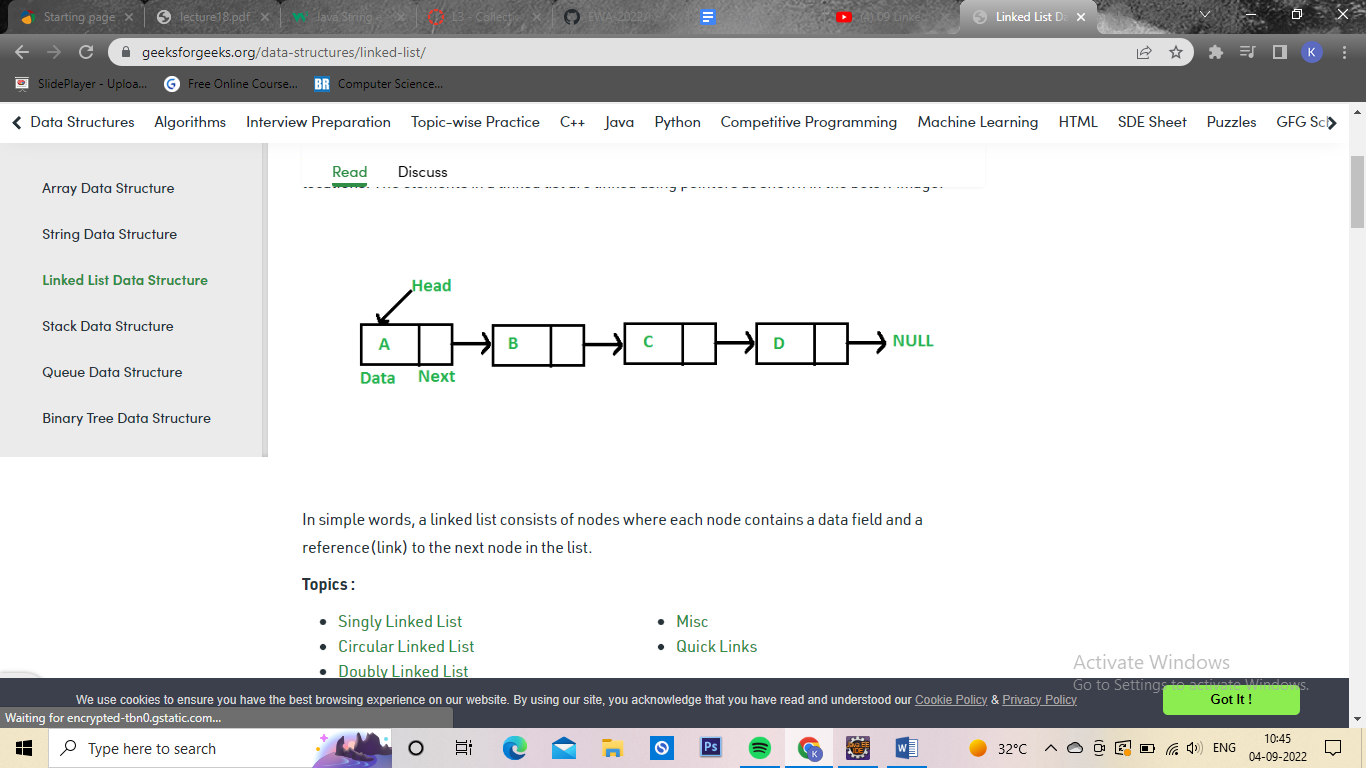
**EWA** PRACTICAL 2(Documentation)

Khyati Dhawan

Linked List and Iterators

LINKED LIST & ITERATORS



# INTRODUCTION

**Linked list**:

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers .

In simple words, a linked list consists of nodes where each node contains a data field and a reference(link) to the next node in the list.

**Linked List (Characteristics):**

1.A generic class (just like ArrayList) present inside Java.util package

2.It can contain duplicate elements

3.Manipulation is fast as no shifting is required

## Array Vs LinkedList

## Advantages of Linked Lists over arrays:

* Dynamic Array.
* Ease of Insertion/Deletion.

