

Rajalakshmi Engineering College

Name: Rahul G
Email: 241901087@rajalakshmi.edu.in
Roll no: 241901087
Phone: 8248227615
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 7_MCQ

Attempt : 1
Total Mark : 15
Marks Obtained : 14

Section 1 : MCQ

1. Which of the following statements about Java interfaces is true?

Answer

A class can implement multiple interfaces.

Status : Correct

Marks : 1/1

2. What is the output of the following code?

```
interface A {  
    static void display() {  
        System.out.println("Static method in A");  
    }  
}
```

```
class B implements A {
    static void display() {
        System.out.println("Static method in B");
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        B.display();
    }
}
```

Answer

Compilation Error

Status : Wrong

Marks : 0/1

3. What is the output of the following code?

```
interface A {
    default void show() {
        System.out.println("A's Default Method");
    }
}
```

```
interface B {
    default void show() {
        System.out.println("B's Default Method");
    }
}
```

```
class C implements A, B {
    public void show() {
        A.super.show();
    }
}
```

```
public class Main {
    public static void main(String[] args) {
```

```
C obj = new C();  
obj.show();  
}  
}
```

Answer

A's Default Method

Status : Correct

Marks : 1/1

4. Consider a class implementing an interface and extending a class, both having a method with the same name. Which method gets called?

Answer

The method from the superclass

Status : Correct

Marks : 1/1

5. What is the output of the following code?

```
interface MathOperations {  
    static int square(int x) {  
        return x * x;  
    }  
}  
  
public class Main {  
    public static void main(String[] args) {  
        System.out.println(MathOperations.square(5));  
    }  
}
```

Answer

25

Status : Correct

Marks : 1/1

6. What happens when an implementing class does not override a default

method from an interface?

Answer

The default method's implementation from the interface will be used.

Status : Correct

Marks : 1/1

7. How do you call a static method from an interface MyInterface?

Answer

MyInterface.staticMethod();

Status : Correct

Marks : 1/1

8. If a class implements two interfaces that have the same default method, what must the class do?

Answer

The class must override the method to resolve ambiguity.

Status : Correct

Marks : 1/1

9. What is the output of the following code?

```
interface A {  
    default void show() {  
        System.out.println("A's Default Method");  
    }  
}
```

```
class B {  
    public void show() {  
        System.out.println("B's Method");  
    }  
}
```

```
class C extends B implements A {
```

```
}  
public class Main {  
    public static void main(String[] args) {  
        C obj = new C();  
        obj.show();  
    }  
}
```

Answer

B's Method

Status : Correct

Marks : 1/1

10. What is the primary purpose of static methods in Java interfaces?

Answer

They allow an interface to provide helper methods without requiring an implementing class.

Status : Correct

Marks : 1/1

11. Which of the following statements is true regarding default methods in Java interfaces?

Answer

A default method can be overridden in a class implementing the interface.

Status : Correct

Marks : 1/1

12. What is the output of the following code?

```
interface X {  
    default void show() {  
        System.out.println("X's Default Method");  
    }  
}
```

```
interface Y {  
    default void show() {  
        System.out.println("Y's Default Method");  
    }  
}
```

```
class Z implements X, Y {  
    public void show() {  
        System.out.println("Z's Method");  
    }  
}
```

```
public class Main {  
    public static void main(String[] args) {  
        Z obj = new Z();  
        obj.show();  
    }  
}
```

Answer

Z's Method

Status : Correct

Marks : 1/1

13. Can a Java interface contain both default and static methods?

Answer

Yes, an interface can have both default and static methods.

Status : Correct

Marks : 1/1

14. How can a class explicitly call a default method from an interface if there is a naming conflict?

Answer

Using InterfaceName.super.methodName();

Status : Correct

Marks : 1/1

15. Which of the following is the correct way to declare an interface in Java?

Answer

```
interface Vehicle { void start();}
```

Status : Correct

Marks : 1/1

Rajalakshmi Engineering College

Name: Rahul G

Email: 241901087@rajalakshmi.edu.in

Roll no: 241901087

Phone: 8248227615

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

2028_REC_OOPS using Java_Week 7_Q1

Attempt : 1

Total Mark : 10

Marks Obtained : 10

Section 1 : Coding

1. Problem Statement:

Rajiv is analyzing the energy consumption in his household and wants to calculate the total cost based on the daily energy usage. He is given the rate per unit of electricity and the energy consumed for multiple days. To structure this calculation efficiently, he decides to use an interface-based approach.

