val);
    }
}
```

Answer

25

Status : Correct ***Marks : 1/1***

7. What will be the output of the following code?

```
class Alpha {
    void greet(String name) {
        System.out.println("Hello " + name);
    }
}
```

```
public class Main {
   public static void main(String[] args) {
      Alpha obj = new Alpha();
      obj.greet("Anu");
   }
}
```

Answer

Hello Anu

Status : Correct ***Marks : 1/1***

8. What will be the output of the following code?

```
class Box {
   int volume(int l, int b, int h) {
      return l * b * h;
   }
}

public class Main {
   public static void main(String[] args) {
      Box b = new Box();
      System.out.println(b.volume(2, 3, 4));
   }
}
```

Answer

24

Status : Correct ***Marks : 1/1***

9. What will be the output of the following code?

```
class Demo {
   void printMessage() {
      System.out.println("Hello from Demo");
   }
}
```

```
public class Main {
   public static void main(String[] args) {
      Demo d = new Demo();
      d.printMessage();
   }
}
```

Answer

Compilation error

Status : Wrong ***Marks : 0/1***

10. What is the output of the following code?

```
class Box {
   int height;
   Box(int height) {
      this.height = height;
   }
   void modifyHeight(Box b) {
      b.height += 10;
   }
}
public class Main {
   public static void main(String[] args) {
      Box b1 = new Box(20);
      b1.modifyHeight(b1);
      System.out.println(b1.height);
   }
}
```

Answer

30

Status : Correct ***Marks : 1/1***

11. What will be the output of the following code?

```
class Test {
   private int value;
   Test(int value) {
      this.value = value;
   }
   public int getValue() {
      return value;
   }
}
public class Main {
   public static void main(String[] args) {
      Test obj = new Test(10);
      System.out.println(obj.value);
   }
}
```

Answer

Compile-time error

Status : Correct ***Marks : 1/1***

12. What will be the output of the following code?

```
class Ball {
   int size = 11;
}

class Game {
   public static void main(String[] args) {
      Ball b1 = new Ball();
      Ball b2 = new Ball();
      b2.size = 10;
      System.out.println(b1.size);
   }
}
```

Answer

11

Status **:** Correct ***Marks : 1/1***

13. What will be the output of the following code?

```
class Person {
   int age = 18;
}

public class Main {
   public static void main(String[] args) {
      Person p = new Person();
      p.age += 2;
      System.out.println("Age: " + p.age);
   }
}
```

Answer

Age: 20

Status **:** Correct ***Marks : 1/1***

14. What will be the output of the following code?

```
class A {
   int y = 30;
}

public class Main {
   public static void main(String[] args) {
      A a1 = new A();
      A a2 = new A();
      a1.y = 50;
      System.out.println(a2.y);
   }
}
```

Answer

50

Status : Wrong ***Marks : 0/1***

15. What will be the output of the following code?

```
class Sample {
   int x = 10;

   void display() {
      System.out.println("x = " + x);
   }

   public static void main(String[] args) {
      Sample s = new Sample();
      s.display();
   }
}
```

Answer

x = 10

Status : Correct ***Marks : 1/1***

Rajalakshmi Engineering College

Name: Jovitha Sheethal B
Email: 241001099@rajalakshmi.edu.in
Roll no: 241001099
Phone: 6385334710
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 5_Q2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

You are working as a developer for CityBank, which wants to build a basic account management system.

Each customer at the bank has:

An Account Number (integer)A Customer Name (string)An Initial Balance (double)

The bank allows two types of transactions:

Deposit – increases the balance.Withdrawal – decreases the balance only if enough funds are available.

If the withdrawal amount is greater than the balance, the withdrawal should not happen, and the balance should remain the same.

You are required to implement this system using:

A class with attributes for account details.A constructor to initialize account details.Setter methods to update details if needed.Getter methods to retrieve details.Objects of the class to represent customers.

Finally, display each customer's account details after all transactions.

Input Format

The first line of input contains an integer N, representing the number of customers.

For each customer:

- The next line contains the account number (integer).
- The following line contains the customer name (string).
- The next line contains the initial balance (double).
- The next line contains the deposit amount (double).
- The next line contains the withdrawal amount (double).

Output Format

For each customer, print the details in the following format:

1. Account Number: <account_number>
2. Customer Name: <customer_name>
3. Final Balance: <final_balance> (rounded to one decimal place)

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
1234
Rahul Sharma
5000
2000
3000
Output: Account Number: 1234
Customer Name: Rahul Sharma

Final Balance: 4000.0

Answer

