## TK1143 TUTORIAL 4 (LIST)

**Section A**

1. Describe the differences between following Structure and when is the right time to use it
   1. Array and ArrayList.

Array

* Array is static in size.
* An array is a fixed-length data structure.
* An array can store both objects and primitives type.
* We use for loop or for each loop to iterate over an array.

ArrayList

* ArrayList is dynamic in size.
* ArrayList is a variable-length data structure. It can be resized itself when needed.
* We cannot store primitive type in ArrayList. It automatically converts primitive type to object.
* We use an iterator to iterate over ArrayList.
  1. ArrayList and LinkedList

ArrayList

* ArrayList internally uses a dynamic array to store the elements.
* An ArrayList class can act as a list only because it implements List only.
* ArrayList is better for storing and accessing data.

LinkedList

* LinkedList internally uses a doubly linked list to store the elements.
* LinkedList class can act as a list and queue both because it implements List and Deque interfaces.
* LinkedList is better for manipulating data.
  1. Stack, Queue and List?

Stack

* Stack is a LIFO (Last-In, First-Out) list, a list-like structure in which elements may be inserted or removed from only one end (last-in, first-out). Stacks are less flexible than lists, but are easier to implement, and more efficient (for those operations they can do)

Queue

* Queue is a FIFO (First-In, First-Out) list, a list-like structure that provides restricted access to its elements: elements may only be inserted at the back and removed from the front.Similarly to stacks, queues are less flexible than lists.

List

* A list is a finite, ordered sequence of data items known as elements ("ordered" means that each element has a position in the list)

1. Consider a following figure of *myList* elements with the specific index.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 14 | 20 | 38 | 5 | 7 |
| 0 | 1 | 2 | 3 | 4 |

* 1. Write Java code to declare *myList* from type of integer.

List <Integer> myList = new ArrayList<Integer>();

* 1. How many elements in *myList*?

5

* 1. Write Java code to add all the elements in *myList*

myListadd(!4,20,38,5,7);

* 1. State an index of element 5.

3

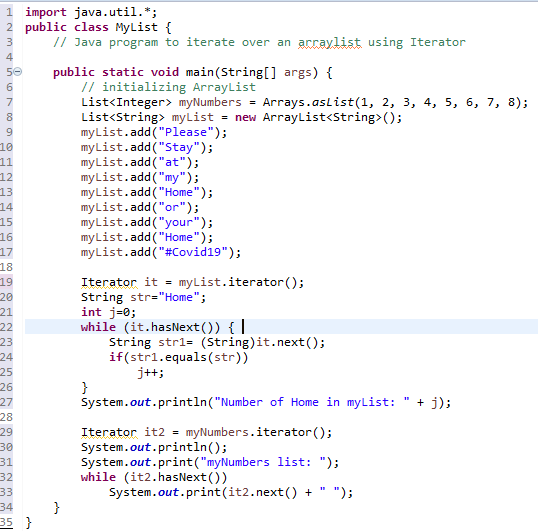
* 1. Write Java code and illustrates the elements of *myList* after adding element 15 at index 1.

myListadd(1, 15);

* 1. Write Java code and illustrates the elements of *myList* after removing element 38 at index 2.

myListremove(2);

1. Consider the following Java code.



* 1. What is the name of the iterator used in the above code?

it , it2

* 1. What is the purpose of the iterator in the above code?

The purpose of iterator is used to traverse each and every element in myList.

it: determines how many string “Home: is in the myList.

it2: determines how many element in myNumbers.

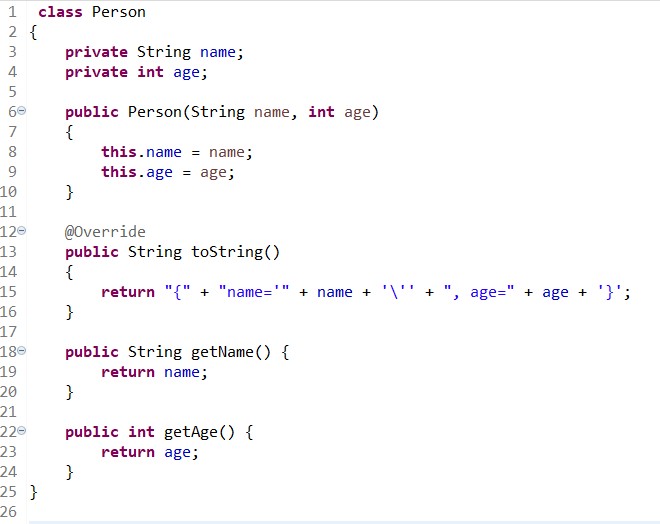
* 1. Write the output of the code.