## Drawbacks of Linked Lists:

* Random access is not allowed. We have to access elements sequentially starting from the first node(head node). So we cannot do a [binary search with linked lists](https://www.geeksforgeeks.org/binary-search-on-singly-linked-list/) efficiently with its default implementation.
* Extra memory space for a pointer is required with each element of the list.
* Not cache friendly. Since array elements are contiguous locations, there is locality of reference which is not there in case of linked lists.

## Types of Linked Lists:

* **Simple Linked List** – In this type of linked list, one can move or traverse the linked list in only one direction
* **Doubly Linked List** – In this type of linked list, one can move or traverse the linked list in both directions (Forward and Backward)
* **Circular Linked List** – In this type of linked list, the last node of the linked list contains the link of the first/head node of the linked list in its next pointer and the first/head node contains the link of the last node of the linked list in its prev pointer

## Basic operations on Linked Lists:

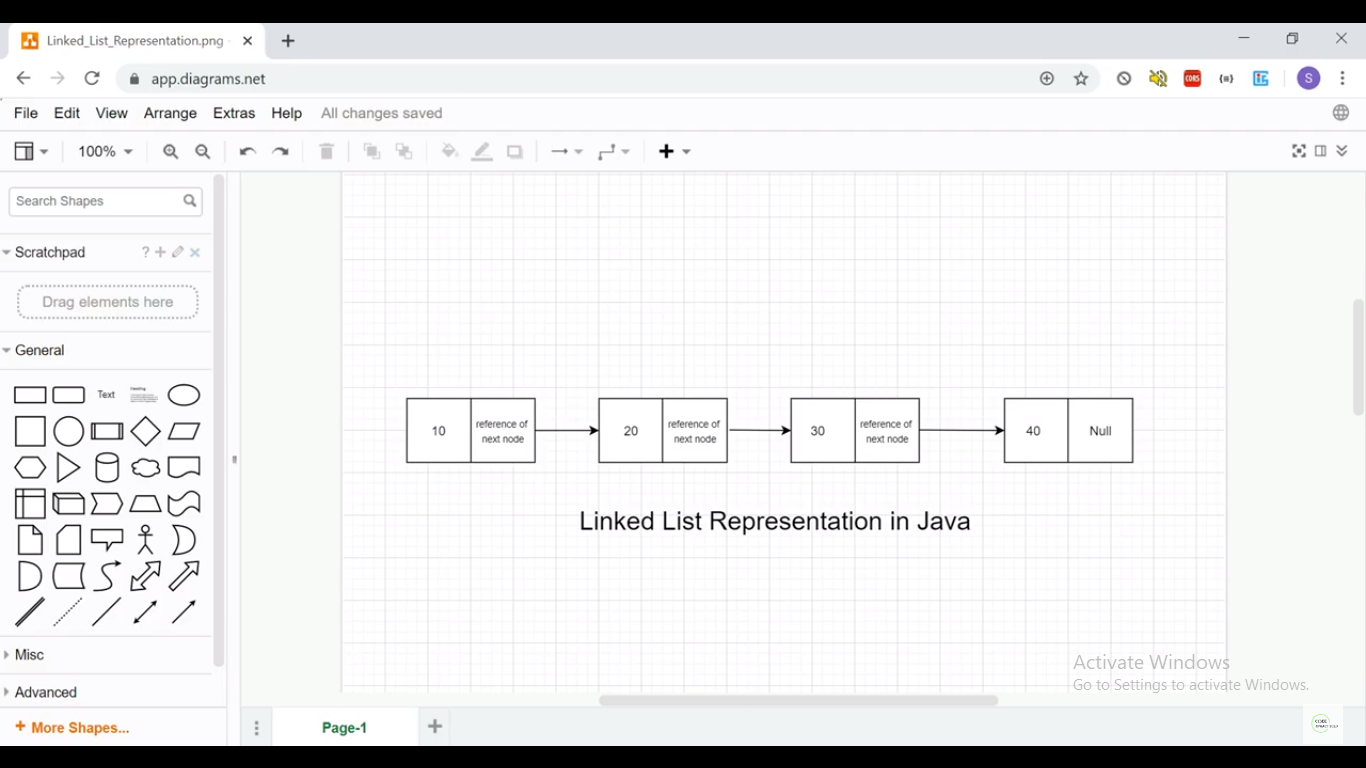
* [Deletion](https://www.geeksforgeeks.org/linked-list-set-3-deleting-node/)
* [Insertion](https://www.geeksforgeeks.org/linked-list-set-2-inserting-a-node/)
* [Search](https://www.geeksforgeeks.org/search-an-element-in-a-linked-list-iterative-and-recursive/)
* Display

