List

Overview

- **List** is an **interface** which **extends** the **Collection interface**. i.e., List is a child interface of Collection interface.
- Lists **can store null and duplicate** elements.
- **Preserve Insertion order**.

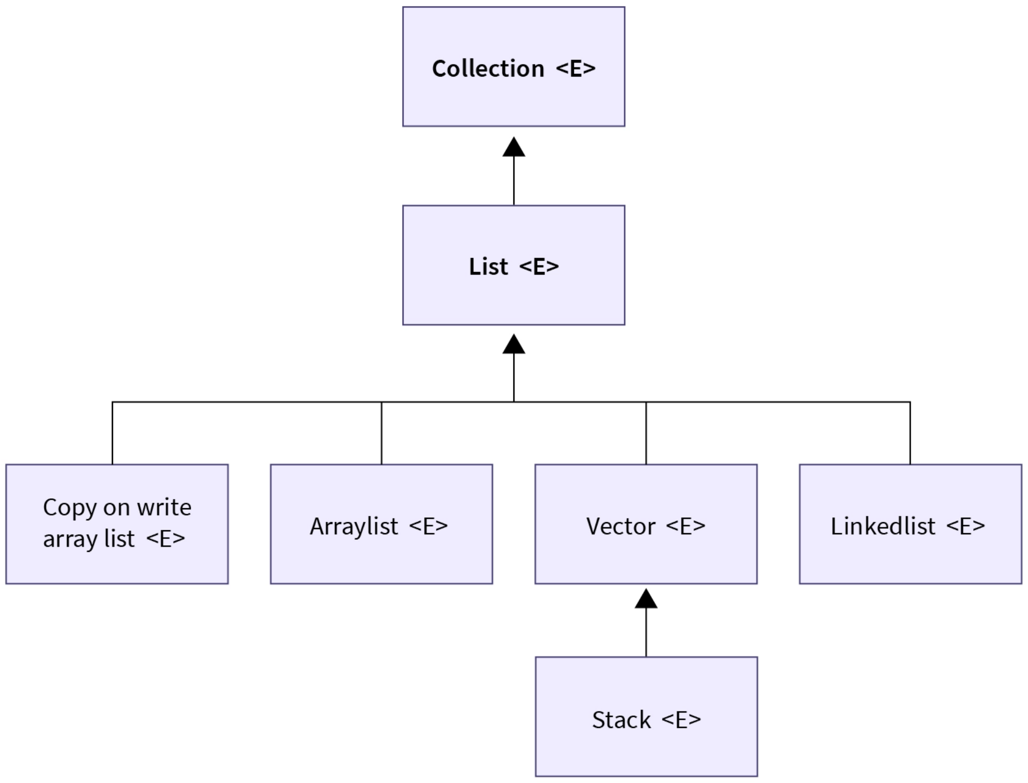
Which classes implement the List interface?

- ArrayList
- LinkedList
- Vector
- Stack
- CopyOnWriteArrayList
- AbstractList

Why List()?

- Main **disadvantage of arrays is fixed length.**
- **List allows dynamic insertion** of elements as well as **elements stored in order**.
- Also **classes** which implements list interface, **provides inbuilt methods** which improves the efficiency of the programmer.

Hierarchy of List



In the above image, we can see all classes which implement List interface. These classes use abstract methods of the list Interface to design the structure of its class.

How to create a List?

1. Using List Interface
2. Without Using List Interface

1. Using List Interface

   a. All the classes which implement List Interface, implements abstract methods of list interface.

    b. We can use them to access the implementation of these methods for creating a list in java
    c. e.g.,
```
        List list = new ArrayList<>();
```
    d. List can not be used for storing the primitive data types elements like int, float, etc.

2. Without using the List interface
    a. In this method, we use classes which implement List interface.
    b. Classes may have the same or different implementation of the List's abstract methods.
    c. e.g.,
```
ArrayList<String> al = new ArrayList<String>();
LinkedList<Integer>list2 = new LinkedList<Integer>();
```

Example of Java List

1. Using The asList() Method
2. Using List.add()
3. Using Collection Methods
4. Using Java 9 List.of() Method

1. **Using The asList() Method** (method belongs to Arrays class)

   - Used to convert an array into a list(ArrayList, LinkedList, Vector, Stack).
   - The type of an array to be converted into a list must be Wrapper class.
   - eg.,
```
        //Integer array
         Integer arr[] = {1, 2, 3, 4, 5};

         //Converting the Integer arr to list
         List<Integer> lst = Arrays.asList(arr);
```

2. **Using List.add()**
   - used for adding the user-defined or wrapper class objects to the existing list.
   - e.g.,
```
        //Creating the list
        List<Integer>list = new ArrayList<Integer>();
        list.add(1);
        list.add(2);

        //Printing the List
        System.out.println(list);
```