```
// You are using Java
import java.util.Scanner;

class Account {
   private int accountNumber;
   private String customerName;
   private double balance;


   public Account(int accountNumber, String customerName, double
initialBalance) {
      this.accountNumber = accountNumber;
      this.customerName = customerName;
      this.balance = initialBalance;
   }


   public void setAccountNumber(int accountNumber) {
      this.accountNumber = accountNumber;
   }

   public void setCustomerName(String customerName) {
      this.customerName = customerName;
   }

   public void setBalance(double balance) {
      this.balance = balance;
   }


   public int getAccountNumber() {
      return accountNumber;
   }

   public String getCustomerName() {
      return customerName;
   }

   public double getBalance() {
        return balance;
    }


    public void deposit(double amount) {
        if (amount >= 0) {
            balance += amount;
        }
    }


    public void withdraw(double amount) {
        if (amount >= 0 && amount <= balance) {
            balance -= amount;
        }
    }
}

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

        int N = Integer.parseInt(sc.nextLine());

        Account[] customers = new Account[N];

        for (int i = 0; i < N; i++) {
            int accountNumber = Integer.parseInt(sc.nextLine().trim());
            String customerName = sc.nextLine().trim();
            double initialBalance = Double.parseDouble(sc.nextLine().trim());
            double depositAmount = Double.parseDouble(sc.nextLine().trim());
            double withdrawalAmount = Double.parseDouble(sc.nextLine().trim());

            Account account = new Account(accountNumber, customerName,
initialBalance);
            account.deposit(depositAmount);
            account.withdraw(withdrawalAmount);

            customers[i] = account;
        }

        for (Account customer : customers) {
            System.out.println("Account Number: " + customer.getAccountNumber());
            System.out.println("Customer Name: " + customer.getCustomerName());
            System.out.printf("Final Balance: %.1f%n", customer.getBalance());
        }

        sc.close();
    }
}
```

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

Rajalakshmi Engineering College

Name: Jovitha Sheethal B
Email: 241001099@rajalakshmi.edu.in
Roll no: 241001099
Phone: 6385334710
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 5_Q3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Neha is working as a developer for CityElectricity Board, which wants to build a household electricity billing system.

Each customer's electricity account has:

A Customer ID (integer)A Customer Name (string)Units Consumed (double)

The electricity bill is calculated based on these rules:

For the first 100 units   5 units charge per unitFor the next 100 units (101–200)   7 units charge per unitFor units above 200   10 units charge per unitIf the total bill exceeds 2000 units, a 5% discount is applied on the final bill.

Neha has been asked to implement this system using:

A class with attributes for customer details.A constructor to initialize customer details.Setter methods to update details if needed.Getter methods to retrieve details.Objects of the class to represent customers.

Finally, display each customer's details and final bill amount.

Input Format

The first line of input contains an integer N, representing the number of customers.

For each customer:

- The next line contains the Customer ID (integer).
- The following line contains the Customer Name (string).
- The next line contains the Units Consumed (double).

Output Format

For each customer, print the details in the following format:

Customer ID: <customer_id>

Customer Name: <customer_name>

Final Bill: <final_bill> (rounded to one decimal place)

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
1001
Ravi Kumar
80

Output: Customer ID: 1001
Customer Name: Ravi Kumar
Final Bill: 400.0

Answer

```
import java.util.Scanner;

class Customer {
   private int customerId;
   private String customerName;
   private double unitsConsumed;


   public Customer(int customerId, String customerName, double
unitsConsumed) {
      this.customerId = customerId;
      this.customerName = customerName;
      this.unitsConsumed = unitsConsumed;
   }


   public void setCustomerId(int customerId) {
      this.customerId = customerId;
   }

   public void setCustomerName(String customerName) {
      this.customerName = customerName;
   }

   public void setUnitsConsumed(double unitsConsumed) {
      this.unitsConsumed = unitsConsumed;
   }


   public int getCustomerId() {
      return customerId;
   }

   public String getCustomerName() {
      return customerName;
   }

   public double getUnitsConsumed() {
      return unitsConsumed;
   }
```

```
    public double calculateBill() {
        double units = unitsConsumed;
        double bill = 0;

        if (units <= 100) {
            bill = units * 5;
        } else if (units <= 200) {
            bill = 100 * 5 + (units - 100) * 7;
        } else {
            bill = 100 * 5 + 100 * 7 + (units - 200) * 10;
        }

        if (bill > 2000) {
            bill = bill * 0.95;
        }

        return bill;
    }
}

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

        int N = Integer.parseInt(sc.nextLine());

        for (int i = 0; i < N; i++) {
            int customerId = Integer.parseInt(sc.nextLine().trim());
            String customerName = sc.nextLine().trim();
            double unitsConsumed = Double.parseDouble(sc.nextLine().trim());