Number of Home in myList: 2

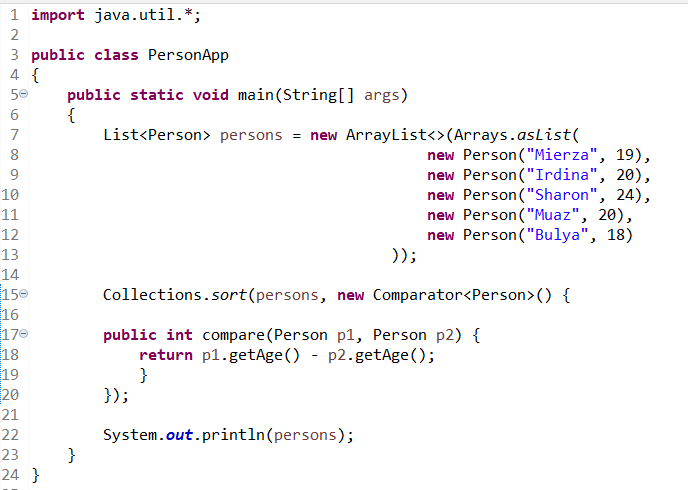
myNumbers List: 1 2 3 4 5 6 7 8

## Section B

1. Based on the Class Person and PersonApp answer the following question.



Class Person



Class PersonApp

* 1. Can you explain class PersonApp code in line 15-18 ? What is the purpose of that

code?

The line in 15-18 means that it sorts person P1 and P2 by age. The purpose of it is to determine the age difference from the list who is younger.

* 1. What is the output from this code?

[{name='Bulya', age=18}, {name='Mierza', age=19}, {name='Irdina', age=20} ,{name='Muaz', age=20}, {name='Sharon', age=24}]

1. Write a java program that read list of integer numbers from input user. If the number is even add it into *evenList*. Otherwise add to *oddList*. At the end of the code, display the elements of the odd and even numbers list respectively with ascending order. Display your output in the following format:

< Odd or Even list> <Size\_of\_list in a bracket>: <the elements of the list that are separated by a space>

[Note: The code must use list and iterator]

Sample IO:

|  |  |
| --- | --- |
| **Input** | **Output** |
| 2 3 0 -15 8 22 -11 6 -7 18 | Odd List (4): -15 -11 -7 3  Even List (6): 0 2 6 8 18 22 |

import java.util.\*;

public class EvenOdd {

public static void main(String[] args) {

Scanner input=new Scanner(System.in);

List<Integer> myList = new ArrayList<Integer>();

for(int x = 0; x<10; x++) {

int i = input.nextInt();

myList.add(i);

}

Collections.sort(myList);

List<Integer> evenList = new ArrayList<Integer>();

List<Integer> oddList = new ArrayList<Integer>();

Iterator it = myList.iterator();

int m=0;

int j=0;

while(it.hasNext()) {

int k=(int)it.next();

if(k%2==0) {

evenList.add(k);

m++;

}

else if(k%2!=0) {

oddList.add(k);

j++;

}

}

Iterator it2 = evenList.iterator();

Iterator it3 = oddList.iterator();

System.out.print("Odd List ("+j+"):");

while(it3.hasNext()) {

System.out.print(it3.next()+" ");

}

System.out.print("\nEven List ("+m+"):");

while(it2.hasNext()) {

System.out.print(it2.next()+" ");

}

}

}

1. Create a program that receives sequence of integers that ends with 0. For every non-repeating number add it into *NumberList*. Display the size and all list elements. Then display all list elements again after composing them in an ascending order. Follow the following output format.

