## National University of Computer & Emerging Sciences, Karachi

**Computer Science Department Spring 2024, Lab Manual – 02**

|  |  |
| --- | --- |
| **Course Code: CL-217** | **Course: Object Oriented Programming Lab** |
| **Instructor(s) :** | **Ali Fatmi** |

LAB - 2 INTRODUCTION TO Array list and Vectors

**Introduction**

A vector is similar to a dynamic array whose size can be increased or decreased. Unlike arrays, it has no size limit and can store any number of elements. Since Java 1.2, it has been a part of the Java Collection framework. It's in the java.util package and implements the List interface, so we can use all of the List interface's methods here. Let us learn about Vector in Java Along with their declarations, constructors, methods and Examples.

|  |  |
| --- | --- |
| **Sr.No.** | **Constructor & Description** |
| 1 | **Vector( )** |
| This constructor creates a default vector, which has an initial size of 10. |
| 2 | **Vector(int size)** |
| This constructor accepts an argument that equals to the required size, and creates a vector whose initial capacity is specified by size. |
| 3 | **Vector(int size, int incr)** |
| This constructor creates a vector whose initial capacity is specified by size and whose increment is specified by incr. The increment specifies the number of elements to allocate each time that a vector is resized upward. |
| 4 | **Vector(Collection c)** |
| This constructor creates a vector that contains the elements of collection c. |
| Apart from the methods inherited from its parent classes, Vector defines the following methods − | |

|  |  |
| --- | --- |
| **Sr.No.** | **Method & Description** |
| 1 | **void add(int index, Object element)** |
| Inserts the specified element at the specified position in this Vector. |
| 2 | **boolean add(Object o)** |
| Appends the specified element to the end of this Vector. |
| 3 | **boolean addAll(Collection c)** |
| Appends all of the elements in the specified Collection to the end of this Vector, in the order that they are returned by the specified Collection's Iterator. |
| 4 | **boolean addAll(int index, Collection c)** |
| Inserts all of the elements in in the specified Collection into this Vector at the specified position. |
| 5 | **void addElement(Object obj)** |
| Adds the specified component to the end of this vector, increasing its size by one. |
| 6 | **int capacity()** |
| Returns the current capacity of this vector. |
| 7 | **void clear()** |
| Removes all of the elements from this vector. |
| 8 | **Object clone()** |
| Returns a clone of this vector. |
| 9 | **boolean contains(Object elem)** |
| Tests if the specified object is a component in this vector. |
| 10 | **boolean containsAll(Collection c)** |
| Returns true if this vector contains all of the elements in the specified Collection. |
| 11 | **void copyInto(Object[] anArray)** |
| Copies the components of this vector into the specified array. |
| 12 | **Object elementAt(int index)** |
| Returns the component at the specified index. |
| 13 | **Enumeration elements()** |
| Returns an enumeration of the components of this vector. |
| 14 | **void ensureCapacity(int minCapacity)** |
| Increases the capacity of this vector, if necessary, to ensure that it can hold at least the number of components specified by the minimum capacity argument. |
| 15 | **boolean equals(Object o)** |
| Compares the specified Object with this vector for equality. |
| 16 | **Object firstElement()** |
| Returns the first component (the item at index 0) of this vector. |
| 17 | **Object get(int index)** |
| Returns the element at the specified position in this vector. |
| 18 | **int hashCode()** |
| Returns the hash code value for this vector. |
| 19 | **int indexOf(Object elem)** |
| Searches for the first occurence of the given argument, testing for equality using the equals method. |
| 20 | **int indexOf(Object elem, int index)** |
| Searches for the first occurence of the given argument, beginning the search at index, and testing for equality using the equals method. |
| 21 | **void insertElementAt(Object obj, int index)** |
| Inserts the specified object as a component in this vector at the specified index. |
| 22 | **boolean isEmpty()** |
| Tests if this vector has no components. |
| 23 | **Object lastElement()** |
| Returns the last component of the vector. |
| 24 | **int lastIndexOf(Object elem)** |
| Returns the index of the last occurrence of the specified object in this vector. |
| 25 | **int lastIndexOf(Object elem, int index)** |
| Searches backwards for the specified object, starting from the specified index, and returns an index to it. |
| 26 | **Object remove(int index)** |
| Removes the element at the specified position in this vector. |
| 27 | **boolean remove(Object o)** |
| Removes the first occurrence of the specified element in this vector, If the vector does not contain the element, it is unchanged. |
| 28 | **boolean removeAll(Collection c)** |
| Removes from this vector all of its elements that are contained in the specified Collection. |
| 29 | **void removeAllElements()** |
| Removes all components from this vector and sets its size to zero. |
| 30 | **boolean removeElement(Object obj)** |
| Removes the first (lowest-indexed) occurrence of the argument from this vector. |
| 31 | **void removeElementAt(int index)** |
| removeElementAt(int index). |
| 32 | **protected void removeRange(int fromIndex, int toIndex)** |
| Removes from this List all of the elements whose index is between fromIndex, inclusive and toIndex, exclusive. |
| 33 | **boolean retainAll(Collection c)** |
| Retains only the elements in this vector that are contained in the specified Collection. |
| 34 | **Object set(int index, Object element)** |
| Replaces the element at the specified position in this vector with the specified element. |
| 35 | **void setElementAt(Object obj, int index)** |
| Sets the component at the specified index of this vector to be the specified object. |
| 36 | **void setSize(int newSize)** |
| Sets the size of this vector. |
| 37 | **int size()** |
| Returns the number of components in this vector. |
| 38 | **List subList(int fromIndex, int toIndex)** |
| Returns a view of the portion of this List between fromIndex, inclusive, and toIndex, exclusive. |
| 39 | **Object[] toArray()** |
| Returns an array containing all of the elements in this vector in the correct order. |
| 40 | **Object[] toArray(Object[] a)** |
| Returns an array containing all of the elements in this vector in the correct order; the runtime type of the returned array is that of the specified array. |
| 41 | **String toString()** |
| Returns a string representation of this vector, containing the String representation of each element. |
| 42 | **void trimToSize()** |
| Trims the capacity of this vector to be the vector's current size. |