Implement an interface CostCalculator with the necessary methods to retrieve energy details and compute the cost. The calculations should be handled in the EnergyConsumptionTracker class, while the EnergyConsumptionApp class should only handle input and output.

Formula

Energy Cost for one day = Energy Consumed per day * Rate Per Unit

Input Format

The first line of input consists of the rate per unit as an 'R' (a double value).

The second line of input consists of the number of days 'N' (an integer).

The third line of input consists of the daily energy consumption values for each day 'D' (double values), separated by space.

Output Format

The first line of the output prints: "Day-wise Energy Cost:"

The next N lines of the output print the day-wise energy costs(double type) and the total energy cost (double type) in Indian Rupees in the following format: "Day [day_number]: Rs. [energy_cost]"

The last line of the output prints: "Total Energy Cost: Rs. [total_cost]"

Note: energy_cost and total_cost are rounded off to two decimal points

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 0.01

3

10.0 20.0 30.0

Output: Day-wise Energy Cost:

Day 1: Rs. 0.10

Day 2: Rs. 0.20

Day 3: Rs. 0.30

Total Energy Cost: Rs. 0.60

Answer

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

```
// You are using Java
```

```
interface CostCalculator {  
    void getEnergyDetails(Scanner scanner);  
    void calculateAndDisplayCost();  
}
```

```
class EnergyConsumptionTracker implements CostCalculator {  
    private double ratePerUnit;  
    private int numDays;  
    private double[] dailyConsumption;  
  
    public EnergyConsumptionTracker(double ratePerUnit, int numDays) {  
        this.ratePerUnit = ratePerUnit;  
        this.numDays = numDays;  
        this.dailyConsumption = new double[numDays];  
    }  
  
    public void getEnergyDetails(Scanner scanner) {  
        for (int i = 0; i < numDays; i++) {  
            dailyConsumption[i] = scanner.nextDouble();  
        }  
    }  
  
    public void calculateAndDisplayCost() {  
        System.out.println("Day-wise Energy Cost:");  
        double totalCost = 0.0;  
        for (int i = 0; i < numDays; i++) {  
            double energyCost = dailyConsumption[i] * ratePerUnit;  
            totalCost += energyCost;  
            System.out.printf("Day %d: Rs. %.2f%n", i + 1, energyCost);  
        }  
        System.out.printf("Total Energy Cost: Rs. %.2f%n", totalCost);  
    }  
}
```

```
class EnergyConsumptionApp {  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
  
        double ratePerUnit = scanner.nextDouble();  
        int numDays = scanner.nextInt();  
  
        CostCalculator tracker = new EnergyConsumptionTracker(ratePerUnit,  
numDays);
```

```
        tracker.getEnergyDetails(scanner);  
        tracker.calculateAndDisplayCost();  
    scanner.close();  
    }  
}
```

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: Rahul G
Email: 241901087@rajalakshmi.edu.in
Roll no: 241901087
Phone: 8248227615
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

2028_REC_OOPS using Java_Week 7_Q2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Jaheer is working on a health monitoring system to help individuals calculate their Body Mass Index (BMI). He has implemented a basic BMI calculator and an interface called HealthCalculator. It should have a method called calculateBMI.

You are tasked with creating a program that takes weight and height as input, calculates the BMI using the BMI Calculator class, and displays the result. If the height or weight is less than or equal to zero, then return -1.

Formula: $BMI = \text{weight} / (\text{height} * \text{height})$

Input Format

The first line of input consists of a double value W, the person's weight in kilograms.

The second line consists of a double value H, the height of the person in meters.