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

[Note: The code must use list and iterator]

Sample IO:

import java.util.\*;

public class Rearrange {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

List<Integer> myList = new ArrayList<Integer>();

int value = input.nextInt(); //user input num

for (int i = 0; value!=0 ; i++) { // repeat until value 0 is inserted

if(!myList.contains(value)) {

myList.add(value);

}

value = input.nextInt(); //user input num again

}

Iterator<Integer> it = myList.iterator();

System.out.print(myList.size()+ " : ");

while (it.hasNext()) {

Integer num = (Integer) it.next();

System.out.print(num+ " ");

}

Collections.sort(myList);

Iterator<Integer> it2 = myList.iterator();

System.out.print("\n"+myList.size()+ " : ");

while (it2.hasNext()) {

System.out.print(it2.next()+ " ");

}

}

}

|  |  |
| --- | --- |
| **Input** | **Output** |
| 5 1 2 1 1 1 6 2 1 0 | 4: 5 1 2 6  4: 1 2 5 6 |

## Section C :

There are four (4) questions given in this section. Your task is to write the program to implement data structure List for each question.

|  |  |
| --- | --- |
| **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 all words.

## 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 String delims = "\\W+"; // split any non-word

String [] words = passage.split(delims);

**for** (String str: words){

}

}

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

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | **import** java.util.\*;  **public class** MyList {  **public static void** main(String args[]){  List<String> list1 = **new** ArrayList <String>(); Scanner in = **new** Scanner(System.***in***);  String passage = in.nextLine(); // read input passage String delims = "\\W+"; // split any non-word  String [] words = passage.split(delims);  **for** (String str: words){  //remove leading and trailing spaces  //add the word athe back of list  }  //display all word  }  } |

## Tutorial Activity:

1. What 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?

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

## 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.

|  |  |
| --- | --- |
| **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 that ‘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 mean 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 the M is less than or equal to N.

### Output

Output of the program is M lines, where each line contains a list of **Sorted** 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  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 all word.

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

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33 | **import** java.util.\*;  **public class** MyList {  **public static void** main(String args[]) { List<String> list1 = **new** ArrayList <String>(); Scanner in = **new** Scanner(System.***in***);  **boolean** isDuplicate;  **int** size=0;  String passage = in.nextLine(); // read input passage String delims = "\\W+"; // split any non word  String [] words = passage.split(delims);  **for** (String str : words){ str = str.trim(); isDuplicate=false;  isDuplicate =*CheckForDuplicates* (list1, str);  **if** (!isDuplicate) {  //add new word into list and update the list size  }  //sort the list  }  *displayList*(list1, size);  }  **static boolean** CheckForDuplicates (List L1, String word) {  // check for duplicate words  // if found duplicate return true else return false  }  **static void** displayList(List l1, **int** s) {  // use the iterator to iterate list  // and display all elements  }  }//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.

|  |  |
| --- | --- |
| **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. At the end of output, print the text analysis such as (i) **Total words**, (ii) **Number of Repeated words**, (iii) **Number of Unique words** and (iv) **Most used word**.

# 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). Display the analysis of Total words, Number of repeated words, Number of unique words and Most used word as shown in Sample Input- Output.

# 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(2)  The(2) also(1) and(1)  big(4)  bus(3)  by(1)  go(1)  is(2) like(1) school(3) to(1) |

|  |  |
| --- | --- |
|  | Total words: 22  Number of repeated words: 6 Number of unique words: 12 Most used word: big |

### 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 and add the word’s frequency to 1

If found:

Update the word’s frequency value in list

Sort the list



Display the list and text analysis

Class Data

\*\* You must use class Data to complete question 4 in Section C for submission to CodeZinger

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40 | **import** java.util.Comparator;  // Class Data  **public class** Data {  String word;  **int** freq;  **public** Data (String item){  **this**.word =item;  **this**.freq=1;  }  **public** String getWord() {  **return** word;  }  **public void** setWord(String newword) {  **this**.word= newword;  }  **public int** getFreq() {  **return** freq;  }  **public void** setFreq(**int** freq2) {  **this**.freq = freq2;  }  **public static** Comparator<Data> *WordComparator* = **new** Comparator<Data>()  {  // Used for sorting in ascending order of word  **public int** compare(Data a, Data b)  {  String word1 = a.getWord(); String word2 = b.getWord();  **return** (word1).compareTo(word2); //string1.compareTo(string2)  }  };  } |