Java Collections API

Collections in Java

- A framework that provides an architecture to store and manipulate the group of objects
- Operations on data:
  - Searching
  - Sorting
  - Insertion
  - Manipulation
  - Deletion

API :

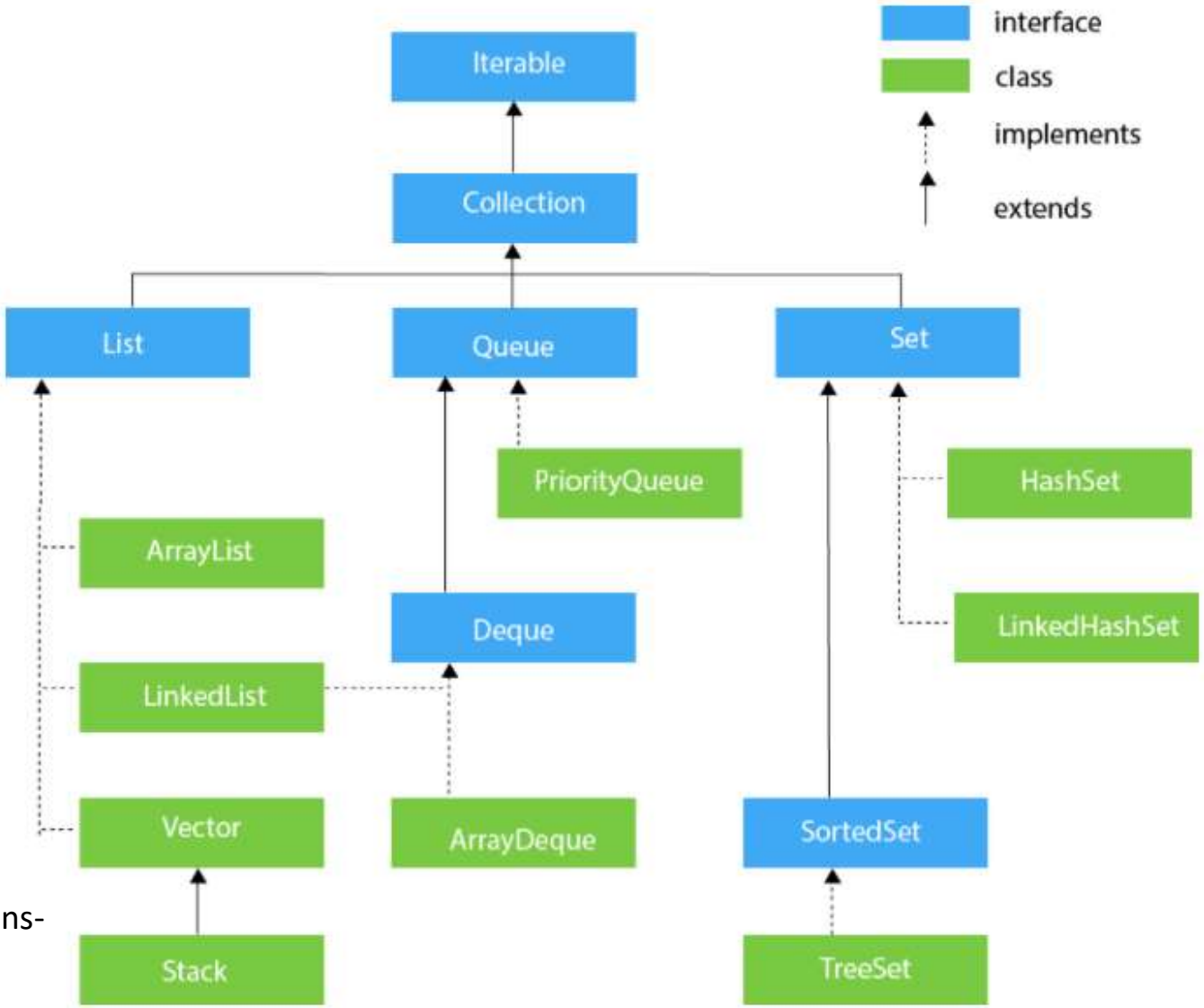

Image source :
https://www.javatpoint.com/collections-in-java

1. Iterable Interface:

```
public interface Iterable<T>
{
        Iterator<T> iterator();
}
```

2. Iterator Interface:

```
public interface Iterator<E>
{
        boolean hasNext();
        E next();
        void remove();
}
```

