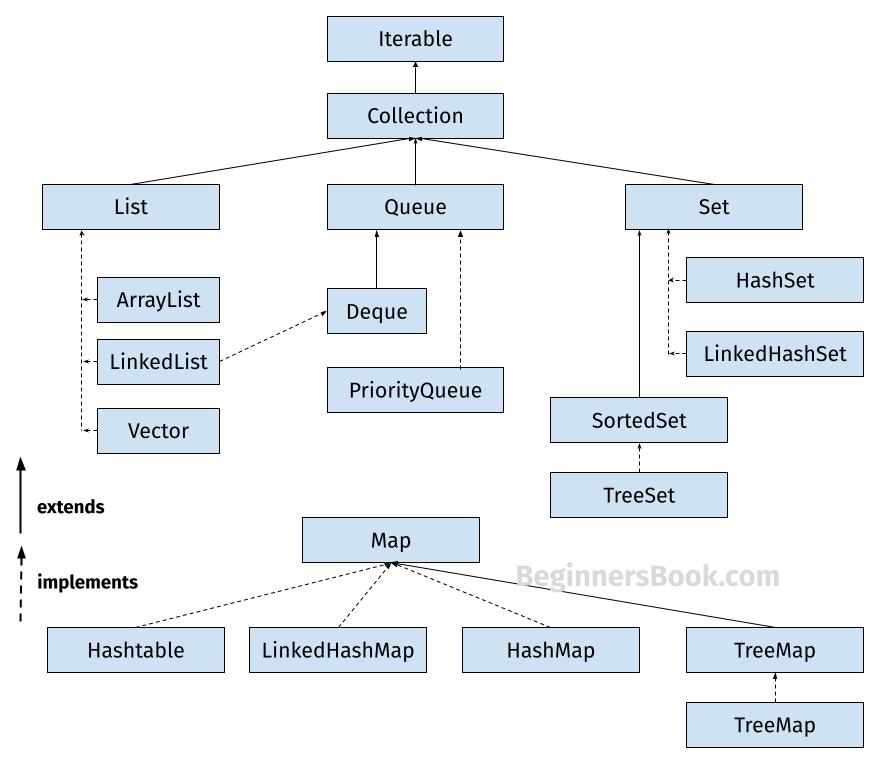
# **Collections in Java**

* The **Java Collections Framework** is a collection of interfaces and classes, which helps in storing and processing the data efficiently. This framework has several useful classes which have tons of useful functions which makes a programmer task super easy.



## **1. List**

A List is an ordered Collection (sometimes called a sequence). Lists may contain duplicate elements. Elements can be inserted or accessed by their position in the list, using a zero-based index. The classes that implements List interface are:

* ArrayList
* LinkedList
* Vector
* Stack

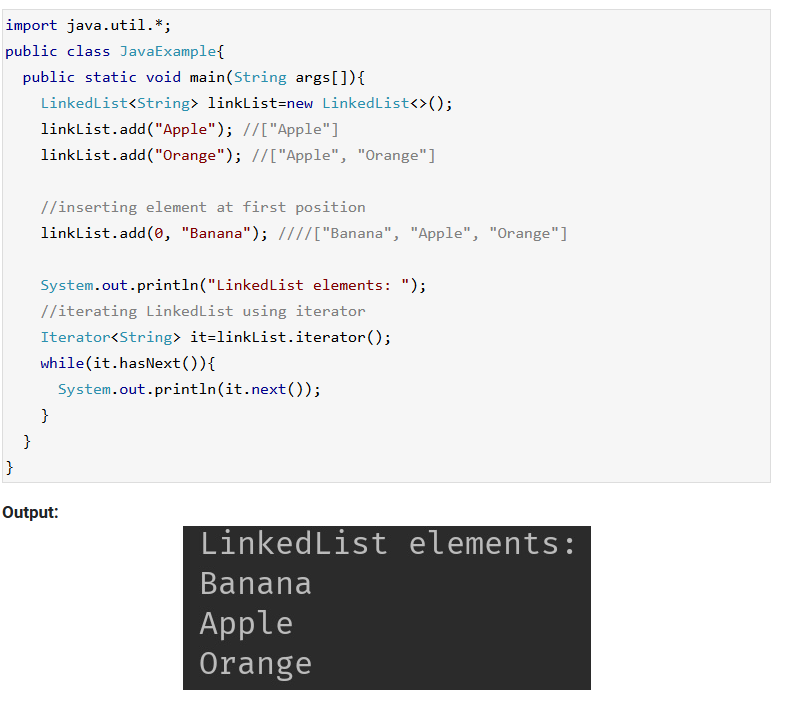
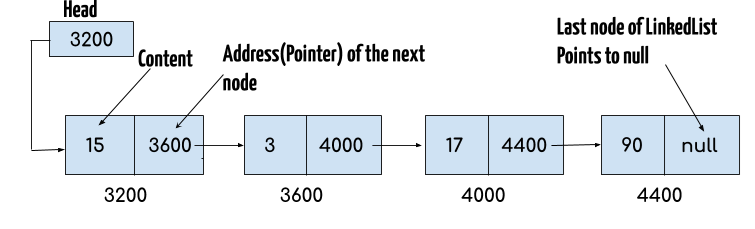
### **1.1 ArrayList**

ArrayList is a popular alternative of [**arrays in Java**](https://beginnersbook.com/2013/05/java-arrays/). It is based on an Array **data structure**. ArrayList is a resizable- array implementation of the List interface. It implements all optional list operations, and permits all elements, including null.



### **1.2 LinkedList**

LinkedList is a linear data structure. However LinkedList elements are not stored in contiguous locations like arrays, they are linked with each other using pointers. Each element of the LinkedList has the reference(address/pointer) to the next element of the LinkedList.



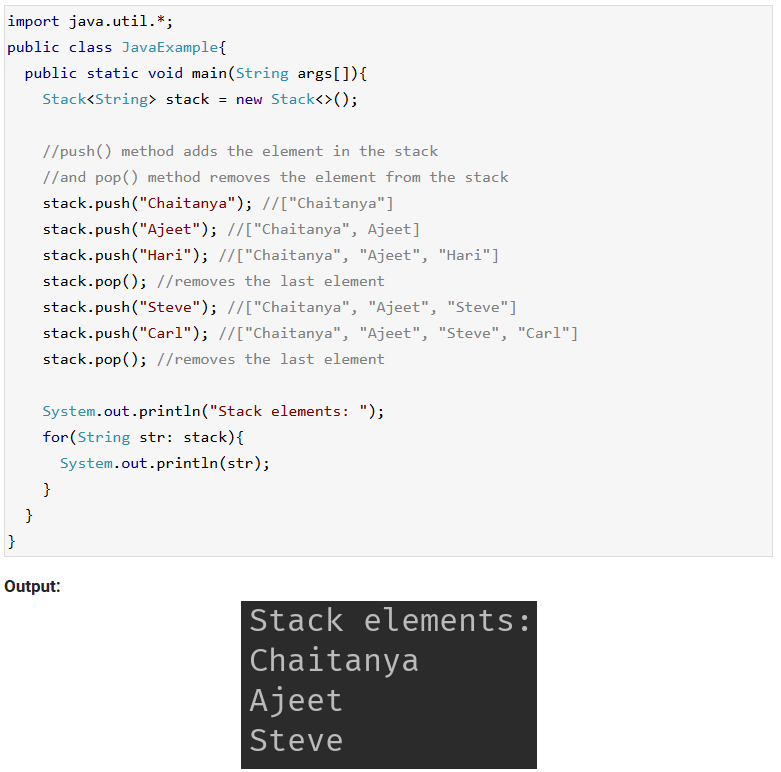
### **1.3 Vector**

Here is the list of all the tutorials published on the Vector.