/ Java code illustrating Vector Constructors

**import** java.util.\*;

**public** **class** Main{

**public** **static** **void** main(String[] args)

    {

            // create default vector

            Vector v1 = **new** Vector();

        // create a vector of given Size

            Vector v2 = **new** Vector(20);

        // create a vector of given Size and Increment

            Vector v3 = **new** Vector(30,10);

            v2.add(100);

            v2.add(100);

            v2.add(100);

        // create a vector with given collection

            Vector v4 = **new** Vector(v2);

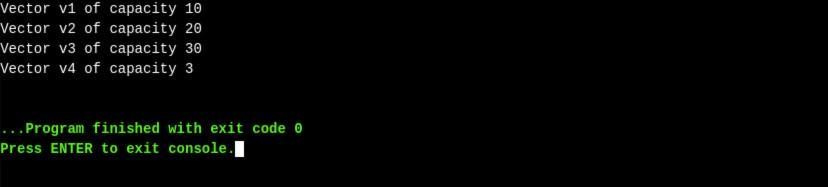
        System.out.println("Vector v1 of capacity " + v1.capacity());

        System.out.println("Vector v2 of capacity " + v2.capacity());

        System.out.println("Vector v3 of capacity " + v3.capacity());

    System.out.println("Vector v4 of capacity " + v4.capacity());

