1.     Method Overloading: Write a class Calculator with overloaded methods add(). Implement add() methods that take:

     - Two integers

     - Two double values

     - Three integers

     - A variable number of integers

Program:-

**package** cal\_demo;

**public** **class** Calculator {

**public** **int** add(**int** a, **int** b) {

**return** a + b;

}

// Method to add two double values

**public** **double** add(**double** a, **double** b) {

**return** a + b;

}

// Method to add three integers

**public** **int** add(**int** a, **int** b, **int** c) {

**return** a + b + c;

}

// Method to add a variable number of integers

**public** **int** add(**int**... numbers) {

**int** sum = 0;

**for** (**int** number : numbers) {

sum += number;

}

**return** sum;

}

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

// **TODO** Auto-generated method stub

Calculator calc = **new** Calculator();

// Testing the overloaded methods

System.***out***.println("Add two integers: " + calc.add(2, 3)); // Output: 5

System.***out***.println("Add two doubles: " + calc.add(2.5, 3.5)); // Output: 6.0

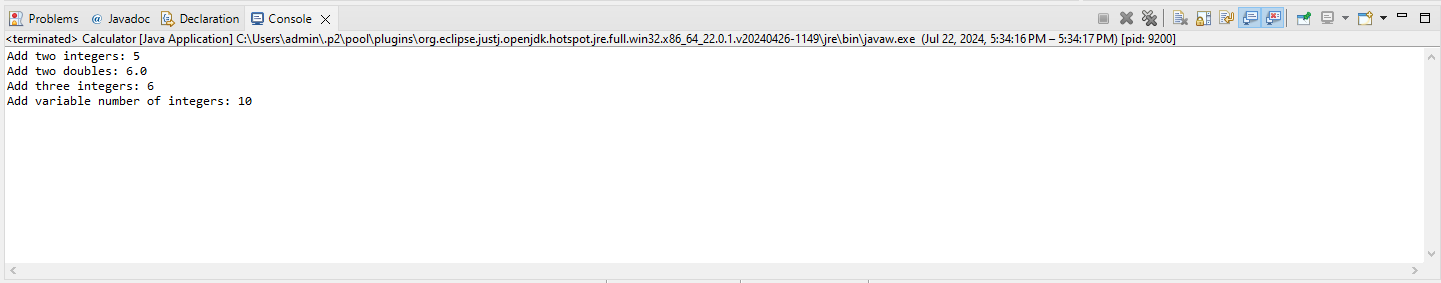
System.***out***.println("Add three integers: " + calc.add(1, 2, 3)); // Output: 6

System.***out***.println("Add variable number of integers: " + calc.add(1, 2, 3, 4)); // Output: 10

}

}

Output:-



2. Super Keyword: Create a class Person with a constructor that accepts and sets name and age.

   - Create a subclass Student that adds a grade property and initializes name and age using the super keyword in its constructor.

   - Demonstrate the creation of Student objects and the usage of super to call the parent class constructor.

Program:-

**package** Demo;

**class** Person {

**protected** String name;

**protected** **int** age;

// Constructor for Person class

**public** Person(String name, **int** age) {

**this**.name = name;

**this**.age = age;

}

// Method to display Person details

**public** **void** display() {

System.***out***.println("Name: " + name);

System.***out***.println("Age: " + age);

}

}

**class** Student **extends** Person {

**private** String grade;

// Constructor for Student class

**public** Student(String name, **int** age, String grade) {

**super**(name, age); // Call the constructor of the Person class

**this**.grade = grade;

}

// Method to display Student details

@Override

**public** **void** display() {

**super**.display(); // Call the display method of the Person class

System.***out***.println("Grade: " + grade);

}

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

// **TODO** Auto-generated method stub

Student student1 = **new** Student("Alice", 20, "A");

Student student2 = **new** Student("Bob", 22, "B");

// Displaying Student details

System.***out***.println("Student 1 details:");