Output Format

The output displays "BMI: " followed by a double value, representing the calculated BMI, rounded off to two decimal places.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 70.0

1.75

Output: BMI: 22.86

Answer

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

```
interface BMIInterface {  
    double calculateBMI(double weight, double height);  
    String categorizeBMI(double bmi);  
}
```

```
class BMICalculator implements BMIInterface {  
    public double calculateBMI(double weight, double height) {  
        return weight / (height * height);  
    }  
}
```

```
    public String categorizeBMI(double bmi) {  
        if (bmi < 18.5)  
            return "Underweight";  
        else if (bmi < 24.9)  
            return "Normal Weight";  
        else if (bmi < 29.9)  
            return "Overweight";  
        else  
            return "Obese";  
    }  
}
```

```
class Main {
```

```
public static void main(String[] args) {  
    Scanner scanner = new Scanner(System.in);  
  
    double weight = scanner.nextDouble();  
    double height = scanner.nextDouble();  
  
    BMICalculator bmiCalculator = new BMICalculator();  
  
    double bmi = bmiCalculator.calculateBMI(weight, height);  
  
    System.out.printf("BMI: %.2f\n", bmi);  
  
    scanner.close();  
}
```

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: Rahul G

Email: 241901087@rajalakshmi.edu.in

Roll no: 241901087

Phone: 8248227615

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

2028_REC_OOPS using Java_Week 7_Q3

Attempt : 1

Total Mark : 10

Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

A financial analyst, Alex, needs a program to calculate simple interest for various financial transactions. He requires a straightforward tool that takes in the principal amount, interest rate, and time in years and computes the interest.

The formula to be used is: $\text{Interest} = \text{Principal} \times \text{Rate} \times \text{Time} / 100$

Implement this functionality using the InterestCalculator interface and the SimpleInterestCalculator class.

Input Format

The first line of input consists of the principal amount P as a double value.

The second line of input consists of the annual interest rate r as a double value.

The third line of input consists of the number of years t as a positive integer, which is an integer value.

Output Format

The output displays the calculated simple interest in the following format: "Simple Interest: [interest_value]", Here, [interest_value] should be replaced with the actual interest value calculated by the program.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 1000.00

5.00

2

Output: Simple Interest: 100.0

Answer

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

```
interface InterestCalculator {  
    double simpleInterest(double principal, double rate, int time);  
}
```

```
class SimpleInterestCalculator implements InterestCalculator {  
    public double simpleInterest(double principal, double rate, int time) {  
        return (principal * rate * time) / 100;  
    }  
}
```

```
class Main {  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
  
        double principal = scanner.nextDouble();  
  
        double rate = scanner.nextDouble();  
  
        int time = scanner.nextInt();
```



```
InterestCalculator calculator = new SimpleInterestCalculator();  
double interest = calculator.simpleInterest(principal, rate, time);  
  
System.out.println("Simple Interest: " + interest);  
  
    }  
}
```

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: Rahul G
Email: 241901087@rajalakshmi.edu.in
Roll no: 241901087
Phone: 8248227615
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

2028_REC_OOPS using Java_Week 7_Q4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Maria, a software developer, is working on an inventory management system project using Java that utilizes an inventory interface to manage a store's products.

The interface should define two methods: `addProduct`, which adds a product by accepting its name, price, and quantity, and `calculateTotalValue`, which computes the total value of all products in the inventory. Implement the interface in a class called `SimpleInventory`, which internally manages a list of `Product` objects.

Each `Product` object should encapsulate the product's name, price, and quantity and include a method to calculate its value as $\text{price} \times \text{quantity}$. The system should allow users to dynamically add products to the inventory and calculate the total value of all products stored.

Help Maria achieve the task.

Input Format

The first line of input consists of an integer to choose one of the following options:

- 1 - to add a product to the inventory.
- 2 - to calculate and view the total inventory value.
- 3 - to exit the program.

For Choice 1 (Add Product):

The next input line is the string representing the product name as a string (single or multi-word, without quotes).

The next line is a double value representing the price as a decimal value

The next line is an integer value representing the quantity as an integer

For Choices 2 and 3, no additional input is required

Output Format

The output displays the results of the commands as follows:

- For the addProduct command, the program should display "Product added to inventory."
- For choice 2, the program should display "Total inventory value [totalvalue].
"The total value should be displayed with one decimal place. If there is no product in the inventory, print the total as 0.0.
- For choice 3, the program should exit

If the choice is not 1, 2, or 3, then print "Invalid choice. Please select a valid option (1/2/3).".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 1

Laptop

800.0

3

2

5

3

Output: Product added to inventory.

Total inventory value: \$2400.0

Invalid choice. Please select a valid option (1/2/3).