    }



// Java code showing boolean add() method

**import** java.util.\*;

**public** **class** Main{

**public** **static** **void** main (String[] args) {

        Vector v = **new** Vector();  // It creates a default vector

        v.add(1);                 // Adds 1 at the end of the list

       v.add("Java");           // Adds "Java" at the end of the list

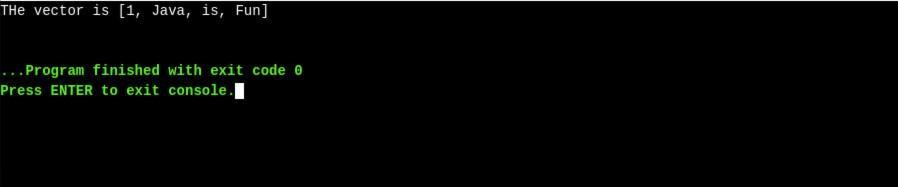
       v.add("is");             // Adds "is" at the end of the list

       v.add("Fun");            // Adds "Fun" at the end of the list

       System.out.println("The vector is " + v);

    }

}



// Java code showing void add() method

**import** java.util.\*;

**public** **class** Main{

**public** **static** **void** main (String[] args) {

        Vector v = **new** Vector();    // It creates a default vector

        v.add(0,1);                   // Adds 1 at the index 0

        v.add(1,"Java");              // Adds "Java" at the index 1

        v.add(2,"is");                // Adds "is" at the index 2

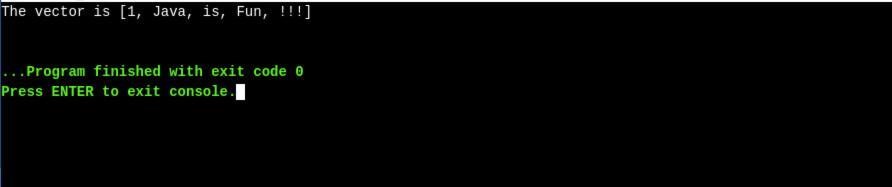
        v.add(3,"Fun");               // Adds "Fun" at the index 3

                    v.add(4,"!!!");               // Adds "Fun" at the index 4

        System.out.println("The vector is " + v);

    }

}



// Java code showing boolean remove() method

**import** java.util.\*;

**public** **class** Main{

**public** **static** **void** main (String[] args) {

        Vector v = **new** Vector();    // It creates a default vector

        v.add(1);                   // Adds 1 at the end of the list

        v.add("Java");              // Adds "Java" at the end of the list

        v.add("is");                // Adds "is" at the end of the list

        v.add("Fun");               // Adds "Fun" at the end of the list

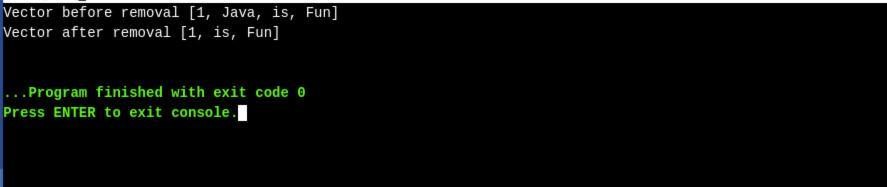
        System.out.println("Vector before removal " + v );

                v.remove(1);

                System.out.println("Vector after removal " + v );

    }

}



// Java code showing removeElement() method

**import** java.util.\*;

**public** **class** Main{

**public** **static** **void** main (String[] args) {

        Vector v = **new** Vector();    // It creates a default vector

        v.add(1);                   // Adds 1 at the end of the list

        v.add("Java");              // Adds "Java" at the end of the list

        v.add("is");                // Adds "is" at the end of the list

        v.add("Fun");               // Adds "Fun" at the end of the list

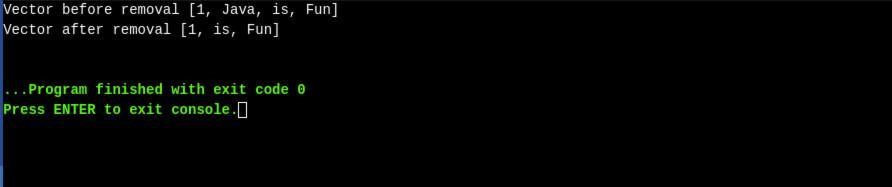
        System.out.println("Vector before removal " + v );

                v.removeElement("Java");