## Representation of Linked Lists:

A linked list is represented by a pointer to the first node of the linked list. The first node is called the head of the linked list. If the linked list is empty, then the value of the head points to NULL.

Each node in a list consists of at least two parts:

* A Data Item (we can store integer, strings, or any type of data).
* Pointer (Or Reference) to the next node (connects one node to another) or An address of another node



## ITERATOR:

available in Collection framework under java.util package.

Help in Traversing the list.

It is applicable for all Collection classes. Hence, it is also known as the Universal Java Cursor.

supports both READ and REMOVE Operations.

Syntax of Iterator:

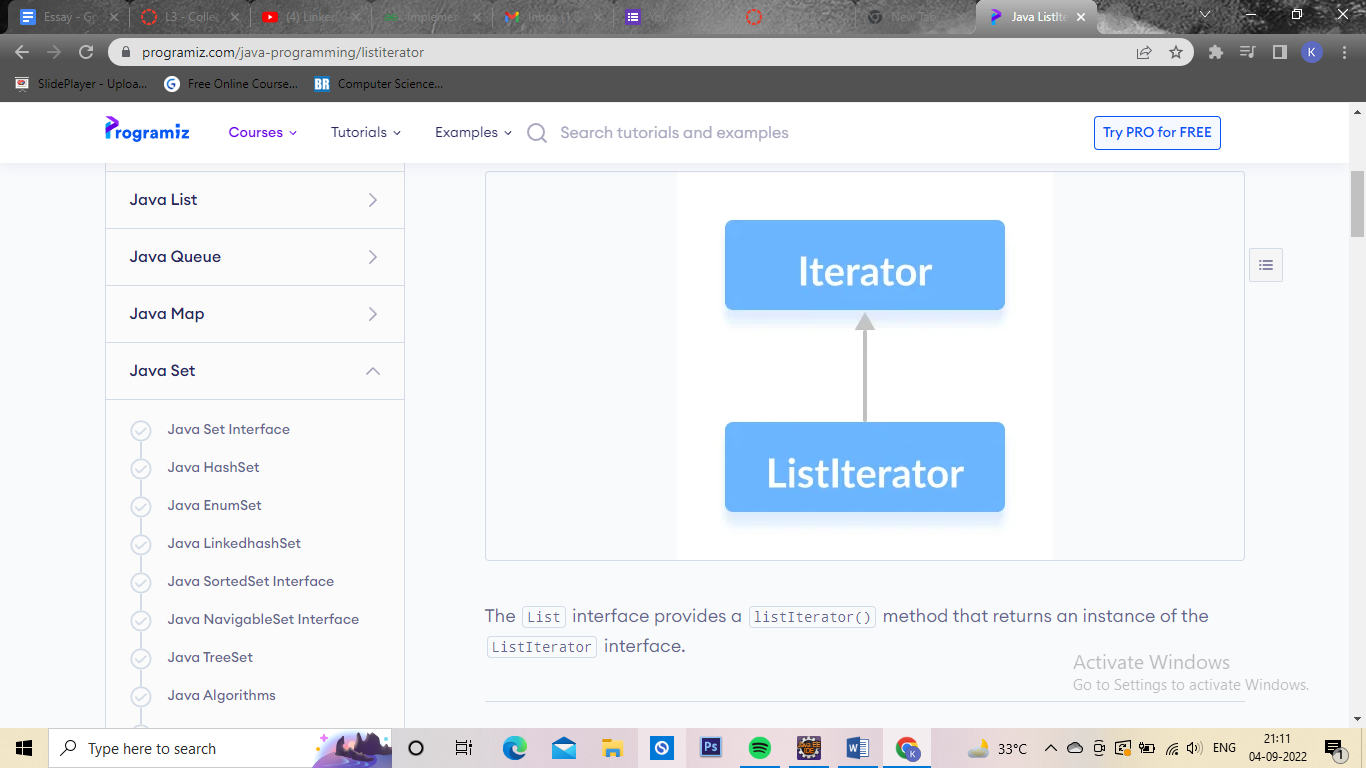
Iterator<type> itr = listRef.iterator()