Answer

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

```
interface Inventory {  
    void addProduct(String name, double price, int quantity);  
    double calculateTotalValue();  
}
```

```
class Product {  
    private String name;  
    private double price;  
    private int quantity;
```

```
    public Product(String name, double price, int quantity) {  
        this.name = name;  
        this.price = price;  
        this.quantity = quantity;  
    }
```

```
    public double getValue() {  
        return price * quantity;  
    }  
}
```

```
class SimpleInventory implements Inventory {  
    private Product[] products;  
    private int count;
```

```
    public SimpleInventory(int capacity) {  
        products = new Product[capacity];
```

```

        count = 0;
    }

    public void addProduct(String name, double price, int quantity) {
        if (count < products.length) {
            products[count++] = new Product(name, price, quantity);
            System.out.println("Product added to inventory.");
        }
    }

    // Return the total value; do NOT print here (footer will print it once)
    public double calculateTotalValue() {
        double total = 0.0;
        for (int i = 0; i < count; i++) {
            total += products[i].getValue();
        }
        return total;
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Inventory inventory = new SimpleInventory(10);
        while (true) {
            int choice = scanner.nextInt();
            if (choice == 1) {
                scanner.nextLine();
                String productName = scanner.nextLine();
                double price = scanner.nextDouble();
                int quantity = scanner.nextInt();
                inventory.addProduct(productName, price, quantity);
            } else if (choice == 2) {
                double totalValue = inventory.calculateTotalValue();
                System.out.println("Total inventory value: $" + totalValue);
            } else if (choice == 3) {
                break;
            } else {
                System.out.println("Invalid choice. Please select a valid option (1/2/3).");
            }
        }
        scanner.close();
    }
}

```

}
}
Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: Rahul G
Email: 241901087@rajalakshmi.edu.in
Roll no: 241901087
Phone: 8248227615
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

2028_REC_OOPS using Java_Week 7_Q5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Raj is curious about how old he is in the current year.

He has asked you to create a simple program that calculates a person's age based on their birth year. You decide to implement this functionality using the AgeCalculator interface and the HumanAgeCalculator class.

Note: The current year is 2024. Calculate the current age by using the formula: current year - birth year.

Input Format

The input consists of an integer representing the birth year.

Output Format

The output displays "You are X years old." where X is an integer representing the calculated age based on the entered birth year.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1934

Output: You are 90 years old.

Answer

```
import java.util.Scanner;

// You are using Java
interface AgeCalculator {
    int calculateAge(int birthYear);
}

class HumanAgeCalculator implements AgeCalculator {
    public int calculateAge(int birthYear) {
        int currentYear = 2024;
        return currentYear - birthYear;
    }
}

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

        AgeCalculator ageCalculator = new HumanAgeCalculator();

        int birthYear = scanner.nextInt();
        int age = ageCalculator.calculateAge(birthYear);

        System.out.println("You are " + age + " years old.");
    }
}
```

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: Rahul G
Email: 241901087@rajalakshmi.edu.in
Roll no: 241901087
Phone: 8248227615
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 7_PAH

Attempt : 1
Total Mark : 40
Marks Obtained : 40

Section 1 : Coding

1. Problem Statement

Develop a program for managing employee information that caters to both full-time and part-time employees. The program should be capable of computing the salary for each category of employee and presenting their particulars. To achieve this, create two classes, FullTimeEmployee and PartTimeEmployee, that adhere to the Employee interface.

The program is expected to accept input data, including the name and monthly salary for full-time employees, as well as the name, hourly rate, and hours worked for part-time employees. Subsequently, it should calculate and exhibit the employee details and their respective salaries.

For Full-Time employees, the annual salary should be calculated as 12 times the monthly salary.

For Part-Time employees, the salary calculation should be based on the formula: hourly rate * hours worked.

Input Format

The first line of input should be a string representing the name of a full-time employee.

The second line of input should be an integer representing the monthly salary of the full-time employee.

The third line of input should be a string representing the name of a part-time employee.

The fourth line of input should be an integer representing the hourly rate of the part-time employee.

The fifth line of input should be an integer representing the number of hours worked by the part-time employee.