                System.out.println("Vector after removal " + v );

    }

}



// Java code showing size() method

**import** java.util.\*;

**public** **class** Main{

**public** **static** **void** main (String[] args) {

        Vector v = **new** Vector();    // It creates a default vector

        v.add(0,1);                   // Adds 1 at the index 0

        v.add(1,"Java");              // Adds "Java" at the index 1

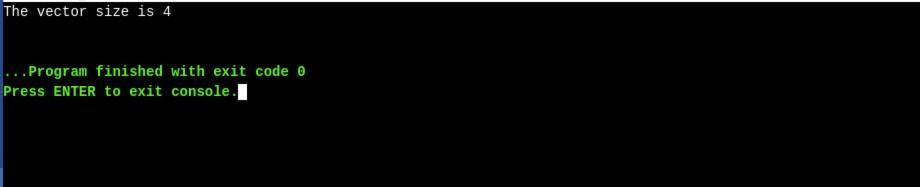
        v.add(2,"is");                // Adds "is" at the index 2

        v.add(3,"Fun");               // Adds "Fun" at the index 3

        System.out.println("The vector size is " + v.size());

    }

}



// Java code showing capacity() method

**import** java.util.\*;

**public** **class** Main{

**public** **static** **void** main (String[] args) {

        Vector v = **new** Vector();    // It creates a default vector

        v.add(0,1);                   // Adds 1 at the index 0

        v.add(1,"Java");              // Adds "Java" at the index 1

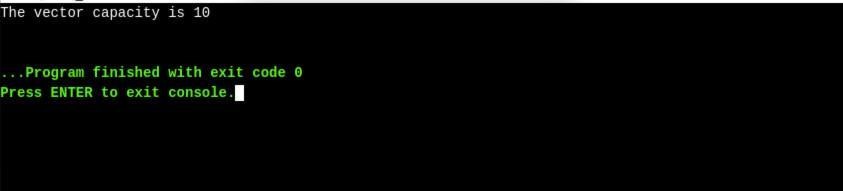
        v.add(2,"is");                // Adds "is" at the index 2

        v.add(3,"Fun");               // Adds "Fun" at the index 3

        System.out.println("The vector capacity  is " + v.capacity());

    }

}



// Java code showing get() method

**import** java.util.\*;

**public** **class** Main{

**public** **static** **void** main (String[] args) {

        Vector v = **new** Vector();    // It creates a default vector

        v.add(1);                   // Adds 1 at the end of the list

        v.add("Java");              // Adds "Java" at the end of the list

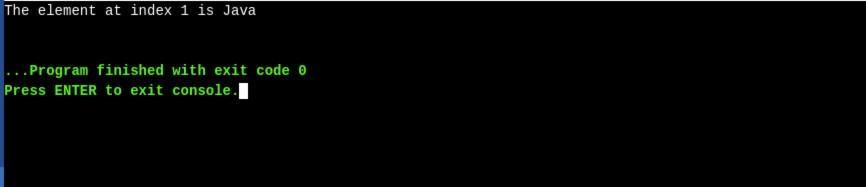
        v.add("is");                // Adds "is" at the end of the list

        v.add("Fun");               // Adds "Fun" at the end of the list

        System.out.println("The element at index 1 is " + v.get(1));