### **1.4 Stack**

Stack class extends Vector class, which means it is a subclass of Vector. Stack works on the concept of Last In First Out (LIFO). The elements are inserted using push() method at the end of the stack, the pop() method removes the element which was inserted last in the Stack.



## **2. Set**

A Set is a Collection that cannot contain duplicate elements. There are three main implementations of Set interface: HashSet, TreeSet, and LinkedHashSet.

### **2.1 HashSet**

[**HashSet**](https://beginnersbook.com/2013/12/hashset-class-in-java-with-example/) which stores its elements in a hash table, is the best-performing implementation. HashSet allows only unique elements. It doesn’t maintain the insertion order which means element inserted last can appear at first when traversing the HashSet.



### **2.2 LinkedHashSet**

Unlike HashSet, the [**LinkedHashSet**](https://beginnersbook.com/2013/12/linkedhashset-class-in-java-with-example/) maintains insertion order.



### **2.3 TreeSet**

[**TreeSet**](https://beginnersbook.com/2013/12/treeset-class-in-java-with-example/) stores elements in a red-black tree. It is substantially slower than HashSet. TreeSet class implements SortedSet interface, which allows TreeSet to order its elements based on their values, which means TreeSet elements are sorted in ascending order.



## **3. Map**

A Map is an object that maps keys to values. A map cannot contain duplicate keys. There are three main implementations of Map interfaces: HashMap, TreeMap, and LinkedHashMap.But can have duplicate values for different keys.

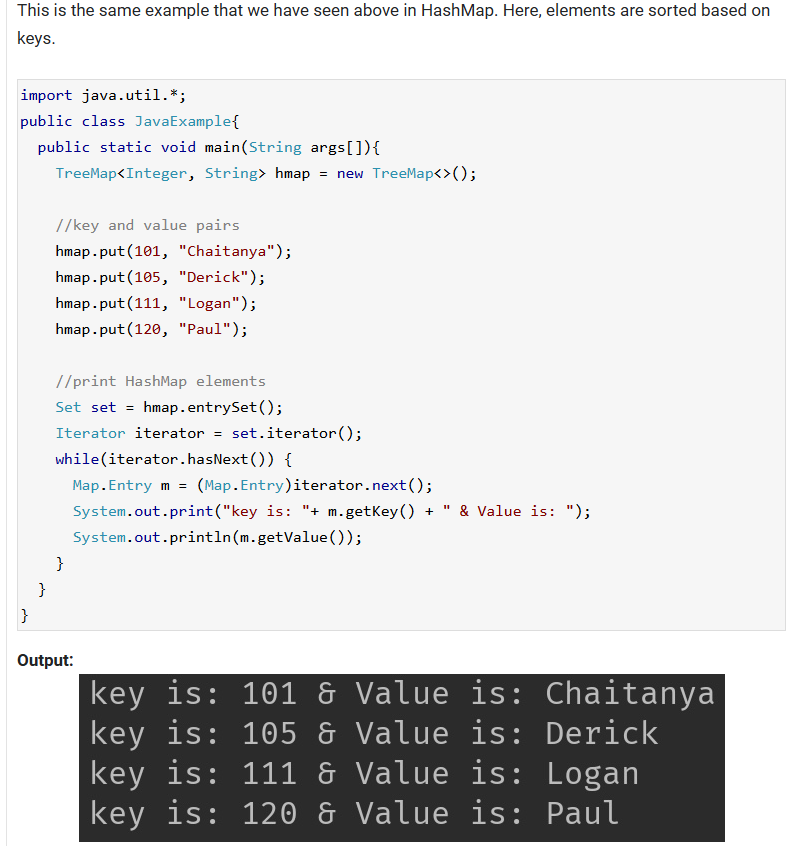
### **3.1 HashMap**

HashMap is like HashSet, it doesn’t maintain insertion order and doesn’t sort the elements in any order.



