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 Lab\_5;

import java.util.Scanner;

import java.util.StringTokenizer;

public class StringTokenizerExample {

public static void main(String[] args) {

// Create a Scanner object to read input from the user

Scanner scanner = new Scanner(System.in);

// Prompt the user to enter a string

System.out.print("Enter a string:");

// Read the entire line of input from the user

String inputString = scanner.nextLine();

// Create a StringTokenizer object with the input string

// The default delimiters are space, tab, newline, and carriage return

StringTokenizer tokenizer = new StringTokenizer(inputString);

// Iterate through the tokens (words) and print each one on a new line

while (tokenizer.hasMoreTokens()) {

String word = tokenizer.nextToken();

System.out.println(word);

}

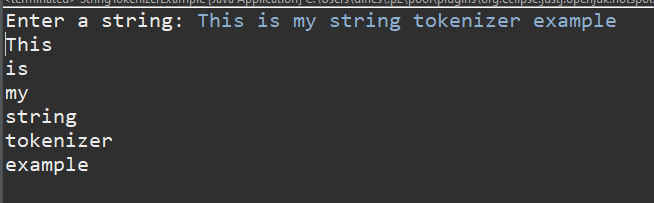
// Close the scanner

scanner.close();

}

}

Output :



2. 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 Lab\_5;

import java.util.Scanner;

import java.util.StringTokenizer;

public class WordCountExample {

public static void main(String[] args) {

// Create a Scanner object to read input from the user

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

// Prompt the user to enter a string

System.***out***.print("Enter a string:");

// Read the entire line of input from the user

String inputString = scanner.nextLine();

// Create a StringTokenizer object with the input string

// The default delimiters are space, tab, newline, and carriage return

StringTokenizer tokenizer = new StringTokenizer(inputString);

// Initialize a count variable to keep track of the number of words

int wordCount = 0;

// Iterate through the tokens (words) and count them

while (tokenizer.hasMoreTokens()) {

tokenizer.nextToken(); // Retrieve the next token (word)

wordCount++; // Increment the word count

}

// Print the total number of words

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

// Close the scanner

scanner.close();

}

}

Output :

A black screen with white text

Description automatically generated

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

Program :

package Lab\_5;

import java.util.LinkedList;

public class LinkedListExample {

public static void main(String[] args) {

// Create a LinkedList of strings

LinkedList<String> list = new LinkedList<>();

// Add elements to the beginning of the list

list.addFirst("First Element");

// Add elements to the end of the list

list.addLast("Last Element");

// Add elements to the middle of the list

// Since the list currently has two elements, add an element at index 1 (middle position)

list.add(1, "Middle Element");

// Print the entire list

System.***out***.println("LinkedList contents:");

for (String element : list) {

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

}

}

}

Output :

A black screen with white text

Description automatically generated

4.     Write a Java program to sort a given array list.

Program :

package Lab\_5;

import java.util.ArrayList;

import java.util.Collections;

public class ArrayListSortNumbers {

public static void main(String[] args) {

// Create an ArrayList of integers

ArrayList<Integer> list = new ArrayList<>();

// Add elements to the ArrayList

list.add(34);

list.add(7);

list.add(23);

list.add(89);

list.add(12);

// Print the ArrayList before sorting

System.out.println("ArrayList before sorting:");

for (Integer number : list) {

System.out.println(number);

}

// Sort the ArrayList using Collections.sort()

Collections.sort(list);

// Print the ArrayList after sorting

System.out.println("\nArrayList after sorting:");

for (Integer number : list) {

System.out.println(number);

}

}

}

Output :

A screenshot of a computer program

Description automatically generated

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

Program :

package Lab\_5;

import java.util.ArrayList;

public class ReplaceElementExample {

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("ArrayList before replacement:");

for (String element : list) {

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

}

// Specify the new element to replace the second element

String newElement = "New Second";

// Replace the second element (index 1) with the new element

list.set(1, newElement);

// Print the ArrayList after replacement

System.***out***.println("\nArrayList after replacement:");

for (String element : list) {

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

}

}

}

Output :

A computer screen shot of a black screen

Description automatically generated

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

Program :

package Lab\_5;

import java.util.LinkedList;

import java.util.ListIterator;

public class LinkedListReverseIterationNumbers {

public static void main(String[] args) {

// Create a LinkedList of integers

LinkedList<Integer> list = new LinkedList<>();

// Add elements to the LinkedList

list.add(10);

list.add(20);

list.add(30);

list.add(40);

list.add(50);

// Print the LinkedList before reverse iteration

System.out.println("LinkedList before reverse iteration:");

for (Integer number : list) {

System.out.println(number);

}

// Get a ListIterator for the LinkedList

ListIterator<Integer> iterator = list.listIterator(list.size());

// Iterate in reverse order using the ListIterator

System.out.println("\nLinkedList in reverse order:");

while (iterator.hasPrevious()) {

System.out.println(iterator.previous());

}

}

}

Output :

A screen shot of a computer

Description automatically generated

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

Program :

package Lab\_5;

import java.util.LinkedList;

public class RetrieveLastElement {

public static void main(String[] args) {

// Create a LinkedList of strings

LinkedList<String> list = new LinkedList<>();

// Add elements to the LinkedList

list.add("First");

list.add("Second");

list.add("Third");

list.add("Fourth");

// Retrieve, but do not remove, the last element

if (!list.isEmpty()) {

String lastElement = list.getLast();

System.***out***.println("The last element in the LinkedList is: " + lastElement);

} else {

System.***out***.println("The LinkedList is empty.");

}

}

}

Output:

A screen shot of a computer

Description automatically generated

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

Program :

package Lab\_5;

import java.util.LinkedList;

public class LinkedListIntegerExample {

public static void main(String[] args) {

// Create a LinkedList of integers

LinkedList<Integer> list = new LinkedList<>();

// Add elements to the LinkedList

list.add(10);

list.add(30);

list.add(50);

list.add(60);

list.add(80);

// Print all the elements in the LinkedList

System.***out***.println("Elements in the LinkedList:");

for (Integer number : list) {

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

}

}

}

Output :

A black screen with white text

Description automatically generated