    }

}



// Java code showing firstElement() method

**import** java.util.\*;

**public** **class** Main{

**public** **static** **void** main (String[] args) {

        Vector v = **new** Vector();    // It creates a default vector

        v.add(1);                   // Adds 1 at the end of the list

        v.add("Java");              // Adds "Java" at the end of the list

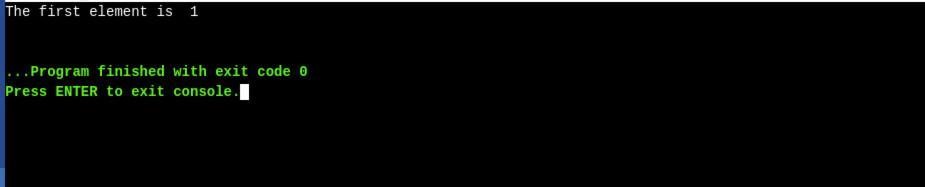
        v.add("is");                // Adds "is" at the end of the list

        v.add("Fun");               // Adds "Fun" at the end of the list

        System.out.println("The first element is  " + v.firstElement());

    }

}



// Java code showing boolean equals() method

**import** java.util.\*;

**public** **class** Main{

**public** **static** **void** main (String[] args) {

        Vector v = **new** Vector();    // It creates a default vector

        Vector vcopy = **new** Vector();

        v.add(1);                   // Adds 1 at the end of the list

        v.add("Java");              // Adds "Java" at the end of the list

        v.add("is");                // Adds "is" at the end of the list

        v.add("Fun");               //Adds "Fun" at the end of the list

        vcopy.add(0,1);             // Adds 1 at the index 0

        vcopy.add(1,"Java");        // Adds "Java" at the index 1

        vcopy.add(2,"is");          // Adds "is" at the index 2

        vcopy.add(3,"Fun");         // Adds "Fun" at the index 3

             vcopy.add(4,"!!!");         // Adds "Fun" at the index 4

**if**(v.equals(vcopy))

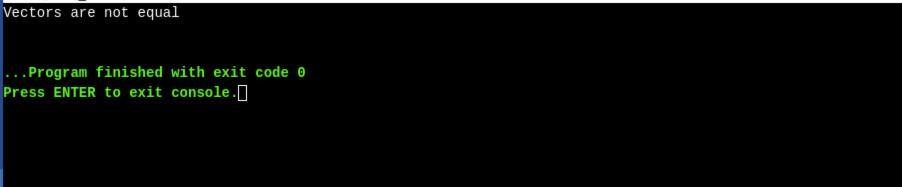
            System.out.println("Both vectors are equal" );

**else**

            System.out.println("Vectors are not equal" );

    }

}



// Java code showing trimToSize() method

**import** java.util.\*;

**public** **class** Main{

**public** **static** **void** main (String[] args) {

        Vector v = **new** Vector();    // It creates a default vector

        v.add(0,1);                   // Adds 1 at the index 0

        v.add(1,"Java");              // Adds "Java" at the index 1

        v.add(2,"is");                // Adds "is" at the index 2

        v.add(3,"Fun");               // Adds "Fun" at the index 3

        System.out.println("The vector capacity is " + v.capacity());

        v.trimToSize();

        System.out.println("The vector capacity is " + v.capacity());

    }

}



**// Java Program to Demonstrate Working of Vector**

**// Via Creating and Using It**

**// Importing required classes**

**import java.io.\*;**

**import java.util.\*;**

**// Main class**

**class GFG {**

**// Main driver method**

**public static void main(String[] args)**

**{**

**// Size of the Vector**

**int n = 5;**

**// Declaring the Vector with**

**// initial size n**

**Vector<Integer> v = new Vector<Integer>(n);**

**// Appending new elements at**

**// the end of the vector**

**for (int i = 1; i <= n; i++)**

**v.add(i);**

**// Printing elements**

**System.out.println(v);**

**// Remove element at index 3**

**v.remove(3);**

**// Displaying the vector**

**// after deletion**

**System.out.println(v);**

**// iterating over vector elements**

**// using for loop**

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

**// Printing elements one by one**

**System.out.print(v.get(i) + " ");}}**

**See the links for practice**

[**https://abhiandroid.com/java/vector-methods-example.html#gsc.tab=0**](https://abhiandroid.com/java/vector-methods-example.html%23gsc.tab=0)

<https://www.javatpoint.com/java-vector>

[**https://www.scaler.com/topics/java/vector-in-java/**](https://www.scaler.com/topics/java/vector-in-java/)

**What is ArrayList in Java?**

ArrayList is a Java class implemented using the List interface. Java ArrayList, as the name suggests, provides the functionality of a dynamic array where the size is not fixed as an array. Also, as a part of the Collection framework, it has many features not available with arrays.