**LISTITERATOR:**

The ListIterator interface of the Java collections framework provides the functionality to access elements of a list.

It is bidirectional. This means it allows us to iterate elements of a list in both directions.

It extends the Iterator interface.



## Methods of ListIterator

The ListIterator interface provides methods that can be used to perform various operations on the elements of a list.

* hasNext() - returns true if there exists an element in the list
* next() - returns the next element of the list
* nextIndex() returns the index of the element that the next() method will return
* previous() - returns the previous element of the list
* previousIndex() - returns the index of the element that the previous() method will return
* remove() - removes the element returned by either next() or previous()
* set() - replaces the element returned by either next() or previous() with the specified element

**Syntax:**

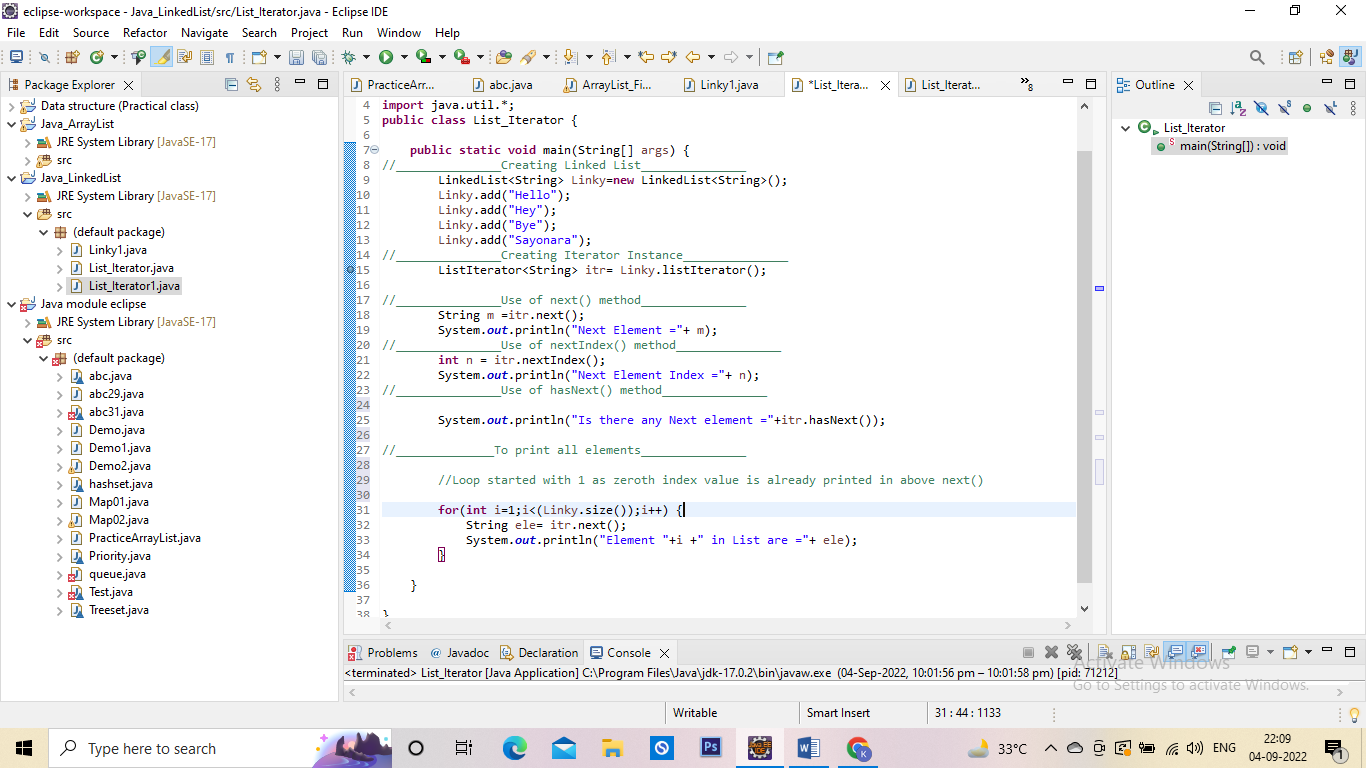
(note:numbers is List name here)