Output Format

The output displays the following details:

Full-Time Employee Details:

Name: [Full-Time Employee Name] (string)

Monthly Salary: \$[Monthly Salary] (integer)

Annual Salary: \$[12 times Monthly Salary] (integer)

Part-Time Employee Details:

Name: [Part-Time Employee Name] (string)

Hourly Rate: \$[Hourly Rate] (integer)

Hours Worked: [Hours Worked] hours (integer)

Monthly Salary: \$[Calculated Monthly Salary] (integer)

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: John Smith

15000

Mary Johnson

100

100

Output: Full-Time Employee Details:

Name: John Smith

Monthly Salary: \$15000

Annual Salary: \$180000

Part-Time Employee Details:

Name: Mary Johnson

Hourly Rate: \$100

Hours Worked: 100 hours

Monthly Salary: \$10000

Answer

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

```
interface Employee {  
    void displayDetails();  
}
```

```
class FullTimeEmployee implements Employee {  
    private String name;  
    private int monthlySalary;
```

```
    public FullTimeEmployee(String name, int monthlySalary) {  
        this.name = name;  
        this.monthlySalary = monthlySalary;  
    }
```

```

    public void displayDetails() {
        int annualSalary = monthlySalary * 12;
        System.out.println("Full-Time Employee Details:");
        System.out.println("Name: " + name);
        System.out.println("Monthly Salary: $" + monthlySalary);
        System.out.println("Annual Salary: $" + annualSalary);
        System.out.println();
    }
}

```

```

class PartTimeEmployee implements Employee {

```

```

    private String name;
    private int hourlyRate;
    private int hoursWorked;

```

```

    public PartTimeEmployee(String name, int hourlyRate, int hoursWorked) {
        this.name = name;
        this.hourlyRate = hourlyRate;
        this.hoursWorked = hoursWorked;
    }

```

```

    public void displayDetails() {
        int monthlySalary = hourlyRate * hoursWorked;
        System.out.println("Part-Time Employee Details:");
        System.out.println("Name: " + name);
        System.out.println("Hourly Rate: $" + hourlyRate);
        System.out.println("Hours Worked: " + hoursWorked + " hours");
        System.out.println("Monthly Salary: $" + monthlySalary);
    }
}

```

```

class EmployeeInheritanceDemo {

```

```

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String fullName = scanner.nextLine();
        int fullTimeSalary = scanner.nextInt();
        scanner.nextLine();
        String partTimeName = scanner.nextLine();
        int hourlyRate = scanner.nextInt();
        int hoursWorked = scanner.nextInt();
        FullTimeEmployee fullTimeEmployee = new FullTimeEmployee(fullName,
fullTimeSalary);

```

```
PartTimeEmployee partTimeEmployee = new
PartTimeEmployee(partTimeName, hourlyRate, hoursWorked);
fullTimeEmployee.displayDetails();
System.out.println();
partTimeEmployee.displayDetails();
scanner.close();
}
}
```

Status : Correct

Marks : 10/10

2. Problem Statement:

Alice has been tasked with implementing a simple calculator interface and a corresponding class for performing basic addition and subtraction operations. The task is to create an interface called Calculator with two methods: add and subtract. The add method should take two numbers as input and return their sum, while the subtract method should take two numbers as input and return their difference.

Implement a class called SimpleCalculator that implements the Calculator interface. This class should provide the functionality for adding and subtracting numbers. Write a code that satisfies the above requirements.

Input Format

The first line of input consists of a single integer, representing the choice

If the choice is 1 or 2, the next two lines consist of 2 double values, representing the numbers to do addition or subtraction.

Output Format

The output prints a float-value with one decimal value representing the sum of two number or difference of two number.

Refer to the sample output for format specification.

Sample Test Case

Input: 1

5.5

3.5

Output: Result: 9.0

Answer

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

```
interface Calculator {  
    double add(double a, double b);  
    double subtract(double a, double b);  
}
```

```
class SimpleCalculator implements Calculator {  
    public double add(double a, double b) {  
        return a + b;  
    }
```

```
    public double subtract(double a, double b) {  
        return a - b;  
    }  
}
```