            Customer customer = new Customer(customerId, customerName,
unitsConsumed);

            System.out.println("Customer ID: " + customer.getCustomerId());
            System.out.println("Customer Name: " + customer.getCustomerName());
            System.out.printf("Final Bill: %.1f%n", customer.calculateBill());
        }

        sc.close();
    }
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: Jovitha Sheethal B
Email: 241001099@rajalakshmi.edu.in
Roll no: 241001099
Phone: 6385334710
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 6_Q3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Preethi is working on a project to automate sales tax calculations for items in a store. She wants to create a program that takes the price of an item and the sales tax rate as input and calculates the final price of the item after applying the sales tax.

Write a program using the class SalesTaxCalculator, which contains an overloaded method named calculateFinalPrice to handle both integer and double inputs. The program should also include a Main class that takes user input, calls the appropriate method from SalesTaxCalculator, and prints the final price of the item.

Formula Used: Final price = price + ((price * sales tax rate) / 100)

Input Format

The first line of input consists of an integer price (the price of the item for integer inputs).

The second line of input consists of an integer taxRate (the sales tax rate for integer inputs).

The third line of input consists of a double price (the price of the item for double inputs).

The fourth line of input consists of a double taxRate (the sales tax rate for double inputs).

Output Format

The first line of output prints an integer, representing the final price of the item after applying the sales tax for integer inputs (a and b).

The second line prints a double value, representing the final price of the item after applying the sales tax for double-value inputs (m and n), rounded to two decimal places.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 100
10
100.0
5.0

Output: 110
105.00

Answer

```
import java.util.Scanner;


class SalesTaxCalculator {
   // Overloaded method for integer inputs
   public static int calculateFinalPrice(int price, int taxRate) {
      return price + (price * taxRate / 100);
```

```
    }

    // Overloaded method for double inputs
    public static double calculateFinalPrice(double price, double taxRate) {
        return price + (price * taxRate / 100);
    }
}

class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int intPrice = scanner.nextInt();
        int intTaxRate = scanner.nextInt();
        double doublePrice = scanner.nextDouble();
        double doubleTaxRate = scanner.nextDouble();

        int finalPriceInt = SalesTaxCalculator.calculateFinalPrice(intPrice,
intTaxRate);
        double finalPriceDouble =
SalesTaxCalculator.calculateFinalPrice(doublePrice, doubleTaxRate);

        System.out.println(finalPriceInt);
        System.out.format("%.2f", finalPriceDouble);
    }
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: Jovitha Sheethal B
Email: 241001099@rajalakshmi.edu.in
Roll no: 241001099
Phone: 6385334710
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 5_Q5

Attempt : 1
Total Mark : 10
Marks Obtained : 0

Section 1 : Coding

1. Problem Statement

Ram is working as a developer for BrightEdu Coaching Center, which wants to build a student fee management system.

Each student’s enrollment has:

An Enrollment ID (integer)A Student Name (string)The Number of Subjects (integer)

The fee calculation rules are:

Registration Fee = 1000 units (flat for every student).Per Subject Fee = 800 units.If the student enrolls in more than 5 subjects, a 20% scholarship (discount) is applied on the total fee.

Ram has been asked to implement this system using:

A class with attributes for student details.A constructor to initialize student details.Setter methods to update details if needed.Getter methods to retrieve details.Objects of the class to represent student enrollments.

Finally, display each student's details and final fee.

Input Format

The first line of input contains an integer N, representing the number of students.

For each student:

- The next line contains the Enrollment ID (integer).
- The following line contains the student's name (string).
- The next line contains the Number of subjects (integer).

Output Format

For each student, print the details in the following format:

- Enrollment ID: <enrollment_id>
- Student Name: <student_name>
- Final Fee: <final_fee> (rounded to one decimal place)

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
1234
Ravi Kumar
3

Output: Enrollment ID: 1234
Student Name: Ravi Kumar
Final Fee: 3400.0

Answer