// Java program to demonstrate the

// working of ArrayList

import java.io.\*;

import java.util.\*;

class ArrayListExample {

public static void main(String[] args)

{

// Size of the

// ArrayList

int n = 5;

// Declaring the ArrayList with

// initial size n

ArrayList<Integer> arr1 = new ArrayList<Integer>(n);

// Declaring the ArrayList

ArrayList<Integer> arr2 = new ArrayList<Integer>();

// Printing the ArrayList

System.out.println("Array 1:" + arr1);

System.out.println("Array 2:" + arr2);

// Appending new elements at

// the end of the list

for (int i = 1; i <= n; i++) {

arr1.add(i);

arr2.add(i);

}

// Printing the ArrayList

System.out.println("Array 1:" + arr1);

System.out.println("Array 2:" + arr2);

}

}

**Output**

Array 1:[]

Array 2:[]

Array 1:[1, 2, 3, 4, 5]

Array 2:[1, 2, 3, 4, 5]

|  |  |
| --- | --- |
| **Method** | **Description** |
| [add(int index, Object element)](https://www.geeksforgeeks.org/java-util-arraylist-add-method-java/) | This method is used to insert a specific element at a specific position index in a list. |
| [add(Object o)](https://www.geeksforgeeks.org/java-util-arraylist-add-method-java/) | This method is used to append a specific element to the end of a list. |
| [addAll(Collection C)](https://www.geeksforgeeks.org/java-util-arraylist-addall-method-java/) | This method is used to append all the elements from a specific collection to the end of the mentioned list, in such an order that the values are returned by the specified collection’s iterator. |
| [addAll(int index, Collection C)](https://www.geeksforgeeks.org/java-util-arraylist-addall-method-java/) | Used to insert all of the elements starting at the specified position from a specific collection into the mentioned list. |
| [clear()](https://www.geeksforgeeks.org/arraylist-clear-java-examples/) | This method is used to remove all the elements from any list. |
| [clone()](https://www.geeksforgeeks.org/clone-method-in-java-2/) | This method is used to return a shallow copy of an ArrayList in Java. |
| [contains? (Object o)](https://www.geeksforgeeks.org/arraylist-contains-java/) | Returns true if this list contains the specified element. |
| [ensureCapacity?(int minCapacity)](https://www.geeksforgeeks.org/arraylist-ensurecapacity-method-in-java-with-examples/) | Increases the capacity of this ArrayList instance, if necessary, to ensure that it can hold at least the number of elements specified by the minimum capacity argument. |
| [forEach?(Consumer<? super E> action)](https://www.geeksforgeeks.org/arraylist-foreach-method-in-java/) | Performs the given action for each element of the Iterable until all elements have been processed or the action throws an exception. |
| [get?(int index)](https://www.geeksforgeeks.org/arraylist-get-method-java-examples/) | Returns the element at the specified position in this list. |
| [indexOf(Object O)](https://www.geeksforgeeks.org/java-util-arraylist-indexof-java/) | The index the first occurrence of a specific element is either returned or -1 in case the element is not in the list. |
| [isEmpty?()](https://www.geeksforgeeks.org/arraylist-isempty-java-example/) | Returns true if this list contains no elements. |
| [lastIndexOf(Object O)](https://www.geeksforgeeks.org/arraylist-lastindexof-java-example/) | The index of the last occurrence of a specific element is either returned or -1 in case the element is not in the list. |
| [listIterator?()](https://www.geeksforgeeks.org/arraylist-listiterator-method-in-java-with-examples/) | Returns a list iterator over the elements in this list (in proper sequence). |
| [listIterator?(int index)](https://www.geeksforgeeks.org/arraylist-listiterator-method-in-java-with-examples/) | Returns a list iterator over the elements in this list (in proper sequence), starting at the specified position in the list. |