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

```
        SimpleCalculator calculator = new SimpleCalculator();
```

```
        int choice = scanner.nextInt();
```

```
        if (choice == 1) {  
            double num1 = scanner.nextDouble();  
            double num2 = scanner.nextDouble();  
            double result = calculator.add(num1, num2);  
            System.out.println("Result: " + result);
```

```
        } else if (choice == 2) {  
            double num1 = scanner.nextDouble();  
            double num2 = scanner.nextDouble();  
            double result = calculator.subtract(num1, num2);  
            System.out.println("Result: " + result);
```

```
        } else {  
            System.out.println("Invalid choice. Please choose 1 for addition or 2 for  
            subtraction.");
```

```
}  
    scanner.close();  
}  
}
```

Status : Correct

Marks : 10/10

3. Problem Statement

Sophia is developing a matrix analysis tool for a data analytics company. The tool needs to analyze square matrices and extract insights from the matrix diagonals.

To organize the code properly, Sophia creates an interface named `Matrix` that declares a method for finding the smallest and largest elements along the principal and secondary diagonals of the matrix.

Sophia then creates a class named `MatrixAnalyzer` that implements the `Matrix` interface. This class provides the logic to process a given square matrix and print:

The smallest and largest elements in the principal diagonal (from top-left to bottom-right). The smallest and largest elements in the secondary diagonal (from top-right to bottom-left).

Your task is to implement the `Matrix` interface and the `MatrixAnalyzer` class. The main driver program (in the class `Main`) will read the input matrix, create an instance of `MatrixAnalyzer`, and invoke its method to display the results.

Input Format

The first line contains an integer `n`, representing the size of the square matrix.

The next `n` lines each contain `n` integers separated by spaces, representing the elements of the matrix.

Output Format

The output prints the four lines:

"Smallest Element - 1: <smallest element in the principal diagonal>" (integer)

"Largest Element - 1: <largest element in the principal diagonal>" (integer)

"Smallest Element - 2: <smallest element in the secondary diagonal>" (integer)

"Largest Element - 2: <largest element in the secondary diagonal>" (integer)

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5

7 8 9 0 1

2 3 4 5 6

5 4 2 0 8

23 5 6 8 9

12 5 6 7 32

Output: Smallest Element - 1: 2

Largest Element - 1: 32

Smallest Element - 2: 1

Largest Element - 2: 12

Answer

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

```
interface Matrix {  
    void diagonalsMinMax(int[][] matrix);  
}
```

```
class MatrixAnalyzer implements Matrix {  
    public void diagonalsMinMax(int[][] matrix) {  
        int n = matrix.length;
```

```
        int smallestPrincipal = matrix[0][0];
```

```
        int largestPrincipal = matrix[0][0];
```

```
        int smallestSecondary = matrix[0][n - 1];
```

```
        int largestSecondary = matrix[0][n - 1];
```



```

        for (int i = 0; i < n; i++) {
            // Principal diagonal
            if (matrix[i][i] < smallestPrincipal)
                smallestPrincipal = matrix[i][i];
            if (matrix[i][i] > largestPrincipal)
                largestPrincipal = matrix[i][i];

            // Secondary diagonal
            if (matrix[i][n - 1 - i] < smallestSecondary)
                smallestSecondary = matrix[i][n - 1 - i];
            if (matrix[i][n - 1 - i] > largestSecondary)
                largestSecondary = matrix[i][n - 1 - i];
        }

        System.out.println("Smallest Element - 1: " + smallestPrincipal);
        System.out.println("Largest Element - 1: " + largestPrincipal);
        System.out.println("Smallest Element - 2: " + smallestSecondary);
        System.out.println("Largest Element - 2: " + largestSecondary);
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int[][] matrix = new int[n][n];
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                matrix[i][j] = sc.nextInt();
            }
        }
        MatrixAnalyzer analyzer = new MatrixAnalyzer();
        analyzer.diagonalsMinMax(matrix);
    }
}

```

Status : Correct

Marks : 10/10

4. Problem Statement

Oviya is fascinated by automorphic numbers and wants to create a

program to determine whether a given number is an automorphic number or not.

An automorphic number is a number whose square ends with the same digits as the number itself. For example, $25 = (25)^2 = 625$

Oviya has defined two interfaces: NumberInput for taking user input and AutomorphicChecker for checking if a given number is automorphic. The class AutomorphicNumber implements both interfaces.

Help her complete the task.

Input Format

The input consists of a single integer n.