### **3.2 TreeMap**

 It stores its elements in a red-black tree. The elements of TreeMap are sorted in ascending order. It is substantially slower than HashMap



### **3.3 LinkedHashMap**

 It maintains insertion order. As you can see: In the following example, the key & value pairs maintained the insertion order.

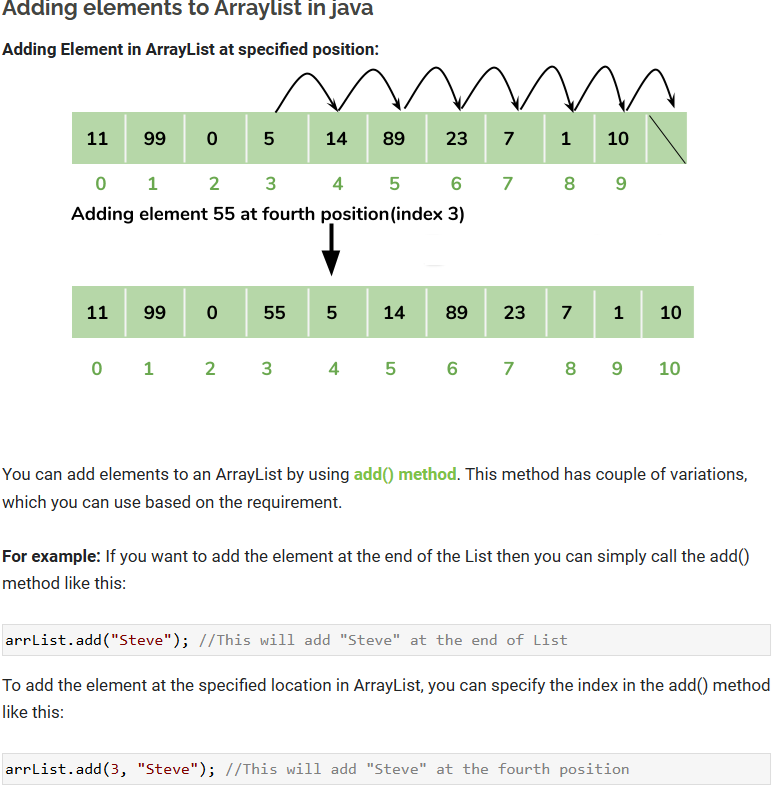
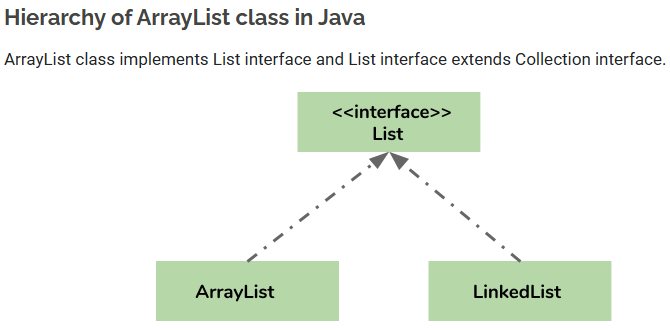


# **ArrayList in Java**

* **Arraylist** class implements List interface and it is based on an Array data structure. It is widely used because of the functionality and flexibility it offers. **ArrayList in Java**, is a resizable-array implementation of the List interface. It implements all optional list operations and permits all elements, including null.
* ArrayList can grow and shrink automatically based on the addition and removal of elements.
* ArrayList can contain duplicate elements
* ArrayList maintains the insertion order, which means the elements appear in the same order in which they are inserted.
* ArrayList is non synchronized.