ListIterator<Integer> iterate = numbers.listIterator();

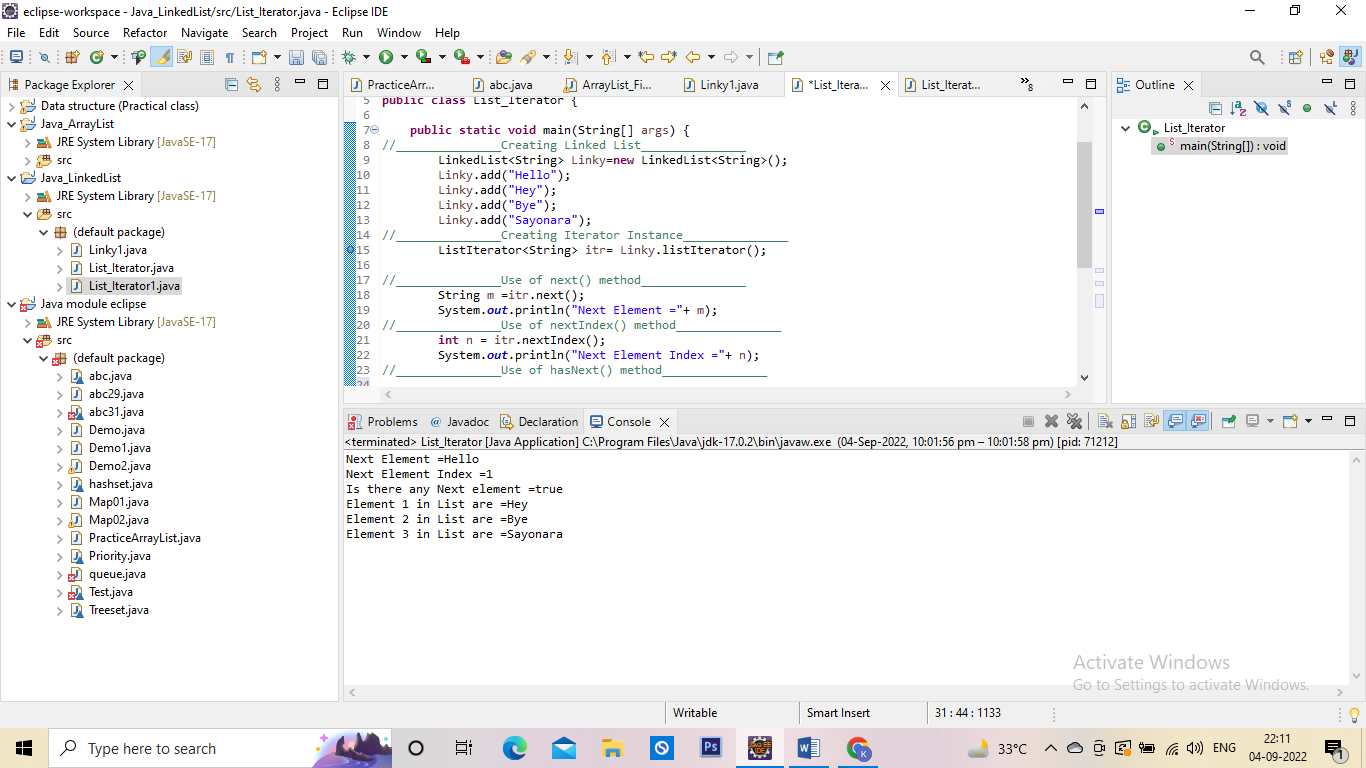
Example 1:

below, we have implemented the

next(), nextIndex() and hasNext() methods of the ListIterator interface in an Linked list.