```
import java.util.Scanner;

class Student {
    private int enrollmentId;
    private String studentName;
    private int numberOfSubjects;


    public Student(int enrollmentId, String studentName, int numberOfSubjects) {
       this.enrollmentId = enrollmentId;
       this.studentName = studentName;
       this.numberOfSubjects = numberOfSubjects;
    }


    public void setEnrollmentId(int enrollmentId) {
       this.enrollmentId = enrollmentId;
    }

    public void setStudentName(String studentName) {
       this.studentName = studentName;
    }

    public void setNumberOfSubjects(int numberOfSubjects) {
       this.numberOfSubjects = numberOfSubjects;
    }


    public int getEnrollmentId() {
       return enrollmentId;
    }

    public String getStudentName() {
       return studentName;
    }

    public int getNumberOfSubjects() {
       return numberOfSubjects;
    }


    public double calculateFee() {
       double registrationFee = 1000;
       double perSubjectFee = 800 * numberOfSubjects;
       double totalFee = registrationFee + perSubjectFee;
```

```
        if (numberOfSubjects > 5) {
            totalFee = totalFee * 0.80;
        }

        return totalFee;
    }
}

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

        int N = Integer.parseInt(sc.nextLine());

        for (int i = 0; i < N; i++) {
            int enrollmentId = Integer.parseInt(sc.nextLine().trim());
            String studentName = sc.nextLine().trim();
            int numberOfSubjects = Integer.parseInt(sc.nextLine().trim());

            Student student = new Student(enrollmentId, studentName,
numberOfSubjects);

            System.out.println("Enrollment ID: " + student.getEnrollmentId());
            System.out.println("Student Name: " + student.getStudentName());
            System.out.printf("Final Fee: %.1f%n", student.calculateFee());
        }

        sc.close();
    }
}
```

Status : Skipped ***Marks : 0/10***

Rajalakshmi Engineering College

Name: Jovitha Sheethal B
Email: 241001099@rajalakshmi.edu.in
Roll no: 241001099
Phone: 6385334710
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

Scan to verify results

2024_28_III_OOPS Using Java Lab

2028_REC_OOPS using Java_Week 5_PAH

Attempt : 1
Total Mark : 50
Marks Obtained : 50

Section 1 : Coding

1. Problem Statement

Neha is working as a developer for CityQuiz Platform, which wants to build a system to calculate quiz scores and identify top scorers among participants.

Each participant's record has:

Participant ID (integer)Participant Name (string)An array of scores in 5 quiz rounds (integers, each between 0 and 100)

The system must calculate:

Total Score = sum of scores in all 5 rounds.Average Score = Total Score ÷ 5.If a participant scores above 80 in all rounds, a bonus of 10 points is added to the total score.Identify the Top Scorer among all participants. If two participants have the same total score, the one with the lower Participant ID is considered the top scorer.

Neha has been asked to implement this system using:

A class with attributes for participant details.A constructor to initialize participant details.Getter and setter methods to retrieve or update participant details.A method to calculate total score and average score (including bonus if applicable).Objects of the class to represent participants.

Finally, display each participant’s details and announce the Top Scorer.

Input Format

The first line of input contains an integer N, representing the number of participants.

For each participant:

- Next line: Participant ID (integer)
- Next line: Participant Name (string)
- Next line: 5 integers separated by spaces (scores for 5 quiz rounds)

Output Format

For each participant:

- Participant ID: <participant_id>
- Participant Name: <participant_name>
- Total Score: <total_score>
- Average Score: <average_score>

Finally, print "Top Scorer: <participant_name> with <total_score> points"

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
1001
Ravi Kumar
85 90 88 92 87

Output: Participant ID: 1001
Participant Name: Ravi Kumar
Total Score: 452
Average Score: 90
Top Scorer: Ravi Kumar with 452 points

Answer

```
import java.util.Scanner;

class Participant {
   private int participantId;
   private String participantName;
   private int[] scores = new int[5];
   private int totalScore;
   private int averageScore;

   // Constructor
   public Participant(int participantId, String participantName, int[] scores) {
      this.participantId = participantId;
      this.participantName = participantName;
      this.scores = scores;
      calculateScores();
   }