Output Format

If the input number is an automorphic number, print "n is an automorphic number". Otherwise, print "n is not an automorphic number".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 25

Output: 25 is an automorphic number

Answer

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

```
// You are using Java
interface NumberInput {
    int getInput();
}
```

```
interface AutomorphicChecker {
    boolean checkAutomorphic(int n);
}
```

```
class AutomorphicNumber implements NumberInput, AutomorphicChecker {
```

```

public int getInput() {
    java.util.Scanner sc = new java.util.Scanner(System.in);
    return sc.nextInt();
}

public boolean checkAutomorphic(int n) {
    int square = n * n;
    String numStr = String.valueOf(n);
    String squareStr = String.valueOf(square);
    return squareStr.endsWith(numStr);
}

public class Main {
    public static void main(String[] args) {
        AutomorphicNumber automorphicNumber = new AutomorphicNumber();
        int inputNumber = automorphicNumber.getInput();

        boolean isAutomorphic =
        automorphicNumber.checkAutomorphic(inputNumber);

        if (isAutomorphic) {
            System.out.println(inputNumber+" is an automorphic number");
        } else {
            System.out.println(inputNumber+" is not an automorphic number");
        }
    }
}

```

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: Rahul G

Email: 241901087@rajalakshmi.edu.in

Roll no: 241901087

Phone: 8248227615

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 7_CY

Attempt : 1

Total Mark : 40

Marks Obtained : 40

Section 1 : Coding

1. Problem Statement

Maria, an online store owner, is looking to implement a pricing system that calculates the final price of products after applying discounts. She needs a program that takes the original price of a product and the discount percentage as input and computes the final discounted price. The discount is applied as a percentage of the original price. Maria wants to ensure that the final price is formatted to display exactly two decimal places.

Implement this functionality using the PriceCalculator interface and the DiscountCalculator class.

Input Format

The first line of input consists of the original price (a double value).

The second line of input consists of a discount percentage (a double value).

Output Format

The output displays the final price after the discount, adhering to the following format: "Final Price after discount: \$[final_price]".

Here, [final_price] should be replaced with the calculated final price, formatted as a currency value with two decimal places.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 100.0
10.0

Output: Final Price after discount: \$90.00

Answer

```
import java.util.Scanner;

// You are using Java
interface PriceCalculator {
    double calculatePrice(double originalPrice, double discount);
}

class DiscountCalculator implements PriceCalculator {
    public double calculatePrice(double originalPrice, double discount) {
        double finalPrice = originalPrice - (originalPrice * (discount / 100));
        return finalPrice;
    }
}

class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        double originalPrice = scanner.nextDouble();
        double discount = scanner.nextDouble();
        PriceCalculator calculator = new DiscountCalculator();
        double finalPrice = calculator.calculatePrice(originalPrice, discount);
        System.out.printf("Final Price after discount: $%.2f\n", finalPrice); //
        Formats output to 2 decimal places
    }
}
```

Status : Correct

Marks : 10/10

2. Problem Statement:

Rathish is planning a road trip and needs a program to convert speeds between miles per hour (MPH) and kilometers per hour (KPH).

Create an interface, SpeedConverter, with a method convertSpeed(double mph). Implement the interface with MPHtoKPHConverter class, allowing Rathish to input MPH and receive the converted speed in KPH, rounded to two decimal points.

Formula: Speed in KPH = 1.60934 * Speed in MPH.

Input Format

The input consists of a single double-point number representing the speed in miles per hour (MPH).

Output Format

The output displays the converted speed (double-point number) in kilometers per hour (KPH) rounded off to two decimal points in the following format:

"Speed in KPH: <<converted speed>>".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 1.0

Output: Speed in KPH: 1.61

Answer

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

```
// You are using Java
```

```

interface SpeedConverter {
    double convertSpeed(double mph);
}

class MPHtoKPHConverter implements SpeedConverter {
    public double convertSpeed(double mph) {
        return 1.60934 * mph;
    }
}

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

        double speedInMPH = scanner.nextDouble();

        SpeedConverter converter = new MPHtoKPHConverter();

        double speedInKPH = converter.convertSpeed(speedInMPH);

        System.out.printf("Speed in KPH: %.2f\n", speedInKPH);

        scanner.close();
    }
}

```

Status : Correct

Marks : 10/10

3. Problem Statement

Alex and Bob are designing a control system for household appliances, and one of the appliances is a washing machine. You want to create a program to help them that models the washing machine as a motor and calculates its electricity consumption based on its capacity.

