* 1. Write a Java program that reads a string from the user and

uses StringTokenizer to split the string into individual words. Print each word on a new line.

## Program:

**package** LAB5;

**import** java.util.Scanner;

**import** java.util.StringTokenizer;

**public class** WordSplitter {

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

// Prompt the user to enter a string System.***out***.println("Enter a string:"); String input = scanner.nextLine();

// Create a StringTokenizer to split the string into words StringTokenizer tokenizer = **new** StringTokenizer(input);

// Iterate through each word and print it

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

{

String word = tokenizer.nextToken(); System.***out***.println(word);

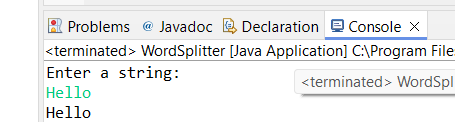
}

// Close the scanner scanner.close();

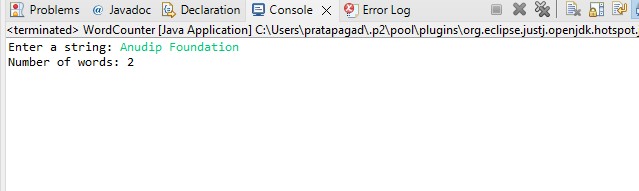
}

}

## Output:



* 1. Write a Java program that reads a string from the user and uses StringTokenizer to count the number of words in the string. **Program:**



**package** LAB5;

**import** java.util.Scanner;

**import** java.util.StringTokenizer;

**public class** WordCounter {

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

// Prompt the user to enter a string System.***out***.println("Enter a string:"); String input = scanner.nextLine();

// Create a StringTokenizer to split the string into words StringTokenizer tokenizer = **new** StringTokenizer(input);

// Count the number of words

**int** wordCount = tokenizer.countTokens();

// Print the word count System.***out***.println("Number of words: " +

wordCount);//printing the result

// Close the scanner scanner.close();

}

}

## Output:

* 1. Write a Java program to create a LinkedList of strings, add elements at specific positions (beginning, middle, end), and print the list. **Program:**

package LAB5;

import java.util.LinkedList;

public class LinkedListExample {

public static void main(String[] args) {

// Create a LinkedList of strings LinkedList<String> linkedList = new LinkedList<>();

// Add elements at the beginning linkedList.addFirst("First");

// Add elements at the end linkedList.addLast("Last");

// Add elements at specific positions (middle) linkedList.add(linkedList.size() / 2, "Middle");

// Print the LinkedList System.*out*.println("LinkedList elements:"); for (String element : linkedList) {

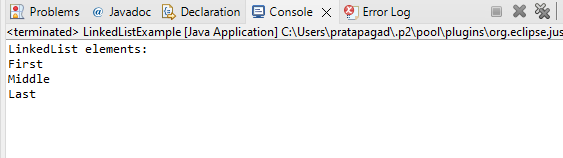
System.*out*.println(element);//printing the elements

}

}

}

## Output:



* 1. Write a Java program to sort a given array list.

## Program:

**package** LAB5;

**import** java.util.ArrayList;

**import** java.util.Collections;

**public class** ArrayListSortExample {

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

// Create an ArrayList of integers ArrayList<Integer> numbers = **new** ArrayList<>();

// Add elements to the ArrayList numbers.add(5);

numbers.add(3); numbers.add(8); numbers.add(1); numbers.add(4);

// Print the unsorted ArrayList System.***out***.println("Before sorting:"); System.***out***.println(numbers);

// Sort the ArrayList Collections.*sort*(numbers);//sorting function of the

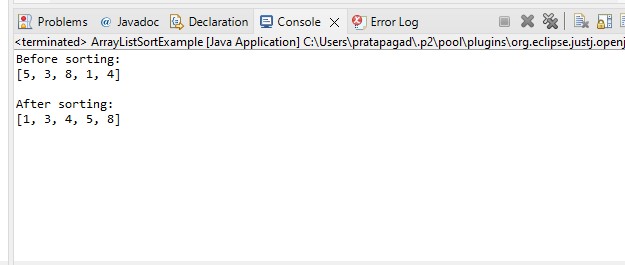
ArrayList

// Print the sorted ArrayList System.***out***.println("\nAfter sorting:"); System.***out***.println(numbers);//printing the result

}

}

## Output:



* 1. Write a Java program to replace the second element of an ArrayList with the specified element.