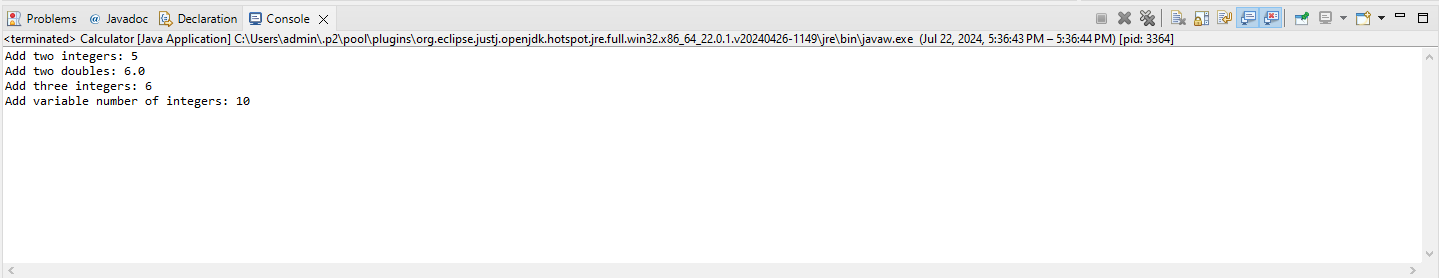
student1.display();

System.***out***.println("\nStudent 2 details:");

student2.display();

}

}

Output:-

3. Super Keyword: Create a base class Shape with a method draw() that prints "Drawing Shape".

   - Create a subclass Circle that overrides draw() to print "Drawing Circle".

   - Inside the draw() method of Circle, call the draw() method of the Shape class using super.draw().

   - Write a main method to demonstrate calling draw() on a Circle object.

Program:-

**package** Demo;

**class** Shape {

**public** **void** draw() {

System.***out***.println("Drawing Shape");

}

}

**class** Circle **extends** Shape {

// Method to draw a circle

@Override

**public** **void** draw() {

**super**.draw(); // Call the draw method of the Shape class

System.***out***.println("Drawing Circle");

}

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

// **TODO** Auto-generated method stub

// Creating a Circle object

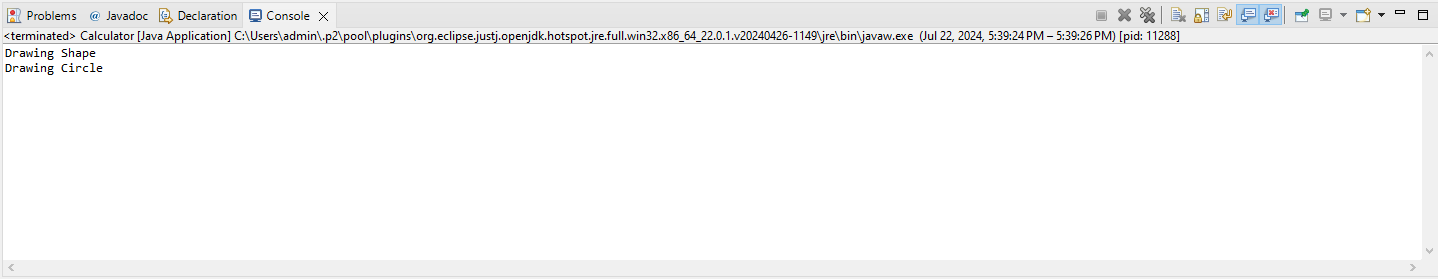
Circle circle = **new** Circle();

// Calling the draw method on the Circle object

circle.draw();

}

}

Output:-

4. Write a Java Program to count the number of words in a String without using the Predefined method?

Program:-

**package** Demo;

**public** **class** Counter {

**public** **static** **int** countWords(String str) {

**int** wordCount = 0;

**boolean** isWord = **false**;

**int** endOfLine = str.length() - 1;

**for** (**int** i = 0; i < str.length(); i++) {

// Check if the current character is a letter

**if** (Character.*isLetter*(str.charAt(i)) && i != endOfLine) {

isWord = **true**;

} **else** **if** (!Character.*isLetter*(str.charAt(i)) && isWord) {

wordCount++;

isWord = **false**;

} **else** **if** (Character.*isLetter*(str.charAt(i)) && i == endOfLine) {

wordCount++;

