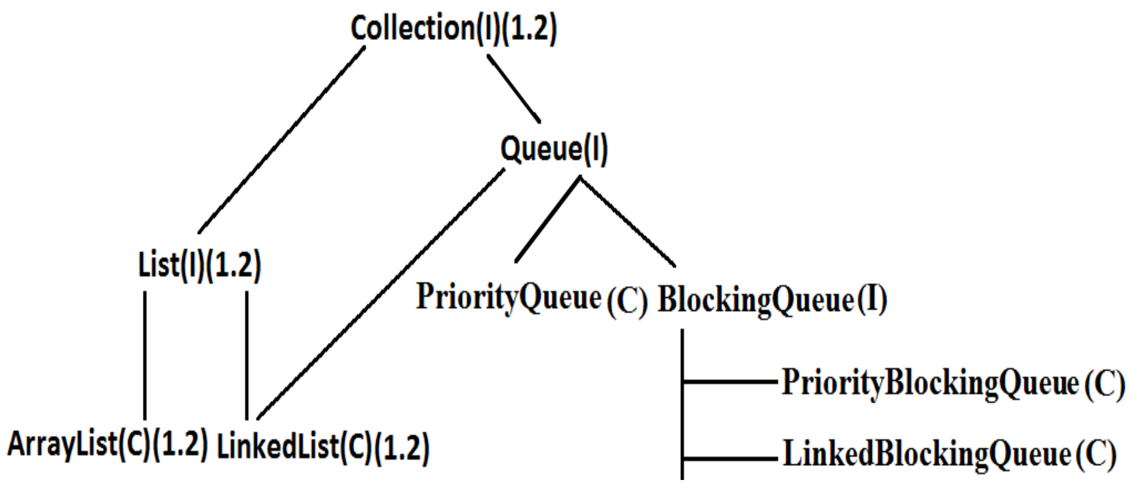
1.5 enhancements

Queue interface

Diagram:



1. Queue is child interface of Collections.
2. If we want to represent a group of individual objects prior (happening before something else) to processing then we should go for Queue interface.
3. Usually Queue follows first in first out(FIFO) order but based on our requirement we can implement our own order also.
4. From 1.5v onwards LinkedList also implements Queue interface.
5. LinkedList based implementation of Queue always follows first in first out order.

Assume we have to send sms for one lakh mobile numbers , before sending messages we have to store all mobile numbers into Queue so that for the first inserted number first message will be triggered(FIFO).

Queue interface methods:

1. boolean affer(Object o);
   To add an object to the Queue.
2. Object poll() ;
   To remove and return head element of the Queue, if Queue is empty then we will get null.
3. Object remove();
   To remove and return head element of the Queue. If Queue is empty then this method raises Runtime Exception saying NoSuchElementException.

4. **Object peek();**
   To return head element of the Queue without removal, if Queue is empty this method returns null.
5. **Object element();**
   It returns head element of the Queue and if Queue is empty then it will raise Runtime Exception saying NoSuchElementException.

<u>PriorityQueue:</u>

1. **PriorityQueue is a data structure to represent a group of individual objects prior to processing according to some priority.**
2. **The priority order can be either default natural sorting order (or) customized sorting order specified by Comparator object.**
3. **If we are depending on default natural sorting order then the objects must be homogeneous and Comparable otherwise we will get ClassCastException.**
4. **If we are defining our own customized sorting order by Comparator then the objects need not be homogeneous and Comparable.**
5. **Duplicate objects are not allowed.**
6. **Insertion order is not preserved but all objects will be inserted according to some priority.**
7. **Null is not allowed even as the 1st element for empty PriorityQueue.Otherwise we will get the "NullPointerException".**

<u>Constructors:</u>

1. **PriorityQueue q=new PriorityQueue();**
   Creates an empty PriorityQueue with default initial capacity 11 and default natural sorting order.
2. **PriorityQueue q=new PriorityQueue(int initialcapacity,Comparator c);**
3. **PriorityQueue q=new PriorityQueue(int initialcapacity);**
4. **PriorityQueue q=new PriorityQueue(Collection c);**
5. **PriorityQueue q=new PriorityQueue(SortedSet s);**

<u>Example 1:</u>

```
import java.util.*;
class PriorityQueueDemo
{
	public static void main(String[] args)
	{
		PriorityQueue q=new PriorityQueue();
		//System.out.println(q.peek());//null

	//System.out.println(q.element());//NoSuchElementExcepti
on
		for(int i=0;i<=10;i++)
		{
			q.offer(i);
		}
		System.out.println(q);//[0, 1, 2, 3, 4, 5, 6, 7, 
8, 9, 10]
		System.out.println(q.poll());//0
		System.out.println(q);//[1, 3, 2, 7, 4, 5, 6, 10, 
8, 9]
		}
}
```

<u>Note:</u> Some platforms may not provide proper supports for PriorityQueue [windowsXP].

<u>Example 2:</u>

```
import java.util.*;
class PriorityQueueDemo
{
	public static void main(String[] args)
	{
		PriorityQueue q=new PriorityQueue(15,new 
MyComparator());
		q.offer("A");
		q.offer("Z");
		q.offer("L");
		q.offer("B");
		System.out.println(q);//[Z, B, L, A]
		}
}
class MyComparator implements Comparator
{
	public int compare(Object obj1,Object obj2)
	{
		String s1=(String)obj1;
		String s2=obj2.toString();
		return s2.compareTo(s1);
	}
}
```

<u>1.6v Enhancements :</u>