3. Iterable <- Collection <- List <- ArrayList

```
class TestJavaCollection1{
        public static void main(String args[]){
                ArrayList<String> list=new ArrayList<String>();
                list.add("Name1");//Adding object in arraylist
                list.add("Name2");
                list.add("Name3");
                list.add("Name4");

                //Traversing list through Iterator
                Iterator itr=list.iterator();
                while(itr.hasNext()){
                        System.out.println(itr.next());
                }
        } }
```

Difference between List, Queue and Set

List	Queue	Set
Ordered Collection	Ordered Collection	Unordered Collection
Can have duplicate elements	Can have duplicate elements	Can not have duplicate elements
Add, remove, or update any element directly	FIFO structure	Add, or remove elements but there is no order.

Iterator Interface

- The object of class which has implemented Iterator interface must have to implement its 3 methods to traverse a collection.

Method	Description
public boolean hasNext()	It returns true if the iterator has more elements otherwise it returns false.
public Object next()	It returns the element and moves the cursor pointer to the next element.
public void remove()	It removes the last elements returned by the iterator.

ArrayList

- It uses a *dynamic array* for storing the elements.
- Same as an array, but there is *no size limit*.
- Elements can be added or removed anytime. So, it is much more flexible than the traditional array.
- It is in *java.util* package.
- creating old non-generic arraylist : *ArrayList list=new ArrayList();*
- creating new generic arraylist : *ArrayList<String> list=new ArrayList<String>();*

LinkedList

- Java LinkedList class uses a doubly linked list to store the elements.
- Java LinkedList class can be used as a list, stack or queue.
- Creation of object is similar to ArrayList.

Vector

- Same as ArrayList.
- But Vector is synchronized, ArrayList is non-synchronized.

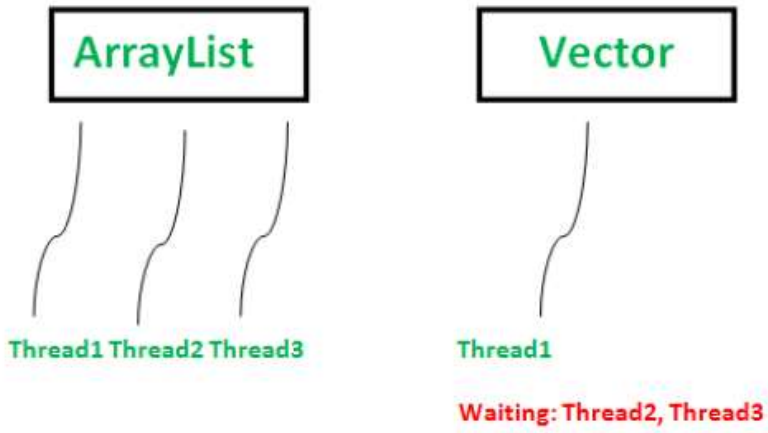

Image Source : https://www.geeksforgeeks.org/vector-vs-arraylist-java/

Difference between ArrayList and Vector

ArrayList	Vector
1) ArrayList is **not synchronized**.	Vector is **synchronized**.
2) ArrayList **increments 50%** of current array size if the number of elements exceeds from its capacity.	Vector **increments 100%** means doubles the array size if the total number of elements exceeds than its capacity.
3) ArrayList is **not a legacy** class. It is introduced in JDK 1.2.	Vector is a **legacy** class.
4) ArrayList is **fast** because it is non-synchronized.	Vector is **slow** because it is synchronized, i.e., in a multithreading environment, it holds the other threads in runnable or non-runnable state until current thread releases the lock of the object.

Stack

- linear data structure.
- It is based on **Last-In-First-Out** (LIFO).
- It has two main methods : push and pop

PriorityQueue

- Provides the facility of using queue. But it does not orders the elements in FIFO manner.
- The elements are ordered as per their priority.
- In Java to implement priority queue, you require to implement comparable interface.

Directory

- An abstract class of Java.util package.
- Facilitates to store key-value pairs.