}

}

**return** wordCount;

}

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

// **TODO** Auto-generated method stub

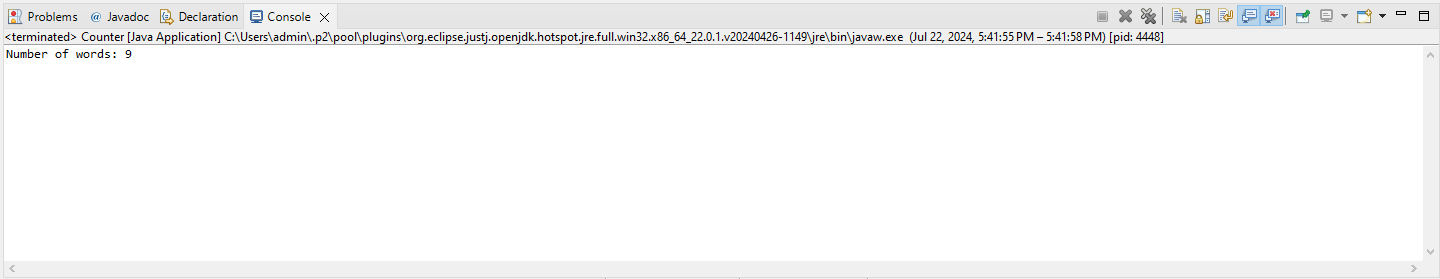
String testString = "Hello, this is a test string to count words!";

**int** numberOfWords = *countWords*(testString);

System.***out***.println("Number of words: " + numberOfWords);

}

}

Output:-

5. Write a Java Program to remove all white spaces from a String?

Program:-

**package** Demo;

**public** **class** whitespaces {

**public** **static** String removeSpaces(String str) {

// Initialize a new StringBuilder to build the result string

StringBuilder result = **new** StringBuilder();

// Loop through each character in the input string

**for** (**int** i = 0; i < str.length(); i++) {

// If the current character is not a white space, append it to the result

**if** (!Character.*isWhitespace*(str.charAt(i))) {

result.append(str.charAt(i));

}

}

// Convert the StringBuilder back to a String and return it

**return** result.toString();

}

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

// **TODO** Auto-generated method stub

String testString = "Hello, this is a test string to remove white spaces!";

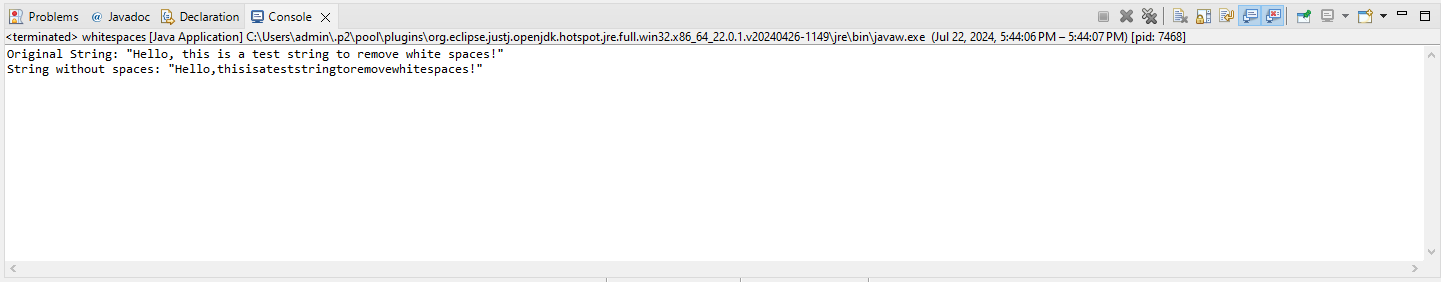
String stringWithoutSpaces = *removeSpaces*(testString);

System.***out***.println("Original String: \"" + testString + "\"");

System.***out***.println("String without spaces: \"" + stringWithoutSpaces + "\"");

}

}

Output:-

6. WAP to find occurrence of given in the given string.

Program:-

**package** Demo;

**public** **class** occurrence {

**public** **static** **int** findOccurrence(String str, **char** ch) {

**int** count = 0;

// Loop through each character in the input string

**for** (**int** i = 0; i < str.length(); i++) {

// If the current character matches the given character, increment the count

**if** (str.charAt(i) == ch) {

count++;

}

}

**return** count;

}

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

// **TODO** Auto-generated method stub

String testString = "Hello, this is a test string to find character occurrence!";

