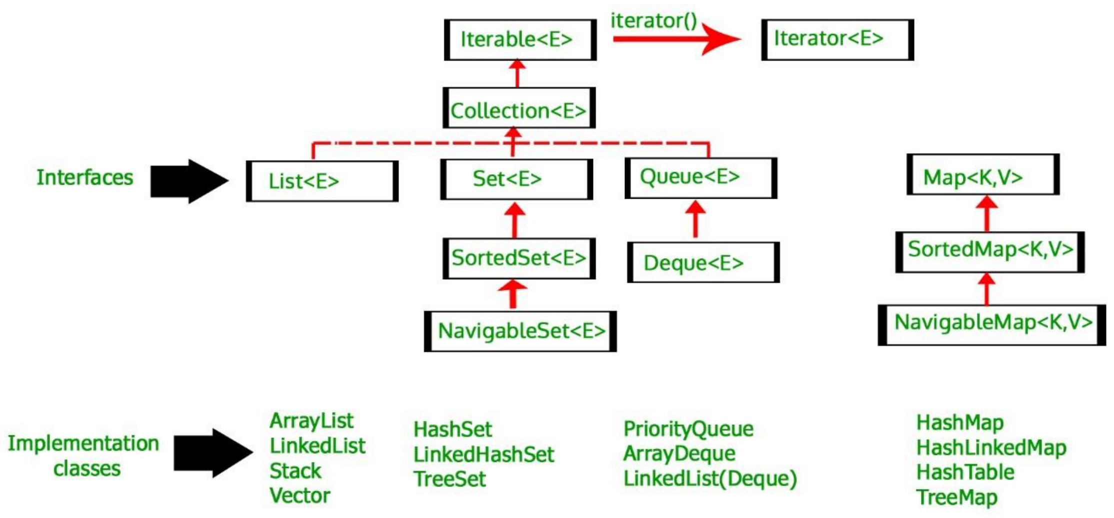
ArrayList in Java

ArrayList is a part of collection framework and is present in java.util package. It provides us dynamic arrays in Java. Though, it may be slower than standard arrays but can be helpful in programs where lots of manipulation in the array is needed.

- ArrayList inherits AbstractList class and implements List interface.
- ArrayList is initialized by a size, however the size can increase if collection grows or shrunk if objects are removed from the collection.
- Java ArrayList allows us to randomly access the list.
- ArrayList can not be used for primitive types, like int, char, etc. We need a wrapper class for such cases (see this for details).
- ArrayList in Java can be seen as similar to vector in C++.



Now primarily the Java ArrayList can constitute of both Constructors and Methods. Below mentioned is a list of few constructors and methods along with there use and functions.

Constructors in Java ArrayList:

1. ArrayList(): This constructor is used to build an empty array list
2. ArrayList(Collection c): This constructor is used to build an array list initialized with the elements from collection c
3. ArrayList(int capacity): This constructor is used to build an array list with initial capacity being specified

Let us look at the code to create generic ArrayList-

```
// Creating generic integer ArrayList

ArrayList<Integer> arrli = new ArrayList<Integer>();
```

```
// Java program to demonstrate working of ArrayList in Java
import java.io.*;
import java.util.*;

class arrayList {
    public static void main(String[] args)throws IOException {
        // size of ArrayList
        int n = 5;

        //declaring ArrayList with initial size n
        ArrayList<Integer> arrli = new ArrayList<Integer>(n);

        // Appending the new element at the end of the list
        for (int i=1; i<=n; i++)
            arrli.add(i);

        // Printing elements
        System.out.println(arrli);

        // Remove element at index 3
        arrli.remove(3);

        // Displaying ArrayList after deletion
        System.out.println(arrli);

        // Printing elements one by one
        for (int i=0; i<arrli.size(); i++)
            System.out.print(arrli.get(i)+" ");
    }
}
```

Output:

```
[1, 2, 3, 4, 5]

[1, 2, 3, 5]

1 2 3 5
```

Methods in Java ArrayList:

1. forEach(Consumer<? super E> action): Performs the given action for each element of the Iterable until all elements have been processed or the action throws an exception.
2. retainAll(Collection<?> c): Retains only the elements in this list that are contained in the specified collection.
3. removeIf(Predicate<? super E> filter): Removes all of the elements of this collection that satisfy the given predicate.
4. contains(Object o): Returns true if this list contains the specified element.
5. remove(int index): Removes the element at the specified position in this list.
6. remove(Object o): Removes the first occurrence of the specified element from this list, if it is present.
7. get(int index): Returns the element at the specified position in this list.
8. subList(int fromIndex, int toIndex): Returns a view of the portion of this list between the specified fromIndex, inclusive, and toIndex, exclusive.
9. spliterator(): Creates a late-binding and fail-fast Spliterator over the elements in this list.

10. set(int index, E element): Replaces the element at the specified position in this list with the specified element.
11. size(): Returns the number of elements in this list.
12. removeAll(Collection<?> c): Removes from this list all of its elements that are contained in the specified collection.
13. ensureCapacity(int minCapacity): 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.
14. listIterator(): Returns a list iterator over the elements in this list (in proper sequence).
15. listIterator(int index): Returns a list iterator over the elements in this list (in proper sequence), starting at the specified position in the list.
16. isEmpty(): Returns true if this list contains no elements.
17. removeRange(int fromIndex, int toIndex): Removes from this list all of the elements whose index is between fromIndex, inclusive, and toIndex, exclusive.
18. void clear(): This method is used to remove all the elements from any list.
19. void add(int index, Object element): This method is used to insert a specific element at a specific position index in a list.
20. void trimToSize(): This method is used to trim the capacity of the instance of the ArrayLis to the list's current size.
21. int indexOf(Object O): The
    index the first occurrence of a specific element is either returned, or -1 in case the element is not in the list.
22. int lastIndexOf(Object O): The index the last occurrence of a specific element is either returned, or -1 in case the element is not in the list.
23. Object clone(): This method is used to return a shallow copy of an ArrayList.
24. Object[] toArray(): This method is used to return an array containing all of the elements in the list in correct order.
25. Object[] toArray(Object[] O): 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.
26. boolean addAll(Collection C): This method is used to append all the elements from a specific collection to the end of the mentioned list, in such a order that the values are returned by the specified collection's iterator.
27. boolean add(Object o): This method is used to append a specificd element to the end of a list.
28. boolean addAll(int index, Collection C): Used to insert all of the elements starting at the specified position from a specific collection into the mentioned list.