## Program:

package LAB5;

import java.util.ArrayList;

public class ReplaceSecondElement {

public static void main(String[] args) {

// Create an ArrayList of strings ArrayList<String> list = new ArrayList<>();

// Add elements to the ArrayList list.add("First");

list.add("Second");

list.add("Third");

// Print the ArrayList before replacement System.*out*.println("Before replacement:"); System.*out*.println(list);

// Specify the element to replace the second element String newElement = "NewSecond";

// Check if the ArrayList has at least two elements if (list.size() >= 2) {

// Replace the second element with the specified element list.set(1, newElement);

// Print the ArrayList after replacement System.*out*.println("\nAfter replacement:"); System.*out*.println(list);

} else {

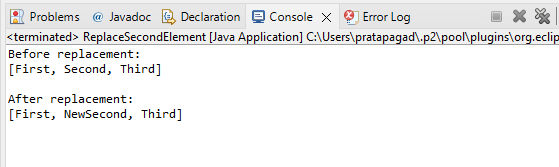
System.*out*.println("ArrayList doesn't have enough elements to replace the second one.");

}

}

}

## Output:



* 1. Write a Java program to iterate a linked list in reverse order.

## Program:

**package** LAB5;

**import** java.util.LinkedList;

**import** java.util.Iterator;

**public class** ReverseLinkedListIterator {

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

// Create a LinkedList of integers LinkedList<Integer> list = **new** LinkedList<>();

// Add elements to the LinkedList list.add(1);

list.add(2);

list.add(3);

list.add(4);

list.add(5);

// Get a reverse iterator for the LinkedList Iterator<Integer> iterator = list.descendingIterator();

// Iterate and print the elements in reverse order System.***out***.println("LinkedList elements in reverse order:"); **while** (iterator.hasNext()) {

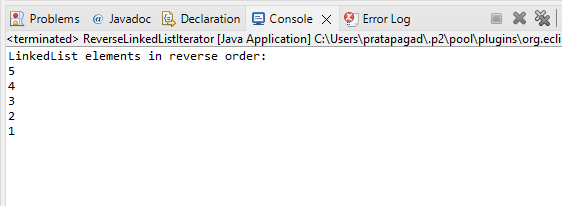
System.***out***.println(iterator.next());

}

}

}

## Output:



* 1. Write a Java program to retrieve, but not remove, the last element of a linked list.

## Program:

package LAB5;

import java.util.LinkedList;

public class RetrieveLastElement {

public static void main(String[] args) {

// Create a LinkedList of integers LinkedList<Integer> list = new LinkedList<>();

// Add elements to the LinkedList list.add(1);

list.add(2);

list.add(3);

list.add(4);

list.add(5);

// Retrieve the last element of the LinkedList (without removing it)

Integer lastElement = list.getLast();

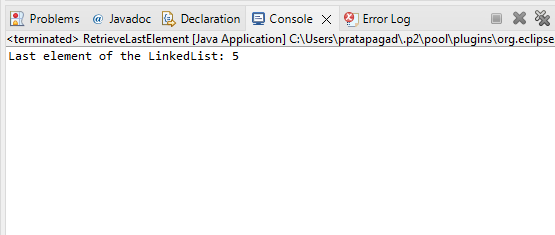
// Print the last element

System.*out*.println("Last element of the LinkedList: " + lastElement);

}

}

## Output:



* 1. Write a Java program to create a LinkedList of integers and print all the elements.

# Program:

**package** LAB5;

**import** java.util.LinkedList;

**public class** LinkedListExample2 {

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

// Create a LinkedList of integers LinkedList<Integer> linkedList = **new** LinkedList<>();

// Add elements to the LinkedList linkedList.add(1); linkedList.add(2); linkedList.add(3); linkedList.add(4); linkedList.add(5);

// Print all the elements in the LinkedList System.***out***.println("LinkedList elements:"); **for** (Integer element : linkedList) {

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

}

}

}

# Output:

