

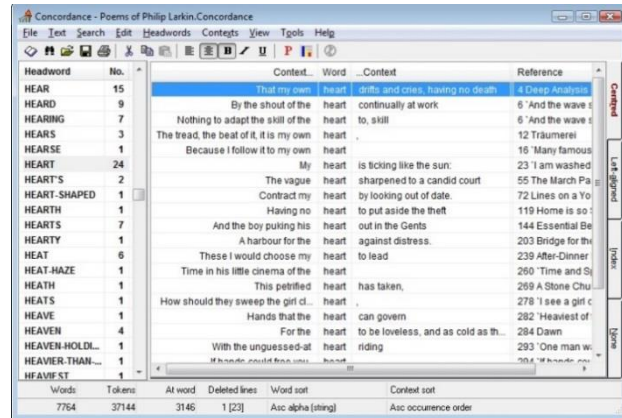
**Data structure: List**  
**Problems: Text Concordance**

### Synopsis of the topic:

In this chapter we will learn about list data structure through the operations of text concordance. The text concordances are frequently used in linguistics when studying a text.

There are many functions that are related to text concordance such as:

- comparing different usages of the same word.
- analysing keywords.
- analysing word frequencies.
- finding and analysing phrases and idioms.
- finding translations of subsentential elements, e.g. terminology, in bitexts and translation memories.
- creating indexes and word lists (also useful for publishing).



This topic will focus on the function of text analysis. The text analysis involve extraction process to identify each words from a passage. After identifying the words, it cannot directly be displayed. It should be hold by a variable for comparison purposes to prevent the same word displayed more than once. Since a passage comprises of several paragraphs and paragraph consists of many words and also symbols with unknown size (refer to About Text-Based input), list is the suitable container to hold these data. Hence in this topic we will use list to implement text analysis, where new words can be added or removed when needed in any direction (refer to About List).

### Displaying all words in a non-recurring form.

This text analysis aims to identify all words from a passage and display it in a non-recurring form. All non-recurring words will be placed in list using a push-back method. At the end, words from the list will be display.

For example, if the passage contains the following words:

*"I go to school by bus. The bus is big. The school is also big. I like big school and big bus."*

The output of this text analysis is:

I  
go  
to  
school  
by  
bus  
the  
is  
big  
also  
like  
and

### Displaying non-recurring words in sorted order

At this point, all words are kept in an unsorted order. The comparing process for determining non-recurring words requires the list to be traversed (refer to About Iterator) from the beginning of the list until (1) the position of word (if the word exist) or (2) the end of the list is reached, if the word is not exist.

Traversing the list until reaching the end will take a long implementation time. Therefore, to reduce the traversing time, elements in the list should be sorted (either in an ascending or descending order). Hence, the right position need to be determined before every new word can be inserted.

If we are using the same input passage, the output for displaying non-recurring words in sorted order will be as follows:

also  
and  
big  
bus  
by  
go  
I  
is  
like  
school  
to  
The

### Displaying non-recurring words with its frequencies in sorted order.

Besides identifying all text being used in a passage, text analysis also performed to get information about all words in the passage with its frequency. In this process, all words will be extract from the input passage and will be push into the list. If the word is new, it will be placed in the list with the default frequency is 1. If the word is already exist, then the frequency value will be add to 1.

If we use the same input passage as above, the output of the words in a sorted order will be as follows:

*also*  
*and*  
*big (4)*  
*bus (3)*  
*by*  
*go*  
*I (2)*  
*is (2)*  
*like*  
*school (3)*  
*the (2)*  
*to*

**My Note:**

**Self-activity:**

1. What are the different between string class and c-string? What is the declartion method used for text analysis? Why?
2. List out the purpose of using the iterator in text analysis.
3. Why do we need to sort elements in list?
4. Which of the problems can be used if we want to display the number of words in a passage?

## About Text analysis

Passage usually consists of combination of words, sentences, and paragraphs. It also can contain of *non-alphabetic characters* or symbols that is not a number or letter such as blank spaces, exclamation points (!), commas (,), question marks (?), periods (.), underscores (\_), apostrophes ('), and at symbols (@). Besides symbols, a passage can have a blank line used to separate one paragraph to another.

The text analysis operation will extract all words and exclude the *non-alphabetic characters* or symbols as well as the leading and trailing spaces. The approach is to use Java [String.split](#) method to split the String, s into an array of substrings. The **java lang.string.trim()** is a built-in function that eliminates leading and trailing spaces.

The following java program show the implementation of these methods.

```
1 import java.util.*;
2
3 public class MyList {
4     public static void main(String args[]) {
5         Scanner in = new Scanner(System.in);
6         String passage = "Be safe..Stay home!. #Kita jaga kita." ;
7
8         String delims = "\\W+"; // split any non word
9         String [] words = passage.split(delims);
10        for (String str: words){
11            str = str.trim();
12            System.out.println(str);
13        }
14    }
15 }
```

### Output:

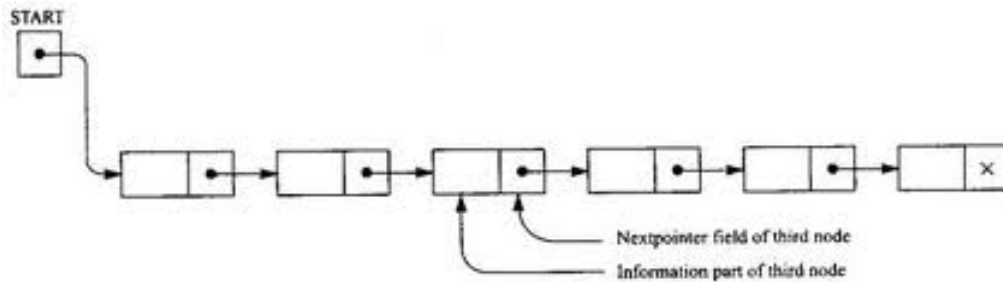
```
Be
safe
Stay
home
Kita
jaga
kita
```

## About List

List Interface in Java can be implemented using [ArrayList](#), [LinkedList](#), [Vector](#) and [Stack](#) classes. Read at [GeeksForGeeks](#) to find the differences between those classes.

In this topic, we will use array list to implement list data structure. Please refer to URL <https://www.geeksforgeeks.org/list-interface-java-examples/> for details.

List structure:



How to declare a list?:

```
List<Integer> myList = new ArrayList<Integer>();  
//myList is a list of Integer.
```

```
List<String> yourList = new ArrayList<String>();  
//yourList is a list of String.
```

The common methods used in list:

1. Add Element to list

Add new element	Code Example
<code>add(E e)</code> : This method Appends the specified element to add the end of this list.	<code>yourList.add("Ali");</code>
<code>add(int index, E element)</code> : This method Inserts the specified element at the specified position in this list.	<code>yourList.add(2, "Siram");</code>
<code>addFirst(E e)</code> : This method Inserts the specified element at the beginning of this list.	<code>myList.addFirst(5);</code>
<code>addLast(E e)</code> : This method Appends the specified element to the end of this list.	<code>myList.addLast(20);</code>

## 2. Remove elements from list

Methods to remove elements	Code Example
<code>remove()</code> : This method retrieves and removes the head (first element) of this list.	<pre>myList.remove(); yourList.remove();</pre>
<code>remove(int index)</code> : This method removes the element at the specified position in this list.	<pre>myList.remove(2);</pre>
<code>remove(Object o)</code> : This method removes the first occurrence of the specified element from this list, if it is present.	<pre>yourList.remove("Ali"); myList.remove(20);</pre>

- `getFirst()` : This method returns the first element in this list.  

```
System.out.println("First Element: " + myList.getFirst());
```
- `size()` : This method returns the number of elements in this list.  

```
int TotalElementsOfmyList = mylist.size();
```

My Note:

### Self-Activity:

Write appropriate Java statements to:

- Declare a list named *OddList* from type of integer.
- Read 10 integer values. If the value is an odd number, push it into the *OddList* as last element.
- Display all elements in *OddList*.

### Tutorial Activity:

- Describe the differences between array and arraylist. When is the right time to use array and arraylist?
- What are the differences between Stack, Queue and List?



## ABOUT The Iterator

An *iterator* is an object used to point to element in a container, particularly list. Basically it is used to traverse a list for some purposes; to display or to search the data in list. It has the ability to iterate through the elements of list using a set of methods (with at least `hasNext` and `next` methods).

Line 11 to 15 in the program below shows the example of code that iterate through *myNumbers*, starting from the first to last element of *myNumbers*.

Note that iterator it as in line 12 is used to set the iterator to point to the first element of *myNumbers*. While the method `hasNext` is to check if there is still have next data in the *myNumbers*. The method `next` is used to read the next data.

```
1 import java.util.*;
2
3 public class ListDemo {
4     // Java program to iterate over an arraylist using Iterator
5
6     public static void main(String[] args) {
7         // initializing ArrayList
8         List<Integer> myNumbers = Arrays.asList(1, 2, 3, 4, 5, 6, 7, 8);
9
10        // Looping ArrayList using Iterator
11        Iterator it = myNumbers.iterator();
12        while (it.hasNext())
13            System.out.print(it.next() + " ");
14    }
15 }
16
```

### Output:

myNumbers contains: 1 2 3 4 5 6 7 8

### Tutorial-Activity:

Write the Java statements to:

1. print the number of element with value 5 from the *myNumbers*.
2. Declare a list named *IntList* from type of integer. Use the iterator to copy all elements of *myNumbers* to *IntList*.
3. Print all elements of *IntList*.



LIST OF WORDS	
Input	Standard Input
Output	Standard Output
Java Elements	Looping
Data Structure	List
Additional Function	Trim and Remove all non-alphabetical characters

### ***Problem Description***

The List of Words problem aims to display all words from a given passage.

Your task is to write a Java program for the **List of words**.

### ***Input***

Input of this program is a passage. A passage consists of N words and symbols. Symbols that will be considered in the passage are full stop (.), comma (,), question mark (?) and exclamation mark (!).

### ***Output***

Output of the program is N lines, where each line contains a word.

### ***Sample Input-Output***

Input	Output
I go to school by bus. The bus is big. The school is also big. I like big school and big bus.	I go to school by bus The bus is big The school is also big I like big school and

	big bus
--	------------

## *Solution*

The algorithm for List of Words:

Read a passage

For all words in the passage:

Add the word at the back of list



Display the list.

Basic Structure:

```
import java.util.*;

public class ListDemo {

    public static void main(String[] args) {
        List<String> myString = new ArrayList <String>();
        Scanner in = new Scanner(System.in);

        String passage = in.nextLine(); // read input passage
        StringTokenizer st = new StringTokenizer(passage);
        while (st.hasMoreTokens()) {
            
            
        }
    }
}
```

### Full structure of worked-example program: List of Words.

```
1 import java.util.*;
2 public class MyList {
3     public static void main(String args[]){
4
5         List<String> list1 = new ArrayList <String>();
6         Scanner in = new Scanner(System.in);
7
8         String passage = in.nextLine(); // read input passage
9         String delims = "\\W+"; // split any non-word
10        String [] words = passage.split(delims);
11        for (String str: words){
12            str = str.trim();
13            list1.add(str);
14        }
15        Iterator it = list1.iterator();
16        while (it.hasNext())
17            System.out.print(it.next() + " ");
18    }
19 }
```

### Tutorial Activity:

1. Why is the purpose of list in the program?
2. What is the purpose of variable *str*?
3. Named the method used to add new string into the list? Which line to indicate this process?
4. What is the purpose of the statements in line 15 to 17?

### Hands-on Activity:

Play with list:

Mission: Create a program that receives 10 integers, add it as into *OddList* if it is an odd number. Then pop and display all the elements of *oddList*. Display your output in the following format:

<Size\_of\_list>: <the elements of the list that are separated by a space>.

Sample IO:

Input	Output
2 34 1 8 1 67 2 0 12 34 0	7: 2 34 1 8 1 67 2 2: 12 34

## Worked Example 2: List of Unique Words

# LIST OF UNIQUE WORDS

Input	Standard Input
Output	Standard Output
Data Structure	List

### *Problem Description*

List of Unique Words problem aims to display all words from a given passage in a non-recurring form. Your task is to write a Java program for the **List of Unique Words**.

### *Input*

Input of this program is a passage. A passage consists of N words and symbols. Symbols that will be considered in the passage are full stop (.), comma (,), question mark (?) and exclamation mark (!). The passage will have M unique words, where the M is less than or equal to N.

### *Output*

Output of the program is M lines, where each line contains a unique word.

### *Sample Input-Output*

Input	Output
I go to school by bus. The bus is big. The school is also big. I like big school and big bus.	I go to school by bus The is big also like and

### *Solution*

**The algorithm for List of Unique Words:**

Read a passage

For all words in the passage:

Remove unnecessary characters  
If the word not exist in list  
Add the word at the back of list

Display the list.

### Full structure of worked-example program: List of Unique Words.

```
1 import java.util.*;
2 public class MyList {
3
4     public static void main(String args[]) {
5
6         List<String> list1 = new ArrayList <String>();
7         boolean isDuplicate;
8         Scanner in = new Scanner(System.in);
9
10        String passage = in.nextLine(); // read input passage
11        String delims = "\\W+"; // split any non word
12        String [] words = passage.split(delims);
13
14        for (String str : words){
15            str = str.trim();
16            isDuplicate=false;
17            isDuplicate =CheckForDuplicates (list1, str);
18
19            if (!isDuplicate) {
20                //add new word into list and update the list size
21            }
22        }
23        // display all elements of list
24    }
25
26    static boolean CheckForDuplicates (List L1, String word) {
27        // check for duplicate words
28        // if found duplicate return true else return false
29    }
30 }
```

### Tutorial Activity:

1. Write the java code to show the if statement to check for duplicates word in the list.
2. Discuss the different between the **list of words** and **list of unique words**.
3. Specify the reason of using iterator in this problem.

