

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.

CODE :

```
package Hellow;

import java.util.Scanner;
import java.util.StringTokenizer;

public class TokenizerExample {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        // 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.println("Enter a string:");

        // Read the entire line of input from the user
        String inputString = scanner.nextLine();

        // Create a StringTokenizer object to tokenize the input string
        // The delimiter is a space character, but you can customize it as needed
        StringTokenizer tokenizer = new StringTokenizer(inputString);

        // Print each token (word) on a new line
        System.out.println("The individual words are:");
        while (tokenizer.hasMoreTokens()) {
            String word = tokenizer.nextToken();
            System.out.println(word);
        }

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

OUTPUT :

```
<terminated> TokenizerExample [
Enter a string:
HEY, How are you
|The individual words are:
HEY,
How
are
you
```

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

CODE :

```
package Hellow;

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.println("Enter a string:");
        String input = scanner.nextLine();

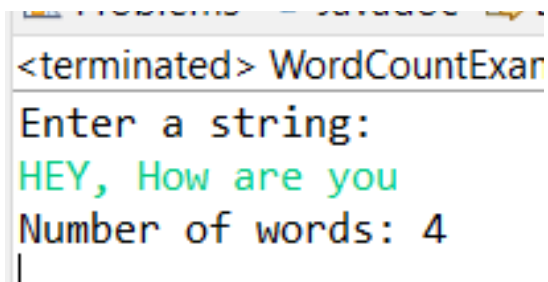
        // Create a StringTokenizer object to split the input string into words
        StringTokenizer tokenizer = new StringTokenizer(input);

        // Count the number of words
        int wordCount = 0;
        while (tokenizer.hasMoreTokens()) {
            tokenizer.nextToken(); // Retrieve and discard each word
            wordCount++; // Increment word count
        }

        // Print the number of words
        System.out.println("Number of words: " + wordCount);

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

OUTPUT :



```
<terminated> WordCountExar
Enter a string:
HEY, How are you
Number of words: 4
```

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

CODE :

```
package Hellow;

import java.util.LinkedList;
public class LinkedListExample {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

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

        // Adding elements to the LinkedList
        linkedList.add("Element 1");
        linkedList.add("Element 2");
        linkedList.add("Element 3");

        // Printing the initial LinkedList
        System.out.println("Initial LinkedList: " + linkedList);

        // Adding an element at the beginning
        linkedList.addFirst("New Element at Beginning");
        System.out.println("After adding at the beginning: " + linkedList);

        // Adding an element at the middle (index 2)
        linkedList.add(2, "New Element in the Middle");
        System.out.println("After adding in the middle: " + linkedList);

        // Adding an element at the end
        linkedList.addLast("New Element at End");
        System.out.println("After adding at the end: " + linkedList);
    }
}
```

OUTPUT :

```
<terminated> LinkedListExample [Java Application] C:\Users\Mr. User\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.0.1.v20240426-1149\jre\bin\javaw.exe
Initial LinkedList: [Element 1, Element 2, Element 3]
After adding at the beginning: [New Element at Beginning, Element 1, Element 2, Element 3]
After adding in the middle: [New Element at Beginning, Element 1, New Element in the Middle, Element 2, Element 3]
After adding at the end: [New Element at Beginning, Element 1, New Element in the Middle, Element 2, Element 3, New Element at End]
```

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

CODE :

```
package Hellow;

import java.util.ArrayList;
import java.util.Collections;
public class SortArrayListExample {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        // Create an ArrayList of Integers
        ArrayList<Integer> numbers = new ArrayList<>();
        numbers.add(5);
        numbers.add(3);
        numbers.add(8);
        numbers.add(1);
        numbers.add(2);

        // Print the original list
        System.out.println("Original list: " + numbers);

        // Sort the ArrayList in ascending order
        Collections.sort(numbers);

        // Print the sorted list
        System.out.println("Sorted list: " + numbers);
    }
}
```

OUTPUT :

```
<terminated> SortArrayListExample [Java
Original list: [5, 3, 8, 1, 2]
Sorted list: [1, 2, 3, 5, 8]
```

