Status	Finished
Started	Monday, 7 October 2024, 7:20 PM
Completed	Monday, 7 October 2024, 8:20 PM
Duration	1 hour

Question **1**

Correct

Marked out of 5.00

1. Final Variable:

- Once a variable is declared `final`, its value cannot be changed after it is initialized.
- It must be initialized when it is declared or in the constructor if it's not initialized at declaration.
- It can be used to define constants

final int MAX_SPEED = 120; // Constant value, cannot be changed

2. Final Method:

- A method declared `final` cannot be overridden by subclasses.
- It is used to prevent modification of the method's behavior in derived classes.

public final void display() {
System.out.println("This is a final method.");
}

3. Final Class:

- A class declared as `final` cannot be subclassed (i.e., no other class can inherit from it).
- It is used to prevent a class from being extended and modified.
- public final class Vehicle {
  // class code
  }

Given a Java Program that contains the bug in it, your task is to clear the bug to the output.

you should delete any piece of code.

For example:

Test	Result
1	The maximum speed is: 120 km/h This is a subclass of FinalExample.

Answer: (penalty regime: 0 %)

Reset answer

```
public class FinalExample {
    final int MAX_SPEED = 120;  // final variable

    public final void display() {  // final method
        System.out.println("The maximum speed is: " + MAX_SPEED + " km/h");
    }

    public static void main(String[] args) {
        Subclass obj = new Subclass();
        obj.display();       // Calls final method from FinalExample
        obj.showMessage();   // Calls method from Subclass
    }
}

class Subclass extends FinalExample {
    public void showMessage() {
        System.out.println("This is a subclass of FinalExample.");
    }
}
```

	Test	Expected	Got	
✓	1	The maximum speed is: 120 km/h This is a subclass of FinalExample.	The maximum speed is: 120 km/h This is a subclass of FinalExample.	✓

Passed all tests! ✓

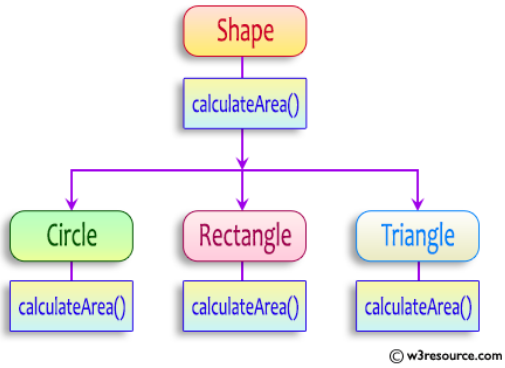
Question **2**

Correct

Marked out of 5.00

Create a base class Shape with a method called calculateArea(). Create three subclasses: Circle, Rectangle, and Triangle. Override the calculateArea() method in each subclass to calculate and return the shape's area.

In the given exercise, here is a simple diagram illustrating polymorphism implementation:




```
 abstract class Shape {
   public abstract double calculateArea() ;
   }
}
```

System.out.printf("Area of a Triangle :%.2f%n",((0.5)*base*height)); // use this statement

sample Input :

4 // radius of the circle to calculate area PI*r*r

5 // length of the rectangle

6 // breadth of the rectangle to calculate the area of a rectangle

4 // base of the triangle

3 // height of the triangle

OUTPUT:

Area of a circle :50.27
Area of a Rectangle :30.00
Area of a Triangle :6.00

For example:

Test	Input	Result
1	4 5 6 4 3	Area of a circle: 50.27 Area of a Rectangle: 30.00 Area of a Triangle: 6.00
2	7 4.5 6.5 2.4 3.6	Area of a circle: 153.94 Area of a Rectangle: 29.25 Area of a Triangle: 4.32