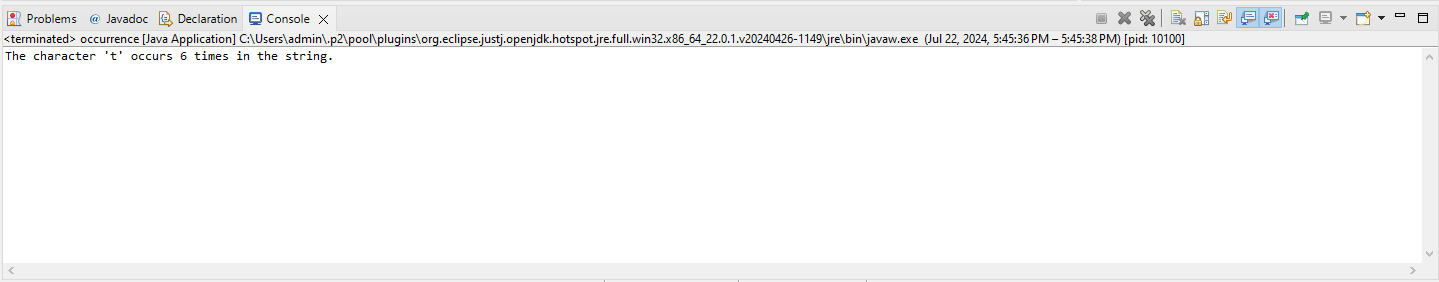
**char** characterToFind = 't';

**int** occurrence = *findOccurrence*(testString, characterToFind);

System.***out***.println("The character '" + characterToFind + "' occurs " + occurrence + " times in the string.");

}

}

Output:-

7. Write a java class to implement any 10 string methods:

● replace ● contains ● replaceAll ● indexOf ● substring ● Equals ● lastIndexOf ● startsWith

● endsWith ● EqualsIgnoreCase ● toLowerCase ● toUpperCase ● isEmpty ● Length ● split

Program:-

**package** Demo;

**public** **class** StringMethodsDemo {

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

// **TODO** Auto-generated method stub

String str = "Hello, this is a test string to demonstrate various string methods!";

String str2 = "HELLO, THIS IS A TEST STRING TO DEMONSTRATE VARIOUS STRING METHODS!";

// replace

String replacedString = str.replace('t', 'T');

System.***out***.println("Replaced 't' with 'T': " + replacedString);

// contains

**boolean** containsTest = str.contains("test");

System.***out***.println("Contains 'test': " + containsTest);

// replaceAll

String replacedAllString = str.replaceAll("string", "STRING");

System.***out***.println("Replaced all 'string' with 'STRING': " + replacedAllString);

// indexOf

**int** indexOfT = str.indexOf('t');

System.***out***.println("Index of 't': " + indexOfT);

// substring

String substring = str.substring(7, 16);

System.***out***.println("Substring (7, 16): " + substring);

// equals

**boolean** equalsStr = str.equals(str2);

System.***out***.println("Equals str2: " + equalsStr);

// lastIndexOf

**int** lastIndexOfT = str.lastIndexOf('t');

System.***out***.println("Last index of 't': " + lastIndexOfT);

// startsWith

**boolean** startsWithHello = str.startsWith("Hello");

System.***out***.println("Starts with 'Hello': " + startsWithHello);

// endsWith

**boolean** endsWithMethods = str.endsWith("methods!");

System.***out***.println("Ends with 'methods!': " + endsWithMethods);

// equalsIgnoreCase

**boolean** equalsIgnoreCaseStr = str.equalsIgnoreCase(str2);

System.***out***.println("Equals ignore case str2: " + equalsIgnoreCaseStr);

// toLowerCase

String lowerCaseString = str.toLowerCase();

System.***out***.println("To lower case: " + lowerCaseString);

// toUpperCase

String upperCaseString = str.toUpperCase();

System.***out***.println("To upper case: " + upperCaseString);

// isEmpty

**boolean** isEmpty = str.isEmpty();

System.***out***.println("Is empty: " + isEmpty);

// length

**int** length = str.length();

System.***out***.println("Length: " + length);

// split

String[] splitString = str.split(" ");

System.***out***.println("Split by space: ");