Define an interface named Motor with the following methods:

```
void run() double consume(double capacity)
```

Create a class called WashingMachine that implements the Motor interface.

In the WashingMachine class:

Implement the run() method to print "Washing machine is running." Implement a consume() method to print "Washing machine is consuming electricity." Implement the consume(double capacity) method to calculate the electricity consumption (in kWh) of the washing machine based on its capacity. The formula for electricity consumption is (capacity * 0.05).

Input Format

The input consists of a double value representing the capacity of the washing machine in kW.

Output Format

The first line of output prints "Washing machine is running."

The second line prints "Washing machine is consuming electricity."

The third line prints "Electricity consumption: X kWh" where X is a double value, rounded off to two decimal places, representing the electricity consumption.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 2.5

Output: Washing machine is running.

Washing machine is consuming electricity.

Electricity consumption: 0.13 kWh

Answer

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

```
interface Motor {  
    void run();  
    void consume();           // no-arg: prints the "consuming electricity" message  
    double consume(double capacity); // returns the numeric consumption  
}
```



```

class WashingMachine implements Motor {
    public void run() {
        System.out.println("Washing machine is running.");
    }

    // no-arg method prints the message once (used by footer)
    public void consume() {
        System.out.println("Washing machine is consuming electricity.");
    }

    // this one only computes and returns the consumption (no printing)
    public double consume(double capacity) {
        return capacity * 0.05;
    }
}

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

        WashingMachine washingMachine = new WashingMachine();

        double capacity = scanner.nextDouble();

        washingMachine.run();
        washingMachine.consume();

        double consumption = washingMachine.consume(capacity);
        System.out.printf("Electricity consumption: %.2f kWh", consumption);

        scanner.close();
    }
}

```

Status : Correct

Marks : 10/10

4. Problem Statement

Jeevan is developing a fitness-tracking application to monitor daily physical activity.

The application incorporates a `FitnessTracker` class that implements two interfaces: `StepCounter` for tracking the number of steps taken and `CalorieCalculator` for estimating total calories burned based on total steps.

Jeevan needs your help creating a program.

Note

The calorie calculation formula is: $\text{Total caloriesBurned} = (\text{total steps} / 100.0) * 20.0$.

Input Format

The first line of input is an integer `n`, representing the number of days Jeevan wants to input data.

The second line consists of space-separated integers, representing the number of steps Jeevan took on each day.

Output Format

The first line of output prints: "Total Steps: <totalSteps>", where '<totalSteps>' is the sum of steps (integer) taken over 'n' days.

The second line prints: "Calories Burned: <caloriesBurned>", where '<caloriesBurned>' is the estimated total calories (double-point number) burned based on the total steps taken rounded off to two decimal places.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3

340 234 987

Output: Total Steps: 1561

Calories Burned: 312.20

Answer

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

```
interface StepCounter {
    void countSteps(int steps); // takes single day's step count
    int getTotalSteps(); // returns total steps
}

interface CalorieCalculator {
    double calculateCaloriesBurned(int totalSteps);
}

class FitnessTracker implements StepCounter, CalorieCalculator {
    private int totalSteps = 0;

    // adds one day's steps
    public void countSteps(int steps) {
        totalSteps += steps;
    }

    public int getTotalSteps() {
        return totalSteps;
    }

    // formula: (totalSteps / 100.0) * 20.0
    public double calculateCaloriesBurned(int totalSteps) {
        return (totalSteps / 100.0) * 20.0;
    }
}

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

        FitnessTracker tracker = new FitnessTracker();

        int n = scanner.nextInt();

        for (int i = 0; i < n; i++) {
            int steps = scanner.nextInt();
            tracker.countSteps(steps);
        }
    }
}
```

```
int totalSteps = tracker.getTotalSteps();
System.out.println("Total Steps: " + totalSteps);

double caloriesBurned = tracker.calculateCaloriesBurned(totalSteps);
System.out.printf("Calories Burned: %.2f%n", caloriesBurned);

scanner.close();
}
}
```

Status : Correct

Marks : 10/10