   // Setter methods
   public void setParticipantId(int participantId) {
      this.participantId = participantId;
   }

   public void setParticipantName(String participantName) {
      this.participantName = participantName;
   }

   public void setScores(int[] scores) {
      this.scores = scores;
      calculateScores();
   }
```

```
    // Getter methods
    public int getParticipantId() {
        return participantId;
    }

    public String getParticipantName() {
        return participantName;
    }

    public int getTotalScore() {
        return totalScore;
    }

    public int getAverageScore() {
        return averageScore;
    }

    // Calculate total and average score, including bonus if applicable
    private void calculateScores() {
        totalScore = 0;
        boolean bonusEligible = true;

        for (int score : scores) {
            totalScore += score;
            if (score <= 80) {
                bonusEligible = false;
            }
        }

        if (bonusEligible) {
            totalScore += 10;
        }

        averageScore = totalScore / 5;
    }
}

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

        int N = Integer.parseInt(sc.nextLine().trim());
        Participant topScorer = null;

        for (int i = 0; i < N; i++) {
            int id = Integer.parseInt(sc.nextLine().trim());
            String name = sc.nextLine().trim();

            String[] scoreStrs = sc.nextLine().trim().split("\\s+");
            int[] scores = new int[5];
            for (int j = 0; j < 5; j++) {
                scores[j] = Integer.parseInt(scoreStrs[j]);
            }

            Participant participant = new Participant(id, name, scores);

            System.out.println("Participant ID: " + participant.getParticipantId());
            System.out.println("Participant Name: " +
participant.getParticipantName());
            System.out.println("Total Score: " + participant.getTotalScore());
            System.out.println("Average Score: " + participant.getAverageScore());

            // Determine top scorer
            if (topScorer == null) {
                topScorer = participant;
            } else {
                if (participant.getTotalScore() > topScorer.getTotalScore() ||
                    (participant.getTotalScore() == topScorer.getTotalScore() &&
                     participant.getParticipantId() < topScorer.getParticipantId())) {
                    topScorer = participant;
                }
            }
        }

        System.out.println("Top Scorer: " + topScorer.getParticipantName() + " with "
+ topScorer.getTotalScore() + " points");

        sc.close();
    }
}
```

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

2. Problem Statement

Neha is working as a developer for CityMovie Theatre, which wants to build a system to calculate total ticket cost for movie-goers based on the number of tickets and type of seats booked.

Each customer's booking has:

Booking ID (integer)Customer Name (string)Number of Tickets (integer)Seat Type (string: "Standard", "Premium", "VIP")

The ticket prices are:

Standard – 250 units per ticketPremium – 400 units per ticketVIP – 600 units per ticket

The calculation rules:

Total Amount = Number of Tickets × Seat Price

If a customer books more than 4 tickets, they get a 10% discount on the total amount.

If the booking is for VIP seats and the total amount exceeds 3000 units, a 5% luxury tax is added after any discount.

Neha has been asked to implement this system using:

A class with attributes for booking details.A constructor to initialize booking details.Getter and Setter methods to retrieve and update booking details if required.A method to calculate the final ticket cost.Objects of the class to represent bookings.

Finally, display each customer's details and final ticket amount.

Input Format

The first line contains an integer N, representing the number of bookings.

For each booking:

- The next line contains the Booking ID (integer).
- The next line contains the Customer Name (string).

- The next line contains Number of Tickets (integer).
- The next line contains Seat Type ("Standard", "Premium", or "VIP").

Output Format

For each booking, print:

- Booking ID: <booking_id>
- Customer Name: <customer_name>
- Final Ticket Amount: <final_amount> (rounded to one decimal place)

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
1001
Ravi Kumar
3
Standard

Output: Booking ID: 1001
Customer Name: Ravi Kumar
Final Ticket Amount: 750.0

Answer

```
import java.util.Scanner;

class Booking {
   private int bookingId;
   private String customerName;
   private int numberOfTickets;
   private String seatType;

   // Constructor
   public Booking(int bookingId, String customerName, int numberOfTickets,
String seatType) {
      this.bookingId = bookingId;
      this.customerName = customerName;
      this.numberOfTickets = numberOfTickets;
      this.seatType = seatType;
   }
```