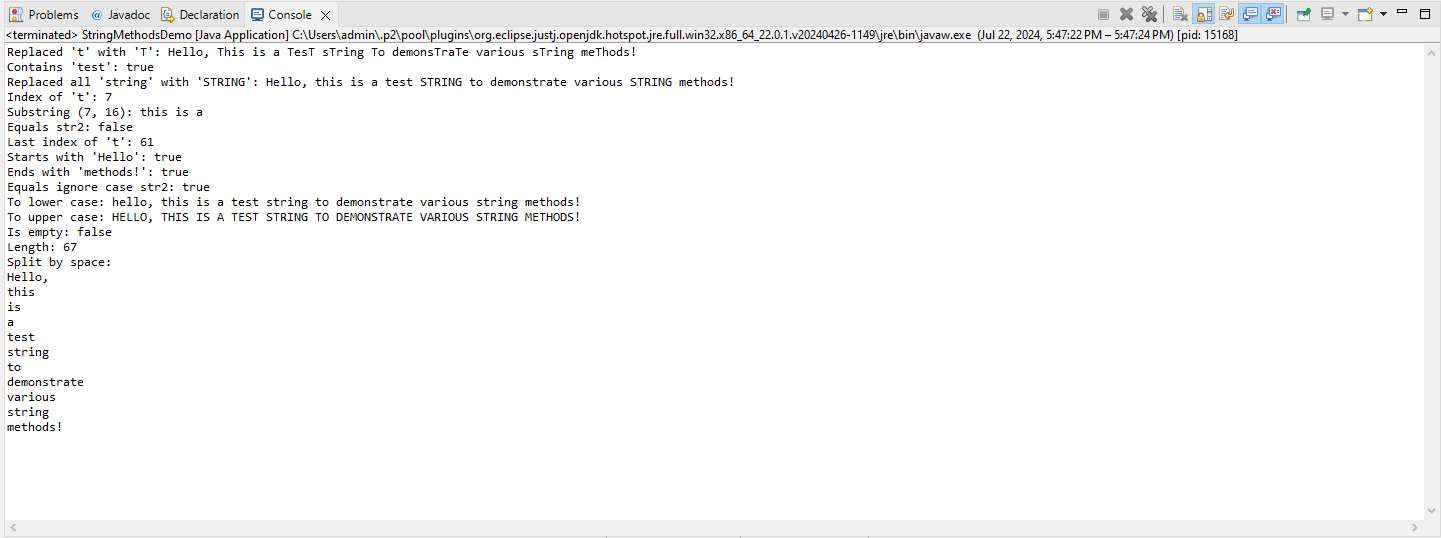
**for** (String s : splitString) {

System.***out***.println(s);

}

}

}

Output:-

8. Write a java program to implement string tokenizer.

Program:-

**package** Demo;

**import** java.util.StringTokenizer;

**public** **class** StringTokenizerDemo {

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

// **TODO** Auto-generated method stub

String str = "Hello, this is a test string to demonstrate StringTokenizer.";

// Create a StringTokenizer with the string and a delimiter

StringTokenizer tokenizer = **new** StringTokenizer(str, " ,.");

// Display the number of tokens

System.***out***.println("Number of tokens: " + tokenizer.countTokens());

// Iterate through the tokens and print each one

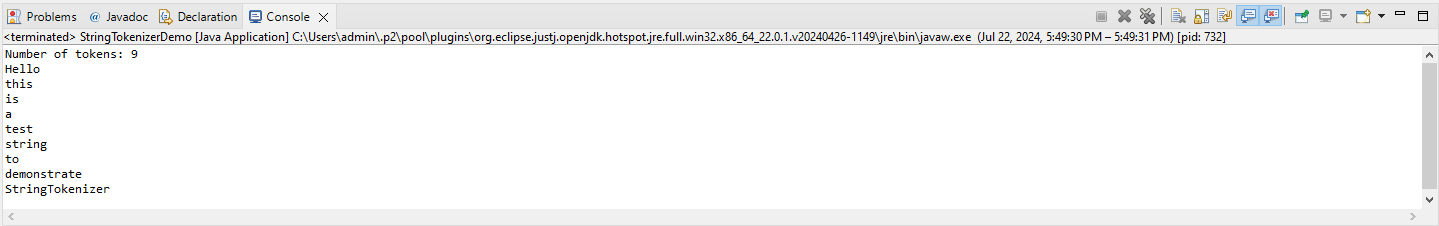
**while** (tokenizer.hasMoreTokens()) {

System.***out***.println(tokenizer.nextToken());

}

}

}

Output:-