Java中的Queue

# queue与deque

## 单词释义

**deque** 英['dek] 美**['dek]** 双端队列;

**queue** 英[kju:] 美**[kju]** n. 队列；**（**人或车辆） 行列，长队; 辫子;

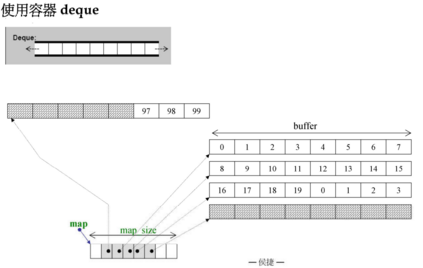
vi. （人、车等） 排队等候; vt. （使） 排队，列队等待;

## deque

deque**双向开口可进可出**的容器；



我们知道**连续内存的容器不能随意扩充**,因为这样容易扩充别人那去；deque却可以,它创造了内存连续的假象。其实**deque由一段一段构成 ,他是分段连续,而不是内存连续**。当走向段的尾端时候自动跳到下一段 所以支持**迭代器++ 操作**,自动跳到下一段的方法由operator++实现；deque每次扩充，申请一个段。



# Queue接口

## 接口的基本介绍

Interface Queue<E>接口存在于java.util包中，继承了Collection<E>接口，Collection<E>接口继承了Iterable<E>接口。

**public interface Queue<E> extends Collection<E>**

继承的接口有两个：All Superinterfaces**: Collection<E>, Iterable<E>**

子接口：All Known Subinterfaces: **BlockingDeque<E>, BlockingQueue<E>, Deque<E>, TransferQueue<E>**

子类：All Known Implementing Classes: AbstractQueue, **ArrayBlockingQueue**, **ArrayDeque**, **ConcurrentLinkedDeque**, **ConcurrentLinkedQueue**, DelayQueue, LinkedBlockingDeque, LinkedBlockingQueue, **LinkedList**, LinkedTransferQueue, **PriorityBlockingQueue**, **PriorityQueue**, SynchronousQueue

## blocking queue methods阻塞队列方法：

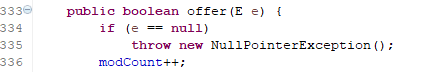
在Queue接口没有定义用于并发编程的阻塞队列相关方法，这些阻塞队列方法定义在了**BlockingQueue接口中，用于扩展Queue接口。**

**The Queue interface does not define the blocking queue methods**, which are common in concurrent programming. These methods, which wait for elements to appear or for space to become available, are defined in the **BlockingQueue interface**, which extends this interface.

**Queue implementations** generally do not define element-based versions of methods equals and hashCode but instead inherit the identity based versions from class Object, because element-based equality is not always well-defined for **queues** with the same elements but different ordering properties.

# Queue对null元素的限制

一般情况下，Queue不允许插入null元素，插入null元素是报出异常。如：



**Queue implementations generally do not allow insertion of null elements**, although some implementations, such as **LinkedList**, do not prohibit insertion of null. Even in the implementations that permit it, null should not be inserted into a Queue, as null is also used as a special return value by the poll method to indicate that the queue contains no elements.

LinkedList实现的接口有:All Implemented Interfaces:

Serializable, Cloneable, Iterable<E>, Collection<E>, Deque<E>, List<E>, **Queue**<E>

但是LinkedList中可以插入null元素也可以获取元素。

LinkedList<Integer> li = **new** LinkedList<Integer>();

li.add(**null**);

li.add(3);

**System.*out*.println(li.get(0));//null**

System.***out***.println(li.get(1));//3

**对于Queue：**

Queue<Integer> queue = new PriorityQueue<>();

queue.add(**null**);//抛出java.lang.NullPointerException

# Queue接口的主要操作方法

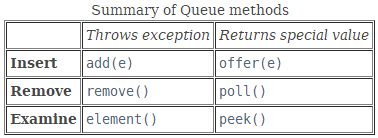
A collection designed for holding elements prior to processing. **Besides basic Collection operations, queues provide additional insertion, extraction, and inspection operations**. Each of these methods exists in two forms: one throws an exception if the operation fails, the other returns a special value (either null or false, depending on the operation). The latter form of the insert operation is designed specifically for use with capacity-restricted Queue implementations; in most implementations, insert operations cannot fail.

除了Collection接口的方法外，Queue还有另加的添加、删除、检查操作，每种都有两个方法，一个抛出异常，一个放回特定值。

增加：add(e)与offer(e)

删除：remove()与poll()

检查：element()与peek()



# Queue中的元素顺序

**Queues typically, but do not necessarily, order elements in a FIFO (first-in-first-out) manner.** Among the exceptions are priority queues, which order elements according to a supplied comparator, or the elements' natural ordering, and LIFO queues (or stacks) which order the elements LIFO (last-in-first-out). Whatever the ordering used, **the head of the queue** is that element which would be removed by a call to **remove() or poll()**. In a FIFO queue, all new elements are inserted at the tail of the queue. Other kinds of queues may use different placement rules. Every Queue implementation must specify its ordering properties.

除了优先队列(priority queues)之外，一般Queue默认为**FIFO先入先出**的方式。满足FIFO的队列，新元素都是插入到队列的尾部。其他方式的队列需要根据自己的规则来定。

优先队列(priority queues)可以根据①提供的比较器Comparator②元素的自然顺序③LIFO方式指定特定的优先级。无论哪种优先级方式，remove和poll方法删除的总是队列的头元素。

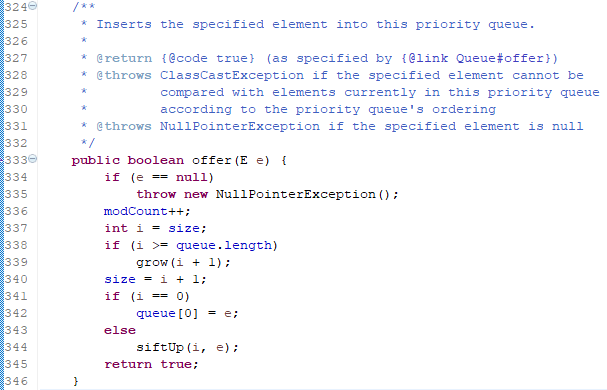
# Queue接口的方法介绍

## offer(e)与add(e)方法

### boolean offer(E e)

Inserts the specified element into this queue if it is possible to do so immediately without violating capacity restrictions.

The **offer** method inserts an element if possible, otherwise returning false. This differs from the Collection.add method, which can fail to add an element only by throwing an unchecked exception. The offer method is designed for use when failure is a normal, rather than exceptional occurrence, for example, in fixed-capacity (or "bounded") queues.



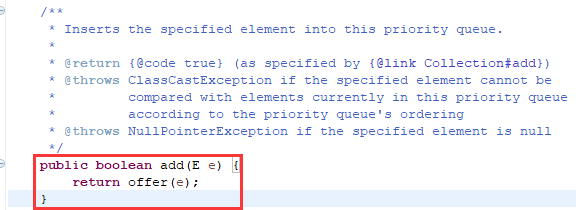
### boolean add(e)

add(e)一般是调用offer(e)实现的。

**boolean add(E e)**

Inserts the specified element into this queue if it is possible to do so immediately without violating capacity restrictions, returning true upon success and throwing an IllegalStateException if no space is currently available.

PriorityQueue的源代码：



## remove() 和 poll()方法

The remove() and poll() methods remove and return the head of the queue. Exactly which element is removed from the queue is a function of the queue's ordering policy, which differs from implementation to implementation. The **remove() and poll()** methods differ only in their behavior when the queue is empty: the remove() method throws an exception, while the poll() method returns null.

### E remove()

Retrieves and removes the head of this queue.

### E poll()

Retrieves and removes the head of this queue, or returns null if this queue is empty.

## element() and peek()获取但不删除

The **element() and peek()** methods return, but do not remove, the head of the queue.

### E element()

**Retrieves**, but does not remove, the head of this queue.

### E peek()

**Retrieves**, but does not remove, the head of this queue, or returns null if this queue is empty.

# AbstractQueue

## 继承关系介绍

public **abstract** class AbstractQueue<E> extends **AbstractCollection**<E>

implements **Queue**<E>

AbstractQueue是一个抽象类，实现了Queue队列中最最基本的实现，整个队列集合框架中所有的实现类一般都直接或间接继承了该AbstractQueue类。

All Implemented Interfaces: Iterable<E>, Collection<E>, Queue<E>

Direct Known Subclasses: **ArrayBlockingQueue**, ConcurrentLinkedQueue, DelayQueue, LinkedBlockingDeque, **LinkedBlockingQueue**, LinkedTransferQueue, PriorityBlockingQueue, PriorityQueue, **SynchronousQueue**

This class is a member of the Java Collections Framework. Since: 1.5

## 功能介绍

AbstractQueue

This class provides **skeletal(骨骼的，提纲性的，最最基本的)** implementations of some Queue operations. The implementations in this class are appropriate when the base implementation does not allow null elements. Methods **add, remove, and element** are based on **offer, poll, and peek,** respectively, but throw exceptions instead of indicating failure via false or null returns.

A Queue implementation that extends this class must minimally define a method **Queue.offer(E)** which does not permit insertion of null elements, along with methods **Queue.peek(), Queue.poll(), Collection.size(), and Collection.iterator()**. Typically, additional methods will be overridden as well. If these requirements cannot be met, consider instead subclassing AbstractCollection.

## 构造方法

protected **AbstractQueue**()

Constructor for use by **subclasses**.

## 一般方法

boolean **add**(E e)

Inserts the specified element into this queue if it is possible to do so immediately without violating capacity restrictions, returning true upon success and throwing an IllegalStateException if no space is currently available.

boolean **addAll**(Collection<? extends E> c)

Adds all of the elements in the specified collection to this queue.

void **clear**()

Removes all of the elements from this queue.

E **element**()

Retrieves, but does not remove, the head of this queue.

E **remove**()

Retrieves and removes the head of this queue.