```
public void setBookingId(int bookingId) {
    this.bookingId = bookingId;
}

public void setCustomerName(String customerName) {
    this.customerName = customerName;
}

public void setNumberOfTickets(int numberOfTickets) {
    this.numberOfTickets = numberOfTickets;
}

public void setSeatType(String seatType) {
    this.seatType = seatType;
}


public int getBookingId() {
    return bookingId;
}

public String getCustomerName() {
    return customerName;
}

public int getNumberOfTickets() {
    return numberOfTickets;
}

public String getSeatType() {
    return seatType;
}


public double calculateFinalAmount() {
    double pricePerTicket = 0;

    switch (seatType) {
        case "Standard":
            pricePerTicket = 250;
            break;
          case "Premium":
            pricePerTicket = 400;
            break;
          case "VIP":
            pricePerTicket = 600;
            break;
          default:
            pricePerTicket = 0;
        }

        double totalAmount = numberOfTickets * pricePerTicket;


        if (numberOfTickets > 4) {
          totalAmount = totalAmount * 0.90;
        }


        if (seatType.equals("VIP") && totalAmount > 3000) {
          totalAmount = totalAmount * 1.05;
        }

        return totalAmount;
      }
    }

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

        int N = Integer.parseInt(sc.nextLine().trim());

        for (int i = 0; i < N; i++) {
          int bookingId = Integer.parseInt(sc.nextLine().trim());
          String customerName = sc.nextLine().trim();
          int numberOfTickets = Integer.parseInt(sc.nextLine().trim());
          String seatType = sc.nextLine().trim();

          Booking booking = new Booking(bookingId, customerName,
    numberOfTickets, seatType);
```

```
            System.out.println("Booking ID: " + booking.getBookingId());
            System.out.println("Customer Name: " + booking.getCustomerName());
            System.out.printf("Final Ticket Amount: %.1f%n",
booking.calculateFinalAmount());
        }

        sc.close();
    }
}
```

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

3. Problem Statement

Ravi is working as a developer for SecureLogin Systems, which wants to build a system to evaluate the strength of user passwords.

Each user record has:

User ID (integer)User Name (string)Password (string)

The system must calculate whether a password is strong or weak.

A password is considered strong if it meets all of the following conditions:

At least 8 characters long.Contains at least one uppercase letter.Contains at least one lowercase letter.Contains at least one digit.Contains at least one special character (from !@#$%^&amp;*).

Ravi has been asked to implement this system using:

A class with attributes for user details.A constructor to initialize user details.Getter and setter methods to retrieve or update user details.A method to check whether the password is strong.Objects of the class to represent users.

Finally, display each user’s details and indicate whether their password is Strong or Weak.

Input Format

The first line contains an integer N, representing the number of users.

For each user:

The next line contains the User ID (integer).

The next line contains the User Name (string).

The next line contains the Password (string).

Output Format

For each user, print the details in the following format:

User ID: <user_id>

User Name: <user_name>

Password: <password>

Password Strength: <Strong/Weak>

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
1001
Ravi Kumar
Abc@1234

Output: User ID: 1001
User Name: Ravi Kumar
Password: Abc@1234
Password Strength: Strong

Answer

```
// You are using Java
import java.util.Scanner;

class User {
   private int userId;
```

```
    private String userName;
    private String password;

    public User(int userId, String userName, String password) {
        this.userId = userId;
        this.userName = userName;
        this.password = password;
    }


    public void setUserId(int userId) {
        this.userId = userId;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }

    public void setPassword(String password) {
        this.password = password;
    }


    public int getUserId() {
        return userId;
    }

    public String getUserName() {
        return userName;
    }

    public String getPassword() {
        return password;
    }


    public boolean isPasswordStrong() {
        String pwd = this.password;

        if (pwd.length() < 8) return false;

        boolean hasUpper = false, hasLower = false, hasDigit = false, hasSpecial =
false;
```

```
        for (char ch : pwd.toCharArray()) {
            if (Character.isUpperCase(ch)) hasUpper = true;
            else if (Character.isLowerCase(ch)) hasLower = true;
            else if (Character.isDigit(ch)) hasDigit = true;
            else if ("!@#$%^&*".indexOf(ch) != -1) hasSpecial = true;
        }

        return hasUpper && hasLower && hasDigit && hasSpecial;
    }
}

class SecureLoginSystem {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int N = Integer.parseInt(sc.nextLine().trim());

        for (int i = 0; i < N; i++) {
            int userId = Integer.parseInt(sc.nextLine().trim());
            String userName = sc.nextLine().trim();
            String password = sc.nextLine().trim();

            User user = new User(userId, userName, password);

