Collections in Java

A Collection is a group of individual objects represented as a single unit. Java provides Collection Framework which defines several classes and interfaces to represent a group of objects as a single unit.

The Collection interface (**java.util.Collection**) and Map interface (**java.util.Map**) are two main root interfaces of Java collection classes.

Need for Collection Framework :

Before Collection Framework (or before JDK 1.2) was introduced, the standard methods for grouping Java objects (or collections) were array or Vector or Hashtable. All three of these collections had no common interface.
For example, if we want to access elements of array, vector or Hashtable. All these three have different methods and syntax for accessing members:

```
// Java program to show whey collection framework was needed
import java.io.*;
import java.util.*;

class Test
{
    public static void main (String[] args)
    {
        // Creating instances of array, vector and hashtable
        int arr[] = new int[] {1, 2, 3, 4};
        Vector<Integer> v = new Vector();
        Hashtable<Integer, String> h = new Hashtable();
        v.addElement(1);
        v.addElement(2);
        h.put(1,"geeks");
        h.put(2,"4geeks");

        // Array instance creation requires [], while Vector
        // and hastable require ()
        // Vector element insertion requires addElement(), but
        // hashtable element insertion requires put()

        // Accessing first element of array, vector and hashtable
        System.out.println(arr[0]);
        System.out.println(v.elementAt(0));
        System.out.println(h.get(1));

        // Array elements are accessed using [], vector elements
```

```
        // using elementAt() and hashtable elements using get()
    }
}
```

Run on IDE

Output:

```
1
1
geek
```

As we can see, none of the collections (Array, Vector or Hashtable) implements a standard member access interface. So, it was very difficult for programmers to write algorithm that can work for all kind of collections.

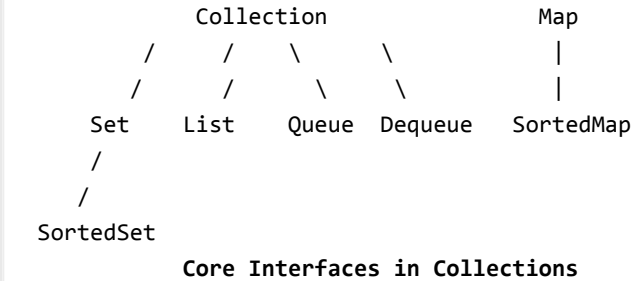
Another drawback is that, most of the 'Vector' methods are final. So, we cannot extend 'Vector' class to implement a similar kind of collection.

Java developers decided to come up with a common interface to deal with the above mentioned problems and introduced Collection Framework, they introduced collections in JDK 1.2 and changed the legacy Vector and Hashtable to conform to the collection framework.

Advantages of Collection Framework:

1. Consistent API : The API has basic set of interfaces like Collection, Set, List, or Map. All those classes (such as ArrayList, LinkedList, Vector etc) which implements, these interfaces have some common set of methods.
2. Reduces programming effort: The programmer need not to worry about design of Collection rather than he can focus on its best use in his program.
3. Increases program speed and quality: Increases performance by providing high-performance implementations of useful data structures and algorithms.

Hierarchy of Collection Framework



Core Interfaces in Collections

Note that this diagram shows only core interfaces.

Collection : Root interface with basic methods like add(), remove(), contains(), isEmpty(), addAll(), ... etc.

Set : Doesn't allow duplicates. Example implementations of Set

```
      interface are HashSet (Hashing based) and TreeSet (balanced
      BST based). Note that TreeSet implements SortedSet.

List : Can contain duplicates and elements are ordered. Example
       implementations are LinkedList (linked list based) and
       ArrayList (dynamic array based)

Queue : Typically order elements in FIFO order except exceptions
        like PriorityQueue.

Deque : Elements can be inserted and removed at both ends. Allows
        both LIFO and FIFO.

Map : Contains Key value pairs. Doesn't allow duplicates.  Example
      implementation are HashMap and TreeMap.
      TreeMap implements SortedMap.

The difference between Set and Map interface is, in Set we have only
keys, but in Map, we have key value pairs.
```

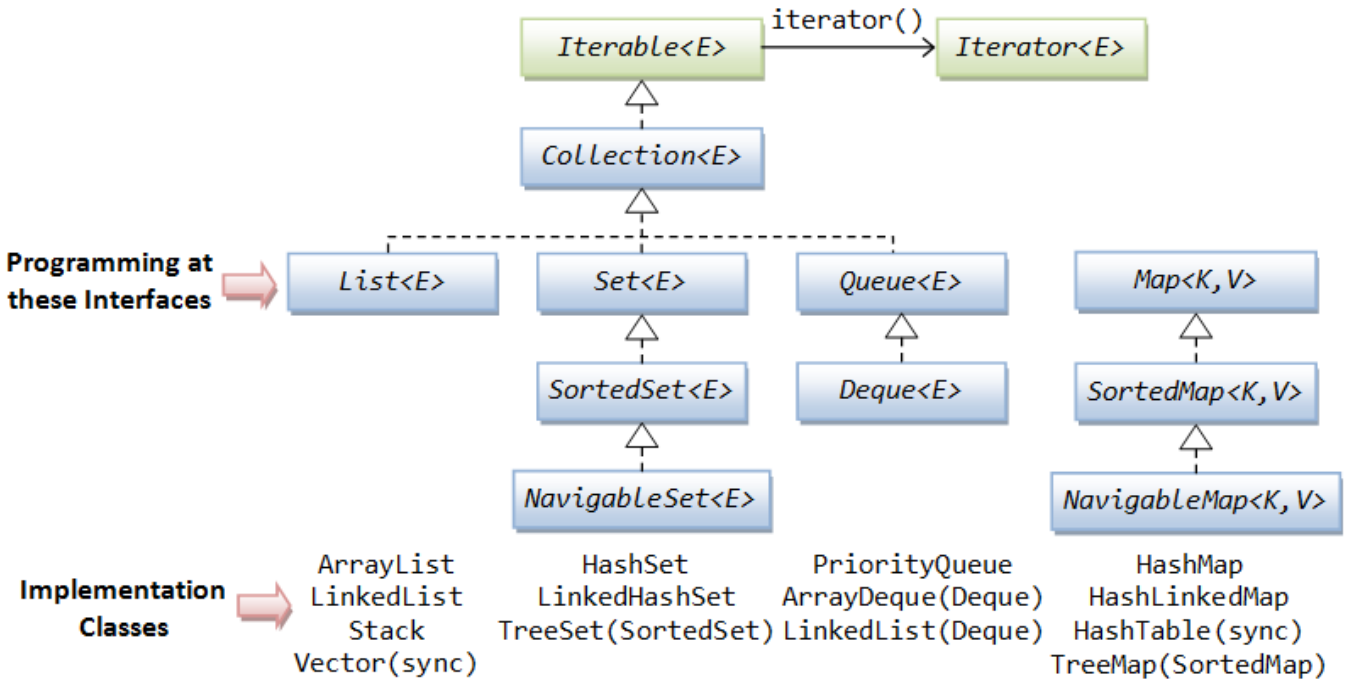


Image Source : https://www.ntu.edu.sg/home/ehchua/programming/java/J5c_Collection.html

We will soon be discussing examples of interface implementations.

This article is contributed by **Dharmesh Singh**. If you like GeeksforGeeks and would like to contribute, you can also write an article and mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information ab the topic discussed above