Answer: (penalty regime: 0 %)

```
import java.util.Scanner;

// Abstract Shape class
abstract class Shape {
    public abstract double calculateArea();
}

// Circle class extending Shape
class Circle extends Shape {
    private double radius;

    public Circle(double radius) {
```

```
        this.radius = radius;
    }

    @Override
    public double calculateArea() {
        return Math.PI * radius * radius;  // PI * r^2
    }
}

// Rectangle class extending Shape
class Rectangle extends Shape {
    private double length;
    private double breadth;

    public Rectangle(double length, double breadth) {
        this.length = length;
        this.breadth = breadth;
    }

    @Override
    public double calculateArea() {
        return length * breadth;  // length * breadth
    }
}

// Triangle class extending Shape
class Triangle extends Shape {
    private double base;
    private double height;

    public Triangle(double base, double height) {
        this.base = base;
        this.height = height;
    }

    @Override
    public double calculateArea() {
        return 0.5 * base * height;  // 0.5 * base * height
    }
}
```

	Test	Input	Expected	Got	
✓	1	4 5 6 4 3	Area of a circle: 50.27 Area of a Rectangle: 30.00 Area of a Triangle: 6.00	Area of a circle: 50.27 Area of a Rectangle: 30.00 Area of a Triangle: 6.00	✓
✓	2	7 4.5 6.5 2.4 3.6	Area of a circle: 153.94 Area of a Rectangle: 29.25 Area of a Triangle: 4.32	Area of a circle: 153.94 Area of a Rectangle: 29.25 Area of a Triangle: 4.32	✓

Passed all tests! ✓

Question **3**

Correct

Marked out of 5.00

As a logic building learner you are given the task to extract the string which has vowel as the first and last characters from the given array of Strings.

Step1: Scan through the array of Strings, extract the Strings with first and last characters as vowels; these strings should be concatenated.

Step2: Convert the concatenated string to lowercase and return it.

If none of the strings in the array has first and last character as vowel, then return no matches found

input1: an integer representing the number of elements in the array.

input2: String array.

Example 1:

input1: 3

input2: {"oreo", "sirish", "apple"}

output: oreoapple

Example 2:

input1: 2

input2: {"Mango", "banana"}

output: no matches found

Explanation:

None of the strings has first and last character as vowel.

Hence the output is no matches found.

Example 3:

input1: 3

input2: {"Ate", "Ace", "Girl"}

output: ateace

For example:

Input	Result
3 oreo sirish apple	oreoapple
2 Mango banana	no matches found
3 Ate Ace Girl	ateace

Answer: (penalty regime: 0 %)

```
import java.util.Scanner;

public class VowelStringConcatenation {

    // Method to check if a character is a vowel
    public static boolean isVowel(char c) {
        c = Character.toLowerCase(c);
        return c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u';
    }

    // Method to extract strings with vowels as first and last characters and concatenate them
    public static String extractAndConcatenate(int numOfStrings, String[] arr) {
        StringBuilder result = new StringBuilder();  // To store the concatenated result

        for (String str : arr) {
            // Check if the string is non-empty and if both the first and last characters are vowels
            if (str.length() > 0 && isVowel(str.charAt(0)) && isVowel(str.charAt(str.length() - 1))) {
                result.append(str);  // Concatenate the matching strings
            }
```

```
        }

        // If no matching string was found, return "no matches found"
        if (result.length() == 0) {
            return "no matches found";
        }

        // Convert the concatenated result to lowercase and return
        return result.toString().toLowerCase();
    }

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

        // Input the number of strings in the array
        int numOfStrings = scanner.nextInt();
        scanner.nextLine();  // Consume the newline after the integer input

        // Input the strings in a single line
        String[] arr = new String[numOfStrings];
        String inputLine = scanner.nextLine(); // Read the entire line of strings
        String[] inputStrings = inputLine.split(" "); // Split the input line into an array of strings

        // Store in the array
        for (int i = 0; i < numOfStrings; i++) {
            arr[i] = inputStrings[i];
        }

        // Get the result by calling the method
        String result = extractAndConcatenate(numOfStrings, arr);

        // Print the result
        System.out.println(result);
```

	Input	Expected	Got	
✓	3 oreo sirish apple	oreoapple	oreoapple	✓
✓	2 Mango banana	no matches found	no matches found	✓
✓	3 Ate Ace Girl	ateace	ateace	✓

Passed all tests! ✓

◄ Lab-08-MCQ

Jump to...

FindStringCode ►