3. Using Collection method
   - Collection interface is a parent interface of list interface.
   - methods available in the Collection class is implemented by list classes so it can be used for creating a list.
   - eg.,

```
Collection<Integer> c = new ArrayList<Integer>();
c.add(1);
c.add(2);
System.out.println(c);
```

4. Using Java 9 **List.of(T…a)** method
   - The List.of() **takes varargs as an argumen**t
   - **Returns the immutable list**.
   - eg.,

```
List<String> lStr = List.of("ABC","XYZ");
System.out.println(lStr);
```

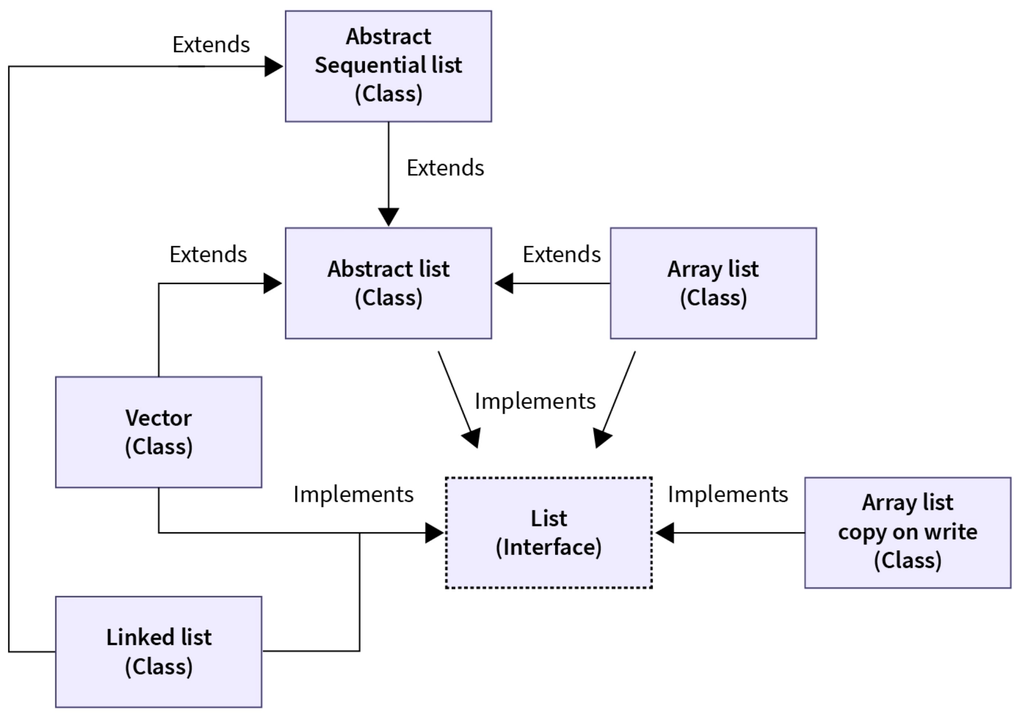
Methods in List Interface

Sr. No.	**Method**	**Description**
1	**void add(int index,E)**	This method of List interface is used for **inserting an element at the particular index** of the list
2	**boolean add(E e)**	This method is used for **appending the element at the end** of the list and returns true if the element is successfully added.
3	**void clear()**	This method is used to **remove all the elements** from the list.
4	**boolean equals(Object o)**	This method is **used for the comparison** of two objects.
5	**int hashcode**	This method is **used to get the hash value** of an object.
6	**E get(int index_value)**	This method is used to **fetch an element** that is present

		at the given index value in the list.
7	**boolean isEmpty()**	This method is used to determine whether the **list is empty or not**, if empty this method returns true, otherwise false.
8	**int lastIndexOf(E e)**	This method is used to **get the last index** value of the given **object**. If the object is not present in the list, this method returns -1.
9	**boolean contains(Object E)**	This method is used to check whether the list **contains** specialized objects in the list or not, if present this method returns true, otherwise returns false.
10	**int indexOf(Object E)**	This method is used to **fetch the first index position of the given object** in the list, if the object is not present, This method returns -1.
11	**List remove(int index)**	This method is used for **removing the element** or object present at the **given index position**. An exception will occur if we provide an invalid index value.
12	**boolean remove(Object E)**	This method is used to **remove an object** from the list and returns a boolean value.
13	**List set(int index, Object)**	This method is used to **set an object at the given index** position in the list.
14	**int size()**	This method is used to **determine the size** of the List. The return type of this method is int.
15	**Spliterator<E> spliterator()**	This method is used to **create a spliterator** over the elements in a list.
16	**boolean addAll(Collection<? extends E> c)**	This method is used for **appending** the particular **collection at the end** of the existing list.

17	**List.toArray()**	This method is used to **return the array containing** all elements of the list in an ordered way

Classes Association with a List Interface



Abstract Sequential list (Class)
Extends
Extends
Extends
Abstract list (Class)
Extends
Array list (Class)
Implements
Vector (Class)
Implements
List (Interface)
Implements
Array list copy on write (Class)
Linked list (Class)