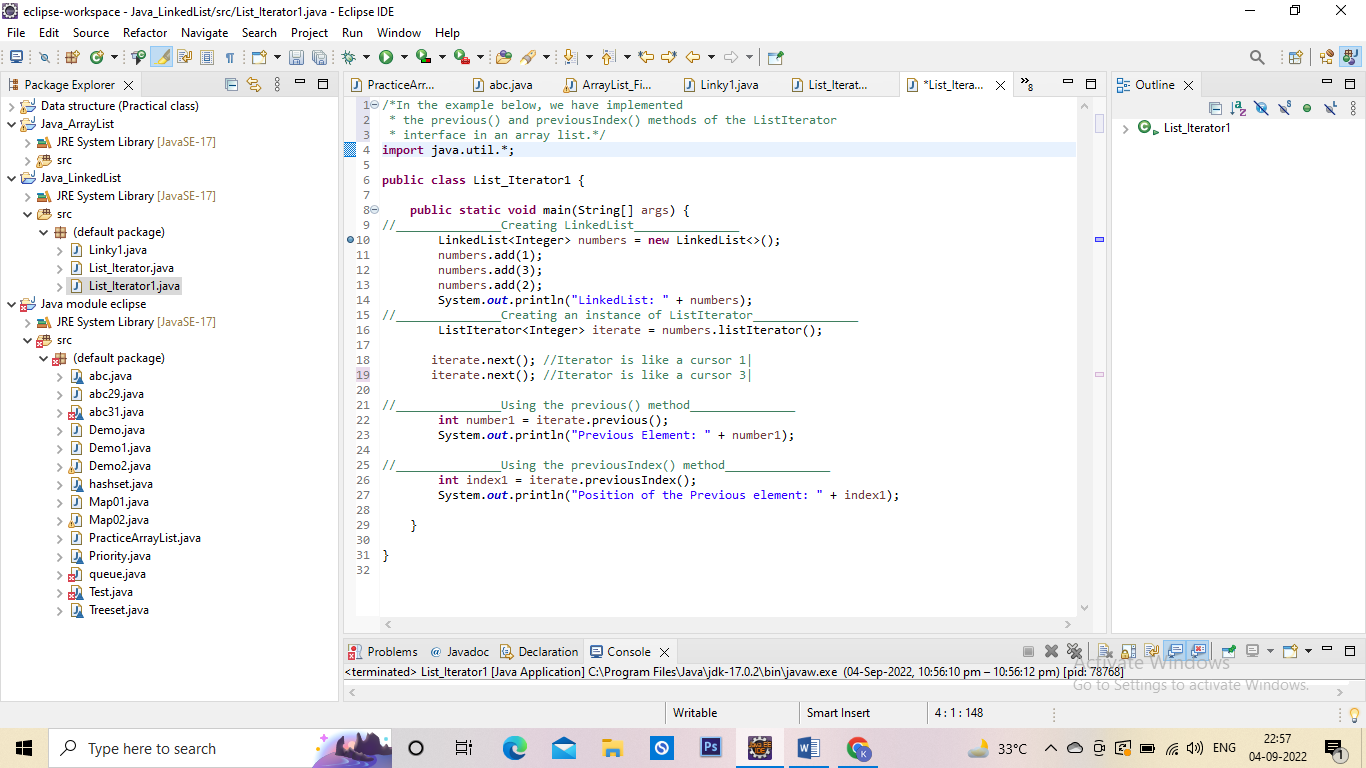
Output:



Example 2:

we have implemented the

previous() and previousIndex() methods of the ListIterator interface in a Linked list.



Output:

