Java Collections

The Java collections perform all the data structure work. Just like an array, a collection can also be used to store objects with an advantage over an array. In a collection, the stored object can grow or shrink dynamically, however, in an array, there is a size limit.

Collection framework

A collection framework is a unified architecture or a set of classes and interfaces for representing and manipulating collections. i.e. collection framework is used to store, retrieve and manipulate collections. Collection framework contains the following:

1. **Interfaces** are abstract data types that represent collections and allow collections to be manipulated independently of the details of their representation.

2. **Classes/Implementations** are the concrete implementations of the collection interfaces.

3. **Algorithms:** are the methods used for collection computations, like searching and sorting.

We can say Java collections has in 4 basic flavors as below:

- Lists
- Sets
- Maps
- Queues

List interface

The List interface extends Collection to define an ordered collection with duplicates allowed. The List interface adds position-oriented operations, as well as a new list iterator that enables the user to traverse the list bi-directionally. ArrayList, LinkedList and vector are classes implementing List interface.

Example

```
import java.util.ArrayList;

class Program
{
   public static void main(String[] args)
   {
      // Create a new List object of integers
      ArrayList<Integer> list = new ArrayList<Integer>();
      // add elements to the List object one by one
      list.add(10);
      list.add(20);
      list.add(30);
      list.add(40);
      list.add(50);
      list.add(60);
      // Display the elements in the List object
      System.out.println("Elements in the list:");

      for (int i: list)
      {
         System.out.println(i);
      }
   }
}
```

Example

```
import java.util.ArrayList;

class Program
{
   public static void main(String[] args)
```

```
    {
        // Creating a List object of Strings
        ArrayList<String> Students = new ArrayList<String>();

        // adding elements to the List object one by one
        Students.add("John");
        Students.add("Mary");
        Students.add("Anderson");
        Students.add("Lucy");
        Students.add("Jenson");
        Students.add("Emily");
        // Display the elements in the List object
        System.out.println("Elements in the Students:");
        for (String s: Students)
        {
            System.out.println(s);
        }
    }
}
```

Example

```
import java.util.ArrayList;

class Program
{
    public static void main(String[] args)
    {
        // Creating a List object of integers
        ArrayList<Integer> list = new ArrayList<Integer>();
        // adding elements to the List object one by one
        list.add(1);
        list.add(2);
        list.add(3);
        list.add(4);
        list.add(5);
        list.add(6);
        // Printing the Count of the List object
        System.out.println("Count: " + list.size());
    }
}
```

Difference Between List and ArrayList

List	ArrayList
List is an interface	ArrayList is a class
List interface extends the Collection framework	ArrayList extends AbstractList class and implements List interface
List cannot be instantiated.	ArrayList can be instantiated.
List interface is used to create a list of elements(objects) that are associated with their index numbers.	ArrayList class is used to create a dynamic array that contains objects.
List interface creates a collection of elements that are stored in a sequence and they are identified and accessed using the index.	ArrayList creates an array of objects where the array can grow dynamically.

Example

```
import java.util.ArrayList;

class Program
{
   public static void main(String[] args)
   {
      // Create a new List object of integers
      ArrayList<Integer> list = new ArrayList<Integer>();
      // add elements to the List object one by one
      list.add(10);
      list.add(20);
      list.add(30);
      list.add(40);
      list.add(50);
      list.add(60);
      // Display the elements in the List object
      System.out.println("Elements in the list: ");

      for (int i: list)
      {
         System.out.println(i + " ");
      }
      System.out.println();

      // Creating an array of integers
      int[] array = { 10, 20, 30, 40, 50, 60 };
      // Display the elements in the Array
      System.out.println("Elements in the array: ");

      for (int j: array)
      {
         System.out.println(j + " ");
      }
   }
}
```