## **Array vs arraylist in java**

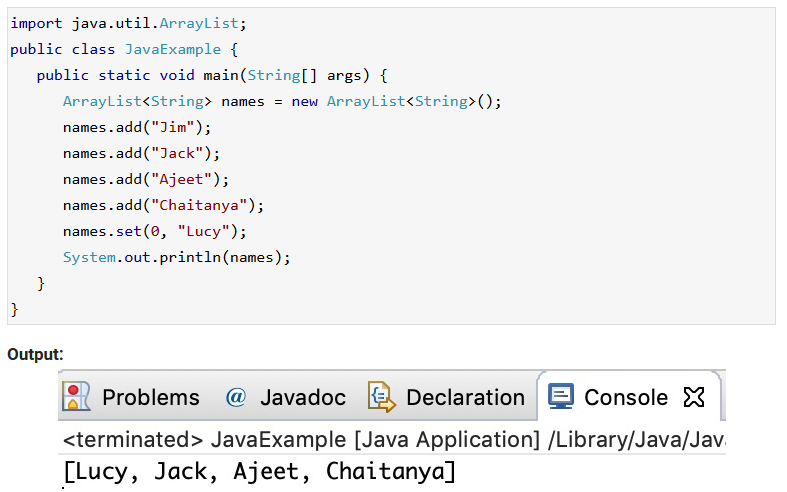
* The main **difference between array and arraylist** is that arraylist can grow and shrink dynamically while an array cannot.
* An array has a **fixed length** so if it is full you cannot add any more elements to it. Similarly, if number of elements are removed from ArrayList, the memory consumption remains same as it doesn’t shrink.
* On the other hand, **ArrayList can dynamically grow and shrink** after addition and removal of elements. ArrayList class has several useful methods that can make our task easy.



### **Change an element in ArrayList**

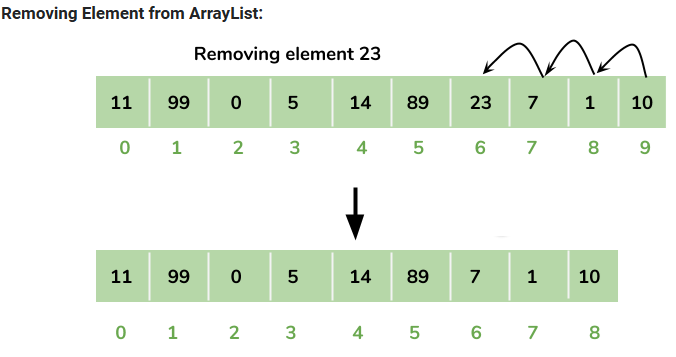
You can use the **set method** to change an element in ArrayList. You need to provide the **index** and **new element**, this method then updates the element present at the **given index** with the **new given element**.

In the following example, we have given the index as 0 and new element as “Lucy” in the set() method. The method updated the element present at the index 0 (“Jim”) with the new String element “Lucy”.



### **How to remove element from Arraylist in Java?**

You can use **remove() method** to remove elements from an ArrayList. Similar to add() method, this method also has couple of variations.



