Rajalakshmi Engineering College

Name: Shivam Jaiswal
Email: 240701499@rajalakshmi.edu.in
Roll no: 240701499
Phone: 7318545479
Branch: REC
Department: CSE - Section 6
Batch: 2028
Degree: B.E - CSE

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;

// You are using Java
interface Employee {
    void displayDetails();
    int calculateSalary();
}

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

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

```
        this.monthlySalary = monthlySalary;
    }

    public int calculateSalary() {
        return monthlySalary * 12;
    }

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

    public void displayDetails() {
        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: $" + calculateSalary());
    }
}

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

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;

interface NumberInput {
    int getInput();
}

interface AutomorphicChecker {
    boolean checkAutomorphic(int n);
}

class AutomorphicNumber implements NumberInput, AutomorphicChecker {
    Scanner scanner = new Scanner(System.in);

    public int getInput() {
        return scanner.nextInt();
    }

    public boolean checkAutomorphic(int n) {
        int square = n * n;
        int temp = n;
        while (temp > 0) {
            if (temp % 10 != square % 10) {
                return false;
            }
            temp /= 10;
            square /= 10;
        }
        return true;
    }
}
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**

4. 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 principalMin = matrix[0][0];
        int principalMax = matrix[0][0];
        int secondaryMin = matrix[0][n - 1];
        int secondaryMax = matrix[0][n - 1];

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

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

        System.out.println("Smallest Element - 1: " + principalMin);
        System.out.println("Largest Element - 1: " + principalMax);
        System.out.println("Smallest Element - 2: " + secondaryMin);
        System.out.println("Largest Element - 2: " + secondaryMax);
    }
}

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**