**WEEK-08**

Question 1

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:



Program:

import java.util.Scanner;

public class VowelStringExtractor {

public static String extractVowelStrings(int n, String[] strings) {

StringBuilder concatenatedResult = new StringBuilder(); // To hold concatenated strings

// Define vowels

String vowels = "aeiouAEIOU";

// Scan through the array of strings

for (String str : strings) {

if (str.length() > 0) { // Check if the string is not empty

char firstChar = str.charAt(0);

char lastChar = str.charAt(str.length() - 1);

// Check if both first and last characters are vowels

if (vowels.indexOf(firstChar) != -1 && vowels.indexOf(lastChar) != -1) {

concatenatedResult.append(str); // Append valid string

}

}

}

// Check if any valid strings were found

if (concatenatedResult.length() == 0) {

return "no matches found"; // Return if no matches found

} else {

return concatenatedResult.toString().toLowerCase(); // Return concatenated result in lowercase

}

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Read number of elements in the array

int n = scanner.nextInt();

scanner.nextLine(); // Consume the newline after nextInt

// Read the single line of strings

String inputLine = scanner.nextLine();

// Split the input line into an array of strings

String[] strings = inputLine.split(" ");

// Ensure we only take n strings if more were provided

if (strings.length > n) {

String[] temp = new String[n];

System.arraycopy(strings, 0, temp, 0, n);

strings = temp; // Trim the array to n elements

}

// Extract the vowel strings and print the result

String result = extractVowelStrings(n, strings);

System.out.println(result);

scanner.close(); // Close the scanner

}

}



Question **2**

**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:**

****

**Program:**

**import java.util.Scanner;**

**// Abstract base class**

**abstract class Shape {**

**public abstract double calculateArea(); // Abstract method for area calculation**

**}**

**// Circle subclass**

**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 a circle: π \* r \* r**

**}**

**}**

**// Rectangle subclass**

**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 a rectangle: length \* breadth**

**}**

**}**

**// Triangle subclass**

**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; // Area of a triangle: 0.5 \* base \* height**

**}**

**}**

**public class ShapeAreaCalculator {**

**public static void main(String[] args) {**

**Scanner scanner = new Scanner(System.in);**

**// Using a try-catch to handle potential input issues**

**try {**

**for (int i = 0; i < 2; i++) { // Loop for two sets of inputs**

**// Read inputs line by line**

**double radius = Double.parseDouble(scanner.nextLine());**

**double length = Double.parseDouble(scanner.nextLine());**

**double breadth = Double.parseDouble(scanner.nextLine());**

**double base = Double.parseDouble(scanner.nextLine());**

**double height = Double.parseDouble(scanner.nextLine());**

**// Creating objects for each shape**

**Circle circle = new Circle(radius);**

**Rectangle rectangle = new Rectangle(length, breadth);**

**Triangle triangle = new Triangle(base, height);**

**// Calculating and displaying areas**

**System.out.printf("Area of a circle: %.2f%n", circle.calculateArea());**

**System.out.printf("Area of a Rectangle: %.2f%n", rectangle.calculateArea());**

**System.out.printf("Area of a Triangle: %.2f%n", triangle.calculateArea());**

**}**

**} catch (NumberFormatException e) {**

**//System.out.println("Invalid number format: " + e.getMessage());**

**} catch (Exception e) {**

**//System.out.println("Error reading input: " + e.getMessage());**

**} finally {**

**scanner.close(); // Close the scanner**

**}**

**}**

**}**

****

**Question 3**

**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:**

****

**Program:**

**class FinalExample {**

**// Final variable**

**final int maxSpeed = 120; // Changed to final to prevent modification**

**// Final method**

**public final void displayMaxSpeed() { // Marked as final to prevent overriding**

**System.out.println("The maximum speed is: " + maxSpeed + " km/h"); // Corrected string concatenation**

**}**

**}**

**class Subclass extends FinalExample {**

**// Cannot override the final method displayMaxSpeed()**

**// public void displayMaxSpeed() {**

**// System.out.println("Cannot override a final method"); // This is commented out since we can't override a final method**

**// }**

**// You can create new methods here**

**public void showDetails() {**

**System.out.println("This is a subclass of FinalExample.");**

**}**

**}**

**public class Prog { // Ensure the class name is capitalized**

**public static void main(String[] args) {**

**FinalExample obj = new FinalExample();**

**obj.displayMaxSpeed(); // Correctly calls the method**

**Subclass subObj = new Subclass();**

**subObj.showDetails(); // Correctly calls the new method**

**}**

**}**

****