# Collection集合

单例集合

Java Collection容器：只能放对象，不能放基本数据类型，而是要使用基本数据类型对应的包装类；

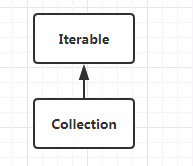
优点：降低编程难度，提升程序性能，提升API间的相互操作性，降低学习难度，增加程序的重用性；

Java Collection能容纳任何类型的对象，表面上是通过泛型机制实现的，但是泛型本身不需要java虚拟机支持，只是在编译的时候做了一下简单的字符串替换即可；实质是java的单继承机制才是保证容器能够容纳任何类型对象的根本，因为所有对象都是Object的子类，容器只要能够存放Object对象就可以了；泛型机制只是简化编程，编译器会自动完成强制类型转换；

Java对象存放在堆上，且对象只能通过引用访问，容器里放的其实是对象的引用而不是对象本身

Collection实现了java.lang.Iterable<T>，那么collection的最终实现类就可以使用for each循环；

实现java.lang.Iterable<T>接口的对象可以使用for each循环；



迭代器：提供遍历容器中元素的方法，只有容器本身清除容器里元素的组织方法，因此迭代器只能通过容器本身的导

Collection的操作：

add(Object o)：增加元素

addAll(Collection c)：...

clear()：...

contains(Object o)：是否包含指定元素

containsAll(Collection c)：是否包含集合c中的所有元素

iterator()：返回Iterator对象，用于遍历集合中的元素

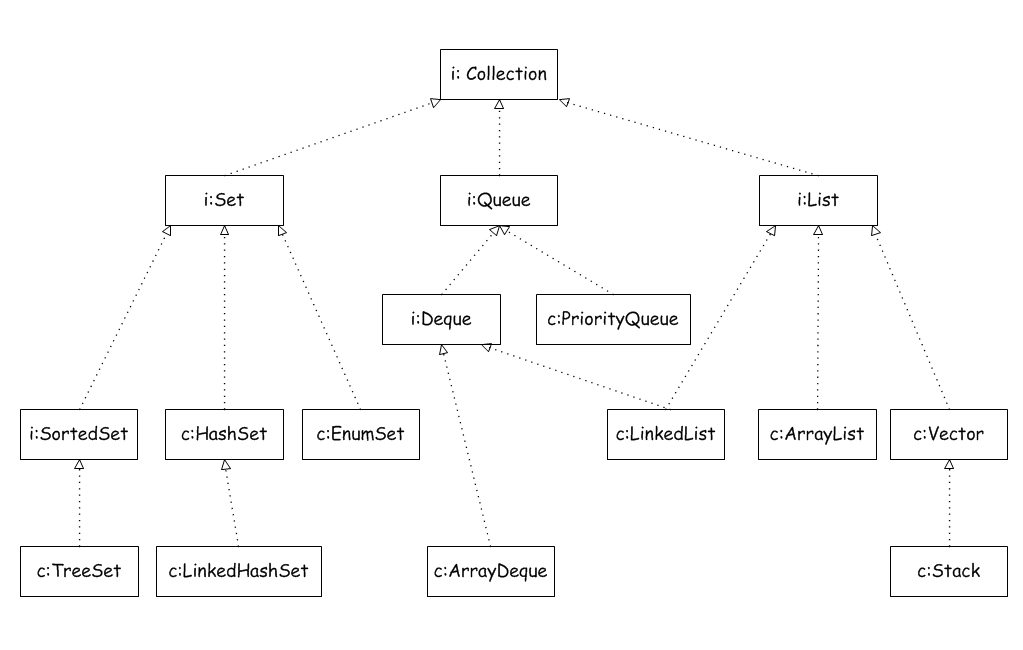
remove(Object o)：移除元素

removeAll(Collection c)：相当于减集合c

retainAll(Collection c)：相当于求与c的交集

size()：返回元素个数

toArray()：把集合转换为一个数组



子接口List,Set,Queue

List代表有序、重复的集合；List子接口是有序集合，每个位置上的元素可以为null，增加了与位置索引相关的操作

add(int index, Object o)：在指定位置插入元素

addAll(int index, Collection c)：

get(int index)：取得指定位置元素

indexOf(Obejct o)：返回对象o在集合中第一次出现的位置

lastIndexOf(Object o)：

remove(int index)：删除并返回指定位置的元素

set(int index, Object o)：替换指定位置元素

subList(int fromIndex, int endIndex)：返回子集合

Set无序的，没有重复对象的集合

Queue队列是一种特殊的线性表，它只允许在表的前端进行删除操作，而在表的后端进行插入操作。

Queue中增加的操作

**boolean** add(E e);队列添加元素，如果队列没有满能够添加成功返回true，队列没有空间就抛出IllegalStateException异常

**boolean** offer(E e); 添加一个元素并返回true，如果队列已满，需要抛出异常则返回false

E remove();删除队列头，这个方法和poll方法的唯一区别是当队列为空时会抛出异常

E poll();删除队列头，如果队列为空了，那么就返回null;

E element();取出队列的头（head）元素,不删除这个节点，和peek不同的是队列为空就会抛出异常

E peek();取出队列的头元素，不删除这个节点，如果队列为空返回null 不抛出异常