            System.out.print("User ID: " + user.getUserId() + " ");
            System.out.print("User Name: " + user.getUserName() + " ");
            System.out.print("Password: " + user.getPassword() + " ");
            System.out.print("Password Strength: " + (user.isPasswordStrong() ?
"Strong" : "Weak"));
            System.out.println();
        }

        sc.close();
    }
}
```

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

4. Problem Statement

Anjali is working as a developer for CityFitness Gym, which wants to build a system to calculate monthly membership fees for gym members based on the type of membership and the number of personal training sessions booked.

Each member’s record has:

Member ID (integer)Member Name (string)Membership Type (string: "Basic", "Premium", "Elite")Number of Personal Training Sessions (integer)

The monthly fees are:

Basic – 1000 unitsPremium – 1500 unitsElite – 2000 units

The cost of personal training sessions is 500 units per session.

The calculation rules:

Total Amount = Membership Fee + (Number of Personal Training Sessions × 500)If the number of sessions is more than 5, a 10% discount is applied on the total amount.If the member has Elite membership and the total amount exceeds 4000, an additional 5% service tax is added after discount.

Anjali has been asked to implement this system using:

A class with attributes for member details.A constructor to initialize member details.Getter and Setter methods to retrieve and update member details if required.A method to calculate the final monthly fee.Objects of the class to represent members.

Finally, display each member’s details and the final monthly fee.

Input Format

The first line contains an integer N, representing the number of members.

For each member:

- Next line contains Member ID (integer)
- Next line contains Member Name (string)
- Next line contains Membership Type ("Basic", "Premium", "Elite")
- Next line contains Number of Personal Training Sessions (integer)

Output Format

For each member, print:

- Member ID: <member_id>
- Member Name: <member_name>
- Final Monthly Fee: <final_fee> (The final fee must be rounded to one decimal place)

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
1001
Ravi Kumar
Basic
3

Output: Member ID: 1001
Member Name: Ravi Kumar
Final Monthly Fee: 2500.0

Answer

```
// You are using Java
import java.util.Scanner;

class Member {
   private int memberId;
   private String memberName;
   private String membershipType;
   private int personalTrainingSessions;

   // Constructor
   public Member(int memberId, String memberName, String membershipType,
int personalTrainingSessions) {
       this.memberId = memberId;
       this.memberName = memberName;
       this.membershipType = membershipType;
       this.personalTrainingSessions = personalTrainingSessions;
   }

   // Getters and Setters
```

```
public int getMemberId() {
   return memberId;
}

public void setMemberId(int memberId) {
   this.memberId = memberId;
}

public String getMemberName() {
   return memberName;
}

public void setMemberName(String memberName) {
   this.memberName = memberName;
}

public String getMembershipType() {
   return membershipType;
}

public void setMembershipType(String membershipType) {
   this.membershipType = membershipType;
}

public int getPersonalTrainingSessions() {
   return personalTrainingSessions;
}

public void setPersonalTrainingSessions(int personalTrainingSessions) {
   this.personalTrainingSessions = personalTrainingSessions;
}

// Method to calculate final monthly fee
public double calculateFinalFee() {
   double membershipFee = 0;

   switch (membershipType) {
      case "Basic":
         membershipFee = 1000;
         break;
      case "Premium":
         membershipFee = 1500;
```

```
            break;
         case "Elite":
            membershipFee = 2000;
            break;
         default:
            // Should not happen based on input constraints
            membershipFee = 0;
      }

      double totalAmount = membershipFee + (personalTrainingSessions * 500);

      // 10% discount if sessions > 5
      if (personalTrainingSessions > 5) {
         totalAmount *= 0.90;
      }

      // For Elite membership, if total exceeds 4000, add 5% service tax
      if (membershipType.equals("Elite") && totalAmount > 4000) {
         totalAmount *= 1.05;
      }

      return totalAmount;
   }
}

class CityFitnessGym {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int N = Integer.parseInt(sc.nextLine().trim());

      for (int i = 0; i < N; i++) {
         int memberId = Integer.parseInt(sc.nextLine().trim());
         String memberName = sc.nextLine().trim();
         String membershipType = sc.nextLine().trim();
         int personalTrainingSessions = Integer.parseInt(sc.nextLine().trim());

         Member member = new Member(memberId, memberName,
membershipType, personalTrainingSessions);