| [remove?(int index)](https://www.geeksforgeeks.org/arraylist-linkedlist-remove-methods-java-examples/) | Removes the element at the specified position in this list. |
| [remove? (Object o)](https://www.geeksforgeeks.org/arraylist-linkedlist-remove-methods-java-examples/) | Removes the first occurrence of the specified element from this list, if it is present. |
| [removeAll?(Collection c)](https://www.geeksforgeeks.org/arraylist-removeall-method-in-java-with-examples/) | Removes from this list all of its elements that are contained in the specified collection. |
| [removeIf?(Predicate filter)](https://www.geeksforgeeks.org/arraylist-removeif-method-in-java/) | Removes all of the elements of this collection that satisfy the given predicate. |
| [removeRange?(int fromIndex, int toIndex)](https://www.geeksforgeeks.org/arraylist-removerange-java-examples/) | Removes from this list all of the elements whose index is between fromIndex, inclusive, and toIndex, exclusive. |
| [retainAll?(Collection<?> c)](https://www.geeksforgeeks.org/arraylist-retainall-method-in-java/) | Retains only the elements in this list that are contained in the specified collection. |
| [set?(int index, E element)](https://www.geeksforgeeks.org/arraylist-set-method-in-java-with-examples/) | Replaces the element at the specified position in this list with the specified element. |
| [size?()](https://www.geeksforgeeks.org/arraylist-size-method-in-java-with-examples/) | Returns the number of elements in this list. |
| [spliterator?()](https://www.geeksforgeeks.org/arraylist-spliterator-method-in-java/) | Creates a late-binding and fail-fast Spliterator over the elements in this list. |
| [subList?(int fromIndex, int toIndex)](https://www.geeksforgeeks.org/arraylist-sublist-method-in-java-with-examples/) | Returns a view of the portion of this list between the specified fromIndex, inclusive, and toIndex, exclusive. |
| [toArray()](https://www.geeksforgeeks.org/arraylist-array-conversion-java-toarray-methods/) | This method is used to return an array containing all of the elements in the list in the correct order. |
| [toArray(Object[] O)](https://www.geeksforgeeks.org/arraylist-array-conversion-java-toarray-methods/) | It is also used to return an array containing all of the elements in this list in the correct order same as the previous method. |
| [trimToSize()](https://www.geeksforgeeks.org/arraylist-trimtosize-java-example/) | This method is used to trim the capacity of the instance of the ArrayList to the list’s current size. |

// Java Program to Add elements to An ArrayList

// Importing all utility classes

import java.util.\*;

// Main class

class GFG {

// Main driver method

public static void main(String args[])

{

// Creating an Array of string type

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

// Adding elements to ArrayList

// Custom inputs

al.add("Geeks");

al.add("Geeks");

// Here we are mentioning the index

// at which it is to be added

al.add(1, "For");

// Printing all the elements in an ArrayList

System.out.println(al);

}

}

[Geeks, For, Geeks]

// Java Program to Change elements in ArrayList

// Importing all utility classes

import java.util.\*;

// main class

class GFG {

// Main driver method

public static void main(String args[])

{

// Creating an Arraylist object of string type

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

// Adding elements to Arraylist

// Custom input elements

al.add("Geeks");

al.add("Geeks");

// Adding specifying the index to be added

al.add(1, "Geeks");

// Printing the Arraylist elements

System.out.println("Initial ArrayList " + al);

// Setting element at 1st index

al.set(1, "For");

// Printing the updated Arraylist

System.out.println("Updated ArrayList " + al);

