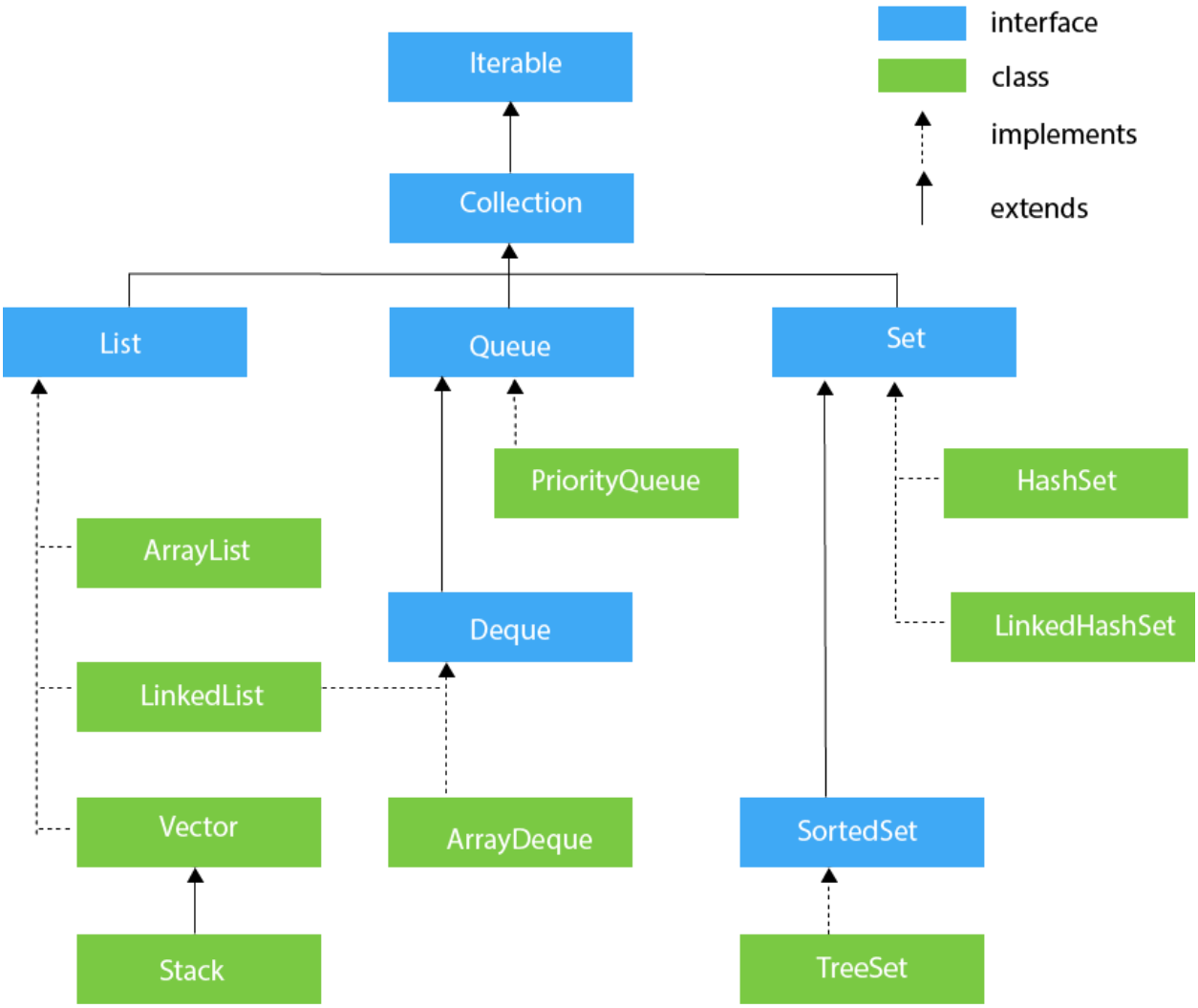
Collections

- Collections in Java is a framework that provides an architecture to store and manipulate the group of objects. In java,everything is an object,therefore collections operate on objects.
- Collections can achieve all the operations on data such as searching, sorting, insertion, manipulation, and deletion.
- Framework provides a ready-made architecture, and represents a set of classes and interface.



- Iterable interface provides the root interface for all the classes in the collection interface. Iterable interface provides the facility of iterating the elements in a forward direction only.

- The most common type of collections used in Java are ArrayList, LinkedList, Stacks, Sets.

List Interface

- List interface is the child interface of Collection interface. It inhibits a list type data structure in which we can store the ordered collection of objects. It can have duplicate values.
- List interface is implemented by the classes ArrayList, LinkedList, Vector, and Stack.

ArrayList

- The ArrayList class implements the List interface. It uses a dynamic array to store the duplicate element of different data types.
- The ArrayList class maintains the insertion order and is non-synchronized. The elements stored in the ArrayList class can be randomly accessed. Consider the following example.

```java
import java.util.*; import java.lang.*; class Main{ public static void main(String args[]){ ArrayList<String> list = new ArrayList<>(); list.add("xyz"); list.add("abc"); list.add("123"); for (String i : list) { System.out.print(i + " "); } list.remove(2); // deletion based on index. for (String i : list){ System.out.print(i + " "); } System.out.println(); list.remove(new String("xyz")); // deletion based on value. for (String i : list){ System.out.print(i + " "); } System.out.println(); System.out.println(list.size());//size of list. } }
```

Java

```
Output : xyz abc 123 xyz abc abc 1
```

Java

Time Complexity

Operation	Time Complexit
add()	O(1)
add(index,element)	O(N)
remove()	O(N)
contains()	O(N)
indexOf()	O(N)

COUNT 5

Custom Comparator

- Comparator interface is used to order the objects of user-defined class.
- This interface is present in java.util package and contains 2 methods compare(Object obj1, Object obj2) and equals(Object element).
- Using comparator, we can sort the elements based on data members. For instance it may be on name, age or anything else. Consider the following example.

```java
import java.util.*; import java.lang.*; import java.io.*; class compare
implements Comparator<Pair>{//custom comparator public int compare(Pair
a, Pair b){ if (a.x != b.x) return a.x - b.x; return a.y - b.y; } }
class Pair{ int x; int y; Pair(int x,int y){ this.x = x; this.y = y; } }
public class Main{ public static void main(String[] args){
TreeMap<Pair,Integer> map = new TreeMap<>(new compare()); map.put(new
Pair(1,2),1); map.put(new Pair(1,3),2); map.put(new Pair(3,2),3);
map.put(new Pair(2,3),4); for (Pair i : map.keySet()){
System.out.println(i.x + "\t" +i.y + "\t" + map.get(i)); } } }
```

Java

```
Output : 1 2 1 1 3 2 2 3 4 3 2 3
```

Java