

TK1143

TUTORIAL 4 (LIST)

Section A

1. Describe the differences between following Structure and when is the right time to use it
 - a) 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.

- b) 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.

- c) 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)

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

14	20	38	5	7
0	1	2	3	4

- a) Write Java code to declare *myList* from type of integer.

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

- b) How many elements in *myList*?

5

- c) Write Java code to add all the elements in *myList*

```
myList.add(14,20,38,5,7);
```

- d) State an index of element 5.

3

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

```
myList.add(1, 15);
```

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

```
myList.remove(2);
```

3. Consider the following Java code.

```

1 import java.util.*;
2 public class MyList {
3     // Java program to iterate over an arraylist using Iterator
4
5     public static void main(String[] args) {
6         // initializing ArrayList
7         List<Integer> myNumbers = Arrays.asList(1, 2, 3, 4, 5, 6, 7, 8);
8         List<String> myList = new ArrayList<String>();
9         myList.add("Please");
10        myList.add("Stay");
11        myList.add("at");
12        myList.add("my");
13        myList.add("Home");
14        myList.add("or");
15        myList.add("your");
16        myList.add("Home");
17        myList.add("#Covid19");
18
19        Iterator it = myList.iterator();
20        String str="Home";
21        int j=0;
22        while (it.hasNext()) { |
23            String str1= (String)it.next();
24            if(str1.equals(str))
25                j++;
26        }
27        System.out.println("Number of Home in myList: " + j);
28
29        Iterator it2 = myNumbers.iterator();
30        System.out.println();
31        System.out.print("myNumbers list: ");
32        while (it2.hasNext())
33            System.out.print(it2.next() + " ");
34    }
35 }

```

a) What is the name of the iterator used in the above code?

it , it2

b) 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.

c) 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.

```

1  class Person
2  {
3      private String name;
4      private int age;
5
6      public Person(String name, int age)
7      {
8          this.name = name;
9          this.age = age;
10     }
11
12     @Override
13     public String toString()
14     {
15         return "{" + "name='" + name + '\'' + ", age=" + age + '}';
16     }
17
18     public String getName() {
19         return name;
20     }
21
22     public int getAge() {
23         return age;
24     }
25 }
26

```

Class Person

```

1  import java.util.*;
2
3  public class PersonApp
4  {
5      public static void main(String[] args)
6      {
7          List<Person> persons = new ArrayList<>(Arrays.asList(
8              new Person("Mierza", 19),
9              new Person("Irdina", 20),
10             new Person("Sharon", 24),
11             new Person("Muaz", 20),
12             new Person("Bulya", 18)
13         ));
14
15         Collections.sort(persons, new Comparator<Person>() {
16
17             public int compare(Person p1, Person p2) {
18                 return p1.getAge() - p2.getAge();
19             }
20         });
21
22         System.out.println(persons);
23     }
24 }

```

Class PersonApp

- a) 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.

- b) 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}]
```

2. 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()+" ");
        }
    }
}

```

3. 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:

Input	Output
5 1 2 1 1 1 6 2 1 0	4: 5 1 2 6 4: 1 2 5 6

```
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()+ " ");

        }
    }
}
```

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  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             //remove leading and trailing spaces
13             //add the word at the back of list
14         }
15             //display all word
16     }
17 }

```

Tutorial Activity:

1. What is the purpose of list in the program?
2. What is the purpose of variable *str*?
3. Name 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  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.

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 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  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          String passage = in.nextLine(); // read input passage
10         String delims = "\\W+"; // split any non word
11         String [] words = passage.split(delims);
12         for (String str : words){
13             str = str.trim();
14             isDuplicate=false;
15             isDuplicate =CheckForDupl//icates (list1, str);
16             if (!isDuplicate) {
17                 //add new word into list and update the list size
18             }
19             //sort the list
20         }
21         displayList(list1, size);
22     }
23     static boolean CheckForDuplicates (List L1, String word) {
24         // check for duplicate words
25         // if found duplicate return true else return false
26     }
27
28     static void displayList(List l1, int s) {
29         // use the iterator to iterate list
30         // and display all elements
31     }
32 }//The Program Ends Here
33

```

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

```