### Hands-on Activity:

Play with list:

Mission: Create a program that receives sequence of integers that ends with 0, and push them onto a *intList*. Do not push if the number exist in *intList*. At the end of your program display all the elements of *intList*. Display your output in the following format:

<Size\_of\_list>: <the elements of the list that are separated by a space>.

Sample IO:

Input	Output
34 1 8 1 67 0	4: 34 1 8 67
12 34 0	2: 12 34

### Worked Example 3: List of Sorted Words

## LIST OF SORTED UNIQUE WORDS

Input	Standard Input
Output	Standard Output
JAVA Elements	Selection, Looping,
Data Structure	List

### *Problem Description*

List of sorted words problem aims to display all words from a given passage in an ascending order. Your task is to write a JAVA program for the **List of Sorted Words**.

**Program Scope:** Value of 'A' is smaller than 'a'. Each character is based on the value of **ASCII CODE**, where ASCII CODE of character 'A' is 65, and character 'a' is 97. This means that the word "The" is greater than word "also".

### *Input*

Input of this program is a passage. A passage consists of N words and symbols. Symbols that will be considered in the passage are full stop (.), comma (,), question mark (?) and exclamation mark (!). The passage will have M unique words, where M is less than or equal to N.

### *Output*

Output of the program is M lines, where each line contains a word followed by symbol (, then followed by an integer that represents the word frequency and ends by symbol).

### *Sample Input-Output*

Input	Output
I go to school by bus. The bus is big. The school is also big. I like big school and big bus.	I The also and big bus by go is like school to

## Solution

### The algorithm for List of Sorted Words:

Read a passage

For all words in the passage:

If the word not exist in list  
Add the word into the list  
Sort the list

Display the list.

### Full structure of worked-example program: List of Sorted Words.

```
1  import java.util.*;
2  public class MyList {
3      public static void main(String args[]) {
4          List<String> list1 = new ArrayList <String>();
5          Scanner in = new Scanner(System.in);
6          boolean isDuplicate;
7          int size=0;
8
9
10         String passage = in.nextLine(); // read input passage
11         String delims = "\\W+"; // split any non word
12         String [] words = passage.split(delims);
13         for (String str : words){
14             str = str.trim();
15             isDuplicate=false;
16             isDuplicate =CheckForDuplicates (list1, str);
17             if (!isDuplicate) {
18                 //add new word into list and update the list size
19             }
20             //sort the list
21         }
22         displayList(list1, size);
23     }
24
25     static boolean CheckForDuplicates (List l1, String word) {
26         // check for duplicate words
27         // if found duplicate return true else return false
28     }
29
30     static void displayList(List l1, int s) {
31         // use the iterator to iterate list
32         // and display all elements
33     }
34 }//The Program Ends Here
```

### Tutorial Activity:

1. State the differences between List of Unique Words and List of Sorted Words.
2. Name the method use to add new word into list.
3. Name the user defined functions used in List of Sorted Unique Words and its purpose.
4. Complete the CheckForDuplicates() and displayList() methods.
5. Write the statement to sort the list.

## Problem: Word Frequencies

# WORD FREQUENCIES

Input	Standard Input
Output	Standard Output
Data Structure	List

### *Problem Description*

Write a JAVA program that will display all words from a given passage with its frequencies in an ascending order.

### *Input*

Input of this program is a passage. A passage consists of N words and symbols. Symbols that will be considered in the passage are full stop (.), comma (,), question mark (?) and exclamation mark (!). The passage will have M unique words, where the M is less than or equal to N.

### *Output*

Output of the program is M lines, where each line contains a word followed by symbol (, then followed by an integer that represent the word frequency and ends by symbol).

### *Sample Input-Output*

Input	Output
I go to school by bus. The bus is big. The school is also big. I like big school and big bus.	also and big (4) bus (3) by go I (2) is (2) like school(3) The (2) to



## ***Solution***

### **The algorithm for Words Frequencies:**

Read a passage

For each word in the passage:

Search for the same word in the list

If not found:

    Add the word into the list

    Set the word's frequency to 1

If found:

    Update the word's frequency value in list

Sort the list

Display the list.