CS23333-Object Oriented Programming Using Java-2023

Quiz navigation

| 1 | 2 | 3 |

Show one page at a time

Finish review

Status	Finished
Started	Tuesday, 8 October 2024, 8:45 PM
Completed	Tuesday, 8 October 2024, 8:47 PM
Duration	2 mins 44 secs

Question 1

Correct

Marked out of 5.00

⚑ Flag question

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 %)

```java
import java.util.Scanner;



public class VowelStringExtractor {

    // Method to extract strings with vowels as first and last characters
    public static String extractVowelStrings(String[] stringArray) {
        StringBuilder result = new StringBuilder();
        String vowels = "aeiouAEIOU"; // String containing all vowels

        // Iterate through the array of strings
        for (String s : stringArray) {
            // Check if the string is not empty and if both the first and last characters are vowels
            if (s.length() > 0 && vowels.indexOf(s.charAt(0)) != -1 && vowels.indexOf(s.charAt(s.length() - 1)) != -1
                result.append(s); // Append matching string to the result
            }
        }

        // Return the concatenated string in lowercase or "no matches found"
        return result.length() > 0 ? result.toString().toLowerCase() : "no matches found";
    }

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

        // Input for the number of strings

        int n = scanner.nextInt();
        scanner.nextLine(); // Consume the newline character

        // Input for the strings in one line

        String input = scanner.nextLine();
        String[] strings = input.split(" "); // Split input into an array

        // Process and output the result
        String result = extractVowelStrings(strings);
        System.out.println(result);

        scanner.close(); // Close the scanner
    }
}
```

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!

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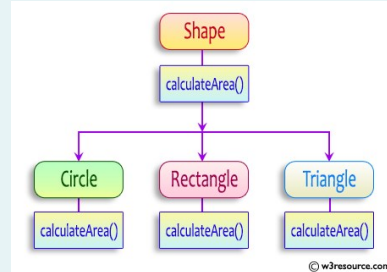
abstract class Shape {
  public abstract double calculateArea() ;
   }
}

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 %)

```java
import java.util.Scanner;



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

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

    public Circle(double radius) {
        this.radius = radius;
    }

    @Override
    public double calculateArea() {
        return Math.PI * radius * radius; // Area of circle: πr²
    }
}

// Rectangle class
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; // Area of rectangle: length * breadth
    }
}

// Triangle class
class Triangle extends Shape {
```

```java
42        private double base;
43        private double height;
44
45        public Triangle(double base, double height) {
46            this.base = base;
47            this.height = height;
48        }
49
50        @Override
51        public double calculateArea() {
52            return 0.5 * base * height; // Area of triangle: 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!

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

```java
1   // Final class definition
2
3   final class FinalExample {
4
5       // Final variable
6       final int MAX_SPEED = 120; // Constant value
7
8       // Final method
9       public final void display() {
10          System.out.println("The maximum speed is: " + MAX_SPEED + " km/h");
11      }
12  }
13
14  // Main class to test the final class
15  public class Test {
16      public static void main(String[] args) {
17          // Create an instance of FinalExample
18          FinalExample example = new FinalExample();
19          example.display();
20
21          // Uncommenting the following line will result in a compile-time error
22          // because FinalExample is a final class and cannot be subclassed.
23          // class SubclassExample extends FinalExample { }
24
25          System.out.println("This is a subclass of FinalExample.");
26      }
27  }
28
```

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!

Jump to...

Jump to...