==========【java.util.Queue.java】==========

--1.Queue为接口（public interface Queue<E> extends Collection<E> ）

--1.1 Queue接口提供对于元素的插入、提取和查看操作

有两类方法：一类方法在操作失败的时候会抛出异常；另一类方法在操作失败时返回特殊值（根据不同的操作，返回null或者false）
后一类方法专门设计用在容量受限的队列。
在大多数实现中，插入操作不会失败

Queue通常(但不一定)是以FIFO(先进先出)方式对元素排序。
例外情况包括：
优先级队列，它根据提供的比较器对元素排序，或者元素的自然顺序，以及后进先出(LIFO)队列(或堆栈)，以后进先出(last-in-first-out)方式对元素排序。
无论使用什么顺序，队列的头部都是通过调用remove()或poll()来删除元素。

在FIFO(先进先出)队列中，所有新元素都插入到队列的尾部。
其他类型的队列可能使用不同的放置规则。每个队列实现都必须指定其排序属性。

--1.2 方法概览

Summary of Queue methods

	Throws exception	*Returns special value*
Insert	`add(e)`	`offer(e)`
Remove	`remove()`	`poll()`
Examine	`element()`	`peek()`

--1.3 boolean add(E e);

如果可以，在不违反容量限制的情况下立即将指定的元素插入到此队列中，

如果成功则返回true，

如果当前没有空间可用，则抛出IllegalStateException异常。

如果指定元素的类阻止将其添加到此队列中，则抛出ClassCastException异常

如果指定的元素为空，且此队列不允许空元素，则抛出NullPointerException异常

如果此元素的某些属性阻止将其添加到此队列中，则抛出IllegalArgumentException异常

--1.4 boolean offer(E e);

如果可以，在不违反容量限制的情况下，立即将指定的元素插入到此队列中。

如果插入成功，返回ture，否则返回false

如果指定元素的类阻止将其添加到此队列中，则抛出ClassCastException异常

如果指定的元素为空，且此队列不允许空元素，则抛出NullPointerException异常

如果此元素的某些属性阻止将其添加到此队列中，则抛出IllegalArgumentException异常

--1.5 E remove();

返回并删除队列头部元素，
如果队列为空是，返回NoSuchElementException异常

--1.6 E poll();

返回并删除队列头部元素，
如果此队列为空，则返回null。

--1.7 E element();

返回队列头部的元素但是不删除它
如果队列为空是，返回NoSuchElementException异常

--1.8 E peek();

返回队列头部的元素但是不删除它
如果队列为空是，返回null

==========【java.util.PriorityQueue.java】==========

--1.PriorityQueue为类（public class PriorityQueue<E> extends AbstractQueue<E> implements java.io.Serializable ）

--1.1 概述

基于优先级堆的无限优先级队列。

优先队列的元素根据它们的自然顺序排序，或者根据使用的构造函数由队列构建时提供的比较器排序。

优先队列不允许空元素。

依赖于自然排序的优先级队列也不允许插入不可比较的对象(这样做可能会导致ClassCastException)。

相对于指定的顺序，此队列的头是最小的元素。

如果多个元素为最小值绑定，则head是这些元素之一——绑定是任意断开的。

队列检索操作：poll, remove, peek, and element 都是访问队列的头部元素。

优先队列是无界的，但是有一个内部容量来控制用于在队列中存储元素的数组的大小。

它总是至少与队列大小一样大。当元素被添加到优先队列时，它的容量会自动增长。增长策略的细节没有指定。

该类及其迭代器实现集合和迭代器接口的所有可选方法。

方法Iterator()中提供的迭代器不能保证以任何特定的顺序遍历优先队列的元素。

如果需要有序遍历，可以考虑使用array .sort(pq.toArray())。

注意，这个实现不是同步的。

如果任何线程修改队列，多线程不应该并发地访问PriorityQueue实例。相反，使用线程安全的PriorityBlockingQueue类。

实现说明:

此实现为进入和退出队列方法(offer、poll、remove()和add)提供O(log(n))时间;

remove(Object)和contains(Object)方法为线性时间O(n);

检索方法(peek、element和size)为常数时间O(1)。

该类是Java集合框架的成员。

--1.2 方法概览（时间有限，当前重点关注对Queue接口方法的实现，不再细看从其他接口或类继承实现的方法）

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
boolean	**add**(E e) Inserts the specified element into this priority queue.
void	**clear**() Removes all of the elements from this priority queue.
Comparator<? super E>	**comparator**() Returns the comparator used to order the elements in this queue, or null if this queue is sorted according to the **natural ordering** of its elements.
boolean	**contains**(Object o) Returns true if this queue contains the specified element.
Iterator<E>	**iterator**() Returns an iterator over the elements in this queue.
boolean	**offer**(E e) Inserts the specified element into this priority queue.
E	**peek**() Retrieves, but does not remove, the head of this queue, or returns null if this queue is empty.
E	**poll**() Retrieves and removes the head of this queue, or returns null if this queue is empty.
boolean	**remove**(Object o) Removes a single instance of the specified element from this queue, if it is present.
int	**size**() Returns the number of elements in this collection.
Spliterator<E>	**spliterator**() Creates a ***late-binding*** and *fail-fast* **Spliterator** over the elements in this queue.
Object[]	**toArray**() Returns an array containing all of the elements in this queue.
<T> T[]	**toArray**(T[] a) Returns an array containing all of the elements in this queue; the runtime type of the returned array is that of the specified array.

Methods inherited from class java.util.AbstractQueue

addAll, element, remove

Methods inherited from class java.util.AbstractCollection

containsAll, isEmpty, removeAll, retainAll, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Methods inherited from interface java.util.Collection

containsAll, equals, hashCode, isEmpty, parallelStream, removeAll, removeIf, retainAll, stream

Methods inherited from interface java.lang.Iterable

forEach

--1.3 public boolean add(E e)

将指定的元素插入此优先队列。

如果成功，返回ture

如果指定的元素不能根据优先队列的顺序与当前优先队列中的元素进行比较，抛出ClassCastException异常

如果指定的元素为空，抛出NullPointerException异常

return offer(e);

直接调用offer()方法。

--1.4 public boolean offer(E e)

将指定的元素插入此优先队列。

如果成功，返回ture

如果指定的元素不能根据优先队列的顺序与当前优先队列中的元素进行比较，抛出ClassCastException异常

如果指定的元素为空，抛出NullPointerException异常

首先判断插入元素如果为null，直接抛出NullPointerException

modCount++;

如果当前优先队列中的元素数已满，进行扩容grow(i + 1);

如果当前队列长度<64，则进行翻倍扩容（oldCapacity + 2）

如果大于>64，在增加50%长度（oldCapacity >> 1））

进行溢出检测，如果长度过大（超过int最大值-8），hugeCapacity(minCapacity);

如果minCapacity<0；抛出OutOfMemoryError；

取Integer.MAX_VALUE ， MAX_ARRAY_SIZE这两者之间较小的一个。

进行数组迁移：queue = Arrays.copyOf(queue, newCapacity);

元素数+1;

如果当前队列为空，没有元素，直接将此元素加入到队列中

如果当前队列不为空，调用siftUp(i, e);进行优先级判断，然后插入到合适的位置

在k位置插入项目x，通过向上提升x直到它大于或等于其父节点，或者是根节点，来保持堆不变。

简化和加快强制和比较。Comparable和Comparator版本被分成不同的方法，它们在其他方面是相同的。(类似siftDown。)

返回true，表示插入成功

--1.5 public boolean remove(Object o);

如果指定元素存在，则从此队列中移除该元素的单个实例。

更正式地说，删除一个元素{ e}，使得{o.equals(e)}，

如果此队列包含一个或多个此类元素。返回{true}

当且仅当此队列包含指定的元素时

首先调用indexOf(o);轮询判断当前队列中存在此元素（o.equals(queue[i])），获取元素下标（o不能为null，否则返回false）

如果不存在，返回下标为-1，返回false

如果存在，调用removeAt(i);

返回true；

--1.6public E poll()

检索并删除此队列的头，如果此队列为空，则返回null。

如果当前队列为空，返回null

modCount++;

取队列头部元素queue[0]作为返回值

int s = --size;

E x = (E) queue[s];

queue[s] = null;

如果取出头后，当前队列不为空，siftDown(0, x);

--1.7 public E element() 【注意：此处是由它的父类抽象类AbstractQueue实现的】

返回队列头部的元素但是不删除它

直接调peek();

如果为空，抛出NoSuchElementException()异常。

--1.8 E peek();

返回队列头部的元素但是不删除它
如果队列为空是，返回null

return (size == 0) ? null : (E) queue[0];

---END