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

CODE :

```
package Hellow;

import java.util.ArrayList;
public class ReplaceSecondElement {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

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

        // Add elements to the ArrayList
        list.add("Apple");
        list.add("Banana");
        list.add("Cherry");
        list.add("Mango");

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

        // Specify the new element
        String newElement = "Grape";

        // Replace the second element (index 1) with the new element
        if (list.size() >= 2) {
            list.set(1, newElement);
        } else {
            System.out.println("ArrayList doesn't have a second element to
replace.");
        }

        // Print the ArrayList after replacement
        System.out.println("ArrayList after replacement: " + list);
    }
}
```

OUTPUT :

```
<terminated> ReplaceSecondElement [Java Application] C:\Users\Mr. User\.p2\poc
ArrayList before replacement: [Apple, Banana, Cherry, Mango]
ArrayList after replacement: [Apple, Grape, Cherry, Mango]
```

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

CODE :

```
package Hellow;

import java.util.Iterator;
import java.util.LinkedList;
import java.util.ListIterator;

public class ReverseLinkedListIteration {
    public static void main(String[] args) {
        // Create a LinkedList
        LinkedList<String> linkedList = new LinkedList<>();

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

        // Get the ListIterator to iterate from end to beginning
        Iterator<String> iterator = linkedList.descendingIterator();

        // Iterate over the LinkedList in reverse order
        System.out.println("Iterating LinkedList in reverse order:");
        while (iterator.hasNext()) {
            System.out.println(iterator.next());
        }
    }
}
```

OUTPUT :

```
<terminated> ReverseLinkedListIteration [Java Applic
Iterating LinkedList in reverse order:
Element 4
Element 3
Element 2
Element 1
```

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

CODE :

```
package Hellow;

import java.util.LinkedList;

public class RetrieveLastElements {
    public static void main(String[] args) {
        // Create a LinkedList
        LinkedList<String> linkedList = new LinkedList<>();

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

        // Retrieve the last element (without removing it)
        String lastElement = linkedList.getLast();

        // Print the last element
        System.out.println("Last Element in the LinkedList: " + lastElement);

        // Print the LinkedList to verify it remains unchanged
        System.out.println("LinkedList after retrieval: " + linkedList);
    }
}
```

OUTPUT :

```
<terminated> RetrieveLastElements [Java Application] C:\Users\Mr. User\.p2\pool\plugins\org.eclipse
Last Element in the LinkedList: Element 4
LinkedList after retrieval: [Element 1, Element 2, Element 3, Element 4]
```

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

CODE :

```
package Hellow;

import java.util.LinkedList;

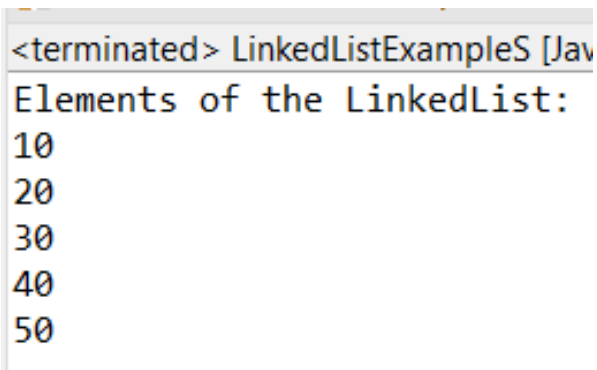
public class LinkedListExampleS {
    public static void main(String[] args) {
        // Create a LinkedList of Integers
        LinkedList<Integer> linkedList = new LinkedList<>();

        // Add elements to the LinkedList
        linkedList.add(10);
        linkedList.add(20);
        linkedList.add(30);
        linkedList.add(40);
        linkedList.add(50);

        // Print all elements of the LinkedList
        System.out.println("Elements of the LinkedList:");

        // Using enhanced for-loop to iterate and print each element
        for (Integer num : linkedList) {
            System.out.println(num);
        }
    }
}
```

OUTPUT :



```
<terminated> LinkedListExampleS [Jav
Elements of the LinkedList:
10
20
30
40
50
```