         System.out.print("Member ID: " + member.getMemberId() + " ");
         System.out.print("Member Name: " + member.getMemberName() + " ");
         System.out.printf("Final Monthly Fee: %.1f%n",
member.calculateFinalFee());
```

```
        }

        sc.close();
    }
}
```

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

5. Problem Statement

Each customer at the bank has an Account Number, Customer Name, and an Initial Balance. The bank allows two types of transactions:

Deposit – Increases the balance.Withdrawal – Decreases the balance, but only if enough funds are available. If the withdrawal amount exceeds the available balance, the transaction should be skipped, and the balance should remain unchanged.

You are required to implement this banking system by:

Creating a class with the necessary attributes to store account details.

Using a constructor to initialize the account details when a new account is created.Providing setter methods to update the details if required.Providing getter methods to retrieve account details.Creating objects of this class to represent different customers, where each customer can perform deposits and withdrawals.

Instructions:

Implement the class to store account details.Implement the logic for performing deposit and withdrawal transactions.Ensure that withdrawals don't exceed the available balance.After performing the transactions, print the account number, customer name, and final balance.

Input Format

The first line of input contains an integer N, representing the number of customers.

For each customer:

- The next line contains the account number (integer).
- The following line contains the customer name (string).
- The next line contains the initial balance (double).
- The next line contains the deposit amount (double).
- The next line contains the withdrawal amount (double).

Output Format

For each customer, print the details in the following format:

1. Account Number: <account_number>
2. Customer Name: <customer_name>
3. Final Balance: <final_balance> (rounded to one decimal place)

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1
1234
Rahul Sharma
5000
2000
3000
Output: Account Number: 1234
Customer Name: Rahul Sharma
Final Balance: 4000.0

Answer

```
import java.util.Scanner;

class BankAccount {
   private int accountNumber;
   private String customerName;
   private double balance;

   // Constructor
   public BankAccount(int accountNumber, String customerName, double
initialBalance) {
       this.accountNumber = accountNumber;
```

```
        this.customerName = customerName;
        this.balance = initialBalance;
    }

    // Getters
    public int getAccountNumber() {
        return accountNumber;
    }

    public String getCustomerName() {
        return customerName;
    }

    public double getBalance() {
        return balance;
    }

    // Setters
    public void setAccountNumber(int accountNumber) {
        this.accountNumber = accountNumber;
    }

    public void setCustomerName(String customerName) {
        this.customerName = customerName;
    }

    // Deposit method
    public void deposit(double amount) {
        if (amount >= 0) {
            balance += amount;
        }
    }

    // Withdrawal method
    public void withdraw(double amount) {
        if (amount >= 0 && amount <= balance) {
            balance -= amount;
        }
        // If withdrawal amount > balance, ignore withdrawal
    }
}
```

```
class BankingSystem {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int N = Integer.parseInt(sc.nextLine().trim());

    for (int i = 0; i < N; i++) {
      int accountNumber = Integer.parseInt(sc.nextLine().trim());
      String customerName = sc.nextLine().trim();
      double initialBalance = Double.parseDouble(sc.nextLine().trim());
      double depositAmount = Double.parseDouble(sc.nextLine().trim());
      double withdrawalAmount = Double.parseDouble(sc.nextLine().trim());

      BankAccount account = new BankAccount(accountNumber,
customerName, initialBalance);
      account.deposit(depositAmount);
      account.withdraw(withdrawalAmount);

      System.out.print("Account Number: " + account.getAccountNumber() + "
");
      System.out.print("Customer Name: " + account.getCustomerName() + " ");
      System.out.printf("Final Balance: %.1f%n", account.getBalance());
    }

    sc.close();
  }
}
```

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

Rajalakshmi Engineering College

Name: Jovitha Sheethal B
Email: 241001099@rajalakshmi.edu.in
Roll no: 241001099
Phone: 6385334710
Branch: REC
Department: IT - Section 2
Batch: 2028
Degree: B.E - IT

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

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;

// You are using Java
class CircleUtils {

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

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

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

```
  public 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_**

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;
   }

   public double calculateInterest() {
      return 0.0;
   }
}

class FixedDeposit extends Account {
   int durationYears;
   double rateOfInterest;
```

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

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

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

    public RecurringDeposit(String accountHolder, int monthlyDeposit, int
durationMonths, double rateOfInterest) {
        super(accountHolder, monthlyDeposit * durationMonths); //
principalAmount is maturity amount
        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***

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 >= 10.0) {
         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***