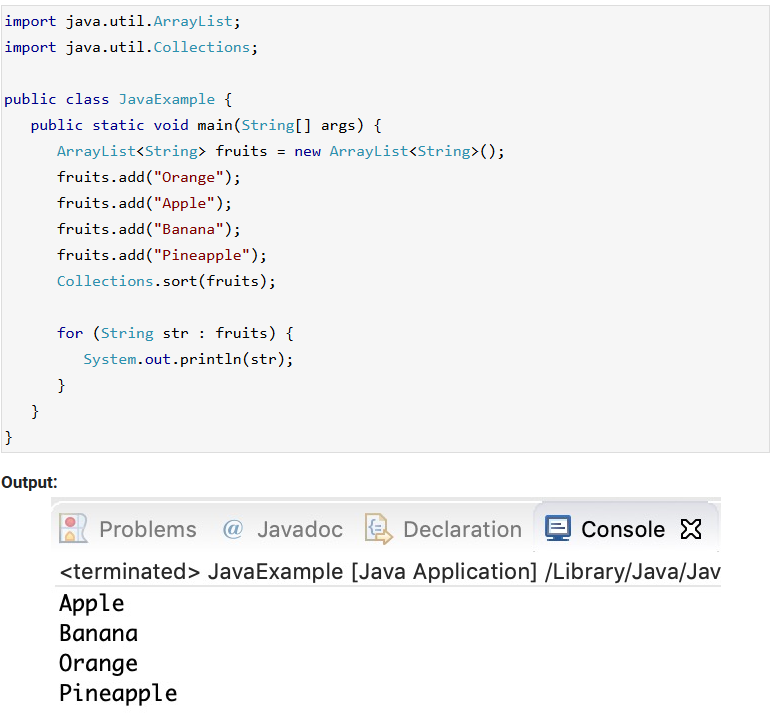
### **Iterating ArrayList**

Here, we are using enhanced for loop to iterate ArrayList elements. This one of the best ways to iterate an ArrayList of string type.



### **Sort ArrayList**

You can use the **sort() method** of the **Collections utility class**to [**sort an ArrayList**](https://beginnersbook.com/2013/12/how-to-sort-arraylist-in-java/). This class is a part of **java.util** package. In the following example we are sorting a list of String type **alphabetically.** This method also works on **numeric lists** (such as Integer type ArrayList).



## **All methods of Arraylist in Java**

In the above examples, we have used methods such as add() and remove(). However there are number of other useful methods available in ArrayList class.

1) [**add**](https://beginnersbook.com/2013/12/java-arraylist-add-method-example/)**( Object o)**: This method adds an object o at the end of the arraylist.

obj.add("hello");

This statement would add a string hello in the arraylist at last position.

2) [**add**](https://beginnersbook.com/2013/12/java-arraylist-addint-index-e-element-example/)**(int index, Object o)**: It adds the object o at the specified index in the ArrayList.

obj.add(2, "bye");

It will add the string “bye” at the 2nd index (third element as array list starts with index 0) of array list.

3) [**remove**](https://beginnersbook.com/2013/12/java-arraylist-removeobject-method-example/)**(Object o)**: Removes the object o from the ArrayList.

obj.remove("Chaitanya");

This statement will remove the string “Chaitanya” from the ArrayList.

4) [**remove**](https://beginnersbook.com/2013/12/java-arraylist-remove-method-example/)**(int index)**: Removes element from a given index.

obj.remove(3);

It would remove the element of index 3 (4th element of the list – List starts with o).

5) [**set**](https://beginnersbook.com/2013/12/java-arraylist-set-method-example/)**(int index, Object o)**: Used for updating an element. It replaces the element present at the specified index with the object o.

obj.set(2, "Tom");

It would replace the 3rd element (index =2 is 3rd element) with the value Tom.

6)**int indexOf(Object o)**: Gives the index of the object o. If the element is not found in the list then this method returns the value -1.

int pos = obj.indexOf("Tom");

This would give the index (position) of the string Tom in the list.

7) **Object**[**get**](https://beginnersbook.com/2013/12/java-arraylist-get-method-example/)**(int index)**: It returns the object of list which is present at the specified index.

String str= obj.get(2);

This would return the string stored at 3rd position (index 2) and would be assigned to the string “str”. We are using string variable to store the get() result because the list is of string type. If the list is of int type then we can use int variable to store the returned element.

8) [**int size()**](https://beginnersbook.com/2013/12/how-to-find-length-of-arraylist-in-java/): It returns the size of the ArrayList (Number of elements of the list).

int numberofitems = obj.size();

9) **boolean**[**contains**](https://beginnersbook.com/2013/12/java-arraylist-contains-method-example/)**(Object o)**: It checks whether the given object o is present in the array list. If the element is found it returns true else it returns false.

obj.contains("Steve");

It would return true if the string “Steve” is present in the list else we would get false.

10) [**clear**](https://beginnersbook.com/2013/12/how-to-empty-an-arraylist-in-java/)**():** It is used for removing all the elements of the array list in one go. The below code will remove all the elements of ArrayList whose object is obj.

obj.clear();