}

}

**Output**

Initial ArrayList [Geeks, Geeks, Geeks]

Updated ArrayList [Geeks, For, Geeks]

// Java program to Remove Elements in ArrayList

// Importing all utility classes

import java.util.\*;

// Main class

class GFG {

// Main driver method

public static void main(String args[])

{

// Creating an object of arraylist class

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

// Adding elements to ArrayList

// Custom addition

al.add("Geeks");

al.add("Geeks");

// Adding element at specific index

al.add(1, "For");

// Printing all elements of ArrayList

System.out.println("Initial ArrayList " + al);

// Removing element from above ArrayList

al.remove(1);

// Printing the updated Arraylist elements

System.out.println("After the Index Removal " + al);

// Removing this word element in ArrayList

al.remove("Geeks");

// Now printing updated ArrayList

System.out.println("After the Object Removal "

+ al);

}

}

**Output**

Initial ArrayList [Geeks, For, Geeks]

After the Index Removal [Geeks, Geeks]

After the Object Removal [Geeks]

// Java program to Iterate the elements

// in an ArrayList

// Importing all utility classes

import java.util.\*;

// Main class

class GFG {

// Main driver method

public static void main(String args[])

{

// Creating an Arraylist of string type

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

// Adding elements to ArrayList

// using standard add() method

al.add("Geeks");

al.add("Geeks");

al.add(1, "For");

// Using the Get method and the

// for loop

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

System.out.print(al.get(i) + " ");

}

System.out.println();

// Using the for each loop

for (String str : al)

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

}

}

**Output**

Geeks For Geeks

Geeks For Geeks

// Java program to get the elemens in ArrayList

import java.io.\*;

import java.util.\*;

class GFG {

public static void main (String[] args) {

ArrayList<Integer> list = new ArrayList();

// add the number

list.add(9);

list.add(5);

list.add(6);

System.out.println(list);

// get method

Integer n= list.get(1);

System.out.println("at indext 1 number is:"+n);

}

}

**Output**

[1, 2, 4]

[1, 2, 3, 4]

// Java Program for ArrayList Sorting

import java.io.\*;

import java.util.\*;

class GFG {

public static void main(String[] args)

{

ArrayList<Integer> list = new ArrayList();

list.add(2);

list.add(4);

list.add(3);

list.add(1);

System.out.println("Before sorting list:");

System.out.println(list);

Collections.sort(list);

System.out.println("after sorting list:");

System.out.println(list);

}

}

**Output**

Before sorting list:

[2, 4, 3, 1]

after sorting list:

[1, 2, 3, 4]

// Java program to find the size

// of elements of an ArrayList

import java.io.\*;

import java.util.\*;

class GFG {

public static void main(String[] args)

{

ArrayList<Integer> list = new ArrayList();

list.add(1);

list.add(2);

list.add(3);

list.add(4);

int b = list.size();

System.out.println("The size is :" + b);

}

}

**Output**

The size is :4

**LAB#02 EXERCISES**

**QUESTION#1**

Create an ArrayList of integers and find the sum of all elements.

### QUESTION#2

Given two ArrayLists of strings, concatenate them into a new ArrayList and sort the result alphabetically.

### QUESTION#3

Write a program to remove all even numbers from an ArrayList of integers.

### QUESTION#4

Implement a method to rotate the elements of an ArrayList to the left by a specified number of positions.

### QUESTION#5

Given two ArrayLists of objects, write a program to find the common elements between them.

### QUESTION#6

Create a Vector of doubles and calculate the average of all elements.

### QUESTION#7

### Write a program to find the union of two Vectors of characters, i.e., all unique elements from both Vectors.

### QUESTION#8

Implement a method to reverse the order of elements in a Vector.

### QUESTION#9

Design a generic Vector class that can handle any data type. Include methods for adding, removing, and accessing elements.

### QUESTION#10

Write a program to find the largest contiguous subsequence sum in a Vector of integers.