**List: Array List & Linked List**

**ArrayList is Dynamic means array can be maximized,minimized,elements can be modified,modifocation of first and any element in the array is possible .**

Collection ->

//import java.util.ArrayList;

import java.util.List.\*;

public class ArrayListDemo

{

public static void main (String[]args)

{

// Creation of ArrayList

ArrayList <Integer> al = new ArrayList <Integer>();

//adding the elements into the list

al.add (10);

al.add (30);

al.add (40);

al.add (50);

//Display elements of the List and its Size

System.out.println (al);

System.out.println (al.size ());

//inserting an element into the List at specified location

al.add (1, 77);

System.out.println (al);

System.out.println (al.size ());

//modifying the existing element of the List by specifying its value

al.remove (new Integer (30));

System.out.println (al);

System.out.println (al.size ());

//removing the element of the List by specifying its index position

al.remove (0);

System.out.println (al);

System.out.println (al.size ());

//Displaying elements of List 1 by 1 using for loop (accessing)

for (int i = 0; i < al.size (); i++)

{

System.out.println (al.get (i));

}

//Displaying elements of List 1 by 1 using forEach Loop (auto-unboxing)

for (int v:al)

{

System.out.println (v);

}

//Searching an element

System.out.println (al.contains (50));

//Copying the array list into another list

ArrayList < Integer > al1 = new ArrayList < Integer > (al);

System.out.println (al1);

}

}

In add(), remove(), get(), set() methods if we write any wrong index which is not available then it will throw a run time exception called IndexOutOfBoundException. ArrayList supports storing multiple null values.

ArrayList Example with Complex Data type in Java:

import java.util.\*;

class Customer

{

String name;

int balance;

int id;

//Costructor

Customer (String s, int i, int j)

{

name = s;

balance = i;

id = j;

}

public static void main (String args[])

{

// ArrayList will contain a collection of Customer's objects.

ArrayList < Customer > arr = new ArrayList < Customer > ();

//Creating Customer objects.

Customer customer1 = new Customer ("Jay", 1000, 2);

Customer customer2 = new Customer ("Shane", 7000, 3);

Customer customer3 = new Customer ("Ricky", 5000, 1);

Customer customer4 = new Customer ("Tom", 3000, 6);

Customer customer5 = new Customer ("Mick", 6000, 4);

//Storing objects in an ArrayList collection class.

arr.add (customer1);

arr.add (customer2);

arr.add (customer3);

arr.add (customer4);

arr.add (customer5);

for (Customer c:arr)

System.out.println (c);

}

}

**LinkedList**

**LinkedList is the implementation class of List Interface which is also used to store a group of individual objects where duplicate values are allowed. LinkedList internally follows a doubly linked list structure where all the elements are stored in the form of nodes that linked each other.**

**The LinkedList in Java is not a synchronized class. LinkedList also supports multiple null values. This provides the functionality of LinkedList Data Structure.**



**Each element in a linked list is known as a node. It consists of 3 fields:**

**Prev – stores an address of the previous element in the list. It is null for the first element.**

**Next – Stores an address of the next element in the list. It is null for the last element.**

**Data – Stores the actual data**

**Note: Elements in linked lists are not stored in sequence. Instead, they are scattered and connected through links (Prev and Next).**

import java.util.\*;

class LinkedListDemo

{

public static void main (String[]args)

{

LinkedList < String > animals = new LinkedList <> ();

// Add elements to LinkedList

animals.add ("Dog");

animals.add ("Cat");

animals.add ("Horse");

System.out.println ("LinkedList: " + animals);

// Get the element from the linked list

String str = animals.get (1);

System.out.print ("Element at index 1: " + str);

System.out.println (" ");

//Iterator method

Iterator < String > itr = animals.iterator ();

while (itr.hasNext ())

{

System.out.println (itr.next ());

}

}

}

**LinkedList Example with Complex Data type in Java:**

import java.util.ArrayList;

import java.util.LinkedList;

class Student

{

String name;

int age;

Student(String na, int ag)

{

name = na;

age = ag;

}

public String toString ()

{

return "Name : " + name + " Age : " + age;

}

}

public class LinkedListDemo

{

public static void main (String[]args)

{

LinkedList < Student > list1 = new LinkedList < Student > ();

list1.add (new Student ("Haj", 20));

list1.add (new Student ("Raj", 19));

list1.add (new Student ("Sar", 18));

list1.add (new Student ("Kan", 17));

for (Student x:list1)

{

System.out.println (x);

}

}

}

1 2 3 4 5 6 7 8

**Difference Between ArrayList and LinkedList in Java:**

ArrayList is slower in insertion and deletion of elements because it internally requires shifting operations, but faster in accessing the elements because ArrayList uses index position for every element.

LinkedList is faster in insertion and deletion of elements because it just requires modifying the links of nodes instead of shifting operations, but slower in accessing the elements because LinkedList does not use any index position.