双端队列Deque

# 基于Deque接口的实现类

一共4个：**ArrayDeque**, **ConcurrentLinkedDeque**, **LinkedBlockingDeque**, **LinkedList**

其中LinkedBlockingDeque是直接实现BlockingDeque接口，间接实现Deque接口。

其余三个**ArrayDeque**, **ConcurrentLinkedDeque，LinkedList**都是直接实现Deque接口**。**

# java.util.Deque接口

Deque双端队列(**d**ouble **e**nded **que**ue==>Deque)发音deck

**A linear collection that supports element insertion and removal at both ends**. The name deque is short for "**double ended queue**" and is usually pronounced "**deck**". Most Deque implementations place **no fixed limits** on the number of elements they may contain, but this interface supports **capacity-restricted deques** as well as those with no fixed size limit.

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

继承的父接口All Superinterfaces: Collection<E>, Iterable<E>, **Queue**<E>

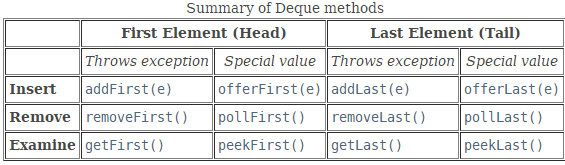
子接口：All Known Subinterfaces**: BlockingDeque<E>阻塞式双端队列**

所有子类：All Known Implementing Classes: **ArrayDeque**, **ConcurrentLinkedDeque**, **LinkedBlockingDeque**, **LinkedList**

# java.util.Deque接口方法介绍

This interface defines methods **to access the elements at both ends of the deque**. Methods are provided to **insert, remove, and examine the element**. 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 Deque implementations; in most implementations, insert operations cannot fail.

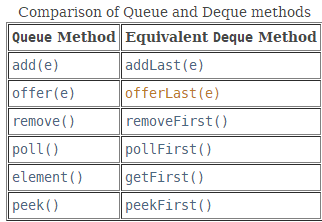
**The twelve methods** described above are summarized in the following table:



# Deque与Queue的区别

**This interface extends the Queue interface.** When a deque is used as a queue, **FIFO** (First-In-First-Out) behavior results. Elements are added at the end of the deque and removed from the beginning. The methods inherited from the Queue interface are precisely equivalent to Deque methods as indicated in the following table:

当双端队列Deque当做Queue使用的时候，表现为先入先出FIFO：即在队列尾部插入元素，从队列头部删除元素。

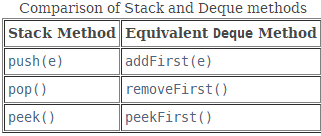


**Queue**的方法扩充到Deque一般都是添加了xxxFirst和xxxLast，实现了**双端队列**。

# Deque与Stack

Deques can also be used as **LIFO (Last-In-First-Out) stacks**. **This interface should be used in preference to(优先于) the legacy Stack class.** **When a deque is used as a stack, elements are pushed and popped from the beginning of the deque.** Stack methods are precisely(严格地，恰好地) equivalent to **Deque** methods as indicated in the table below:

当双端队列deque用作一个**栈stack**时，添加和删除都是从队列的头部操作。

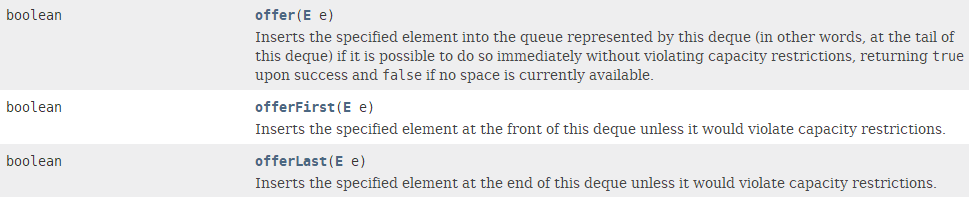


注意：Note that the **peek** method works equally well **when a deque is used as a queue or a stack**; in either case, elements are drawn from the beginning of the deque.

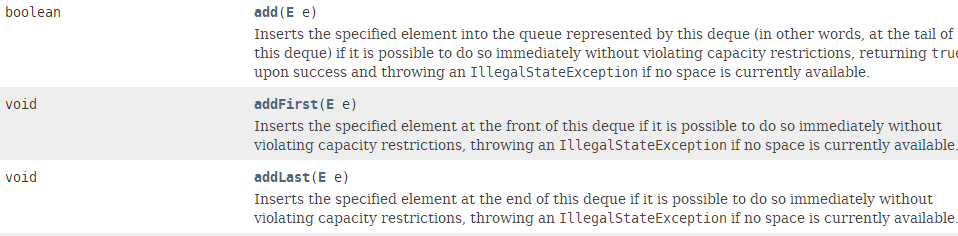
Deque中的peek方法无论用作queue还是stack都是从队列的头部取元素。

# Deque的方法

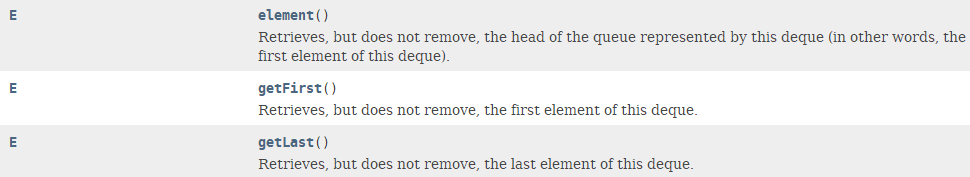
## offer、offerFirst、offerLast



## add、addFirst、addLast



## element、getFirst、getLast



**removeFirstOccurrence和removeLastOccurrence方法：删除interior elements**

This interface provides two methods **to remove interior elements**, **removeFirstOccurrence** and **removeLastOccurrence**.

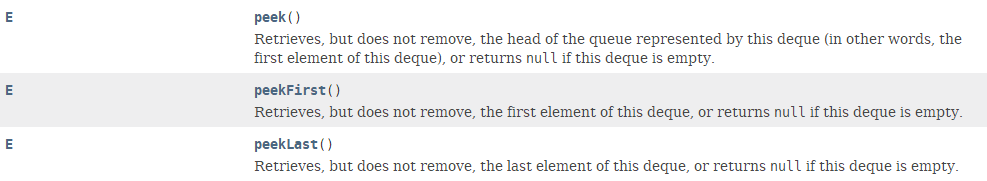
Unlike the **List** interface, this interface does not provide support for indexed access to elements.

和List接口不一样，**Deque**接口没有提供索引访问元素。

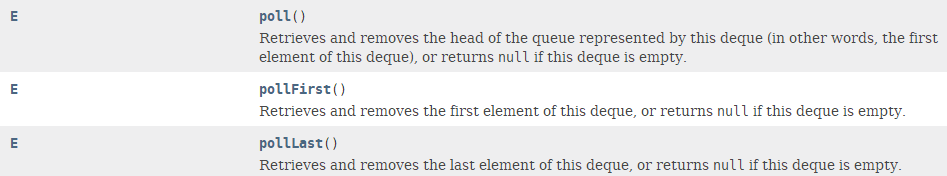
注意：LinkedList既实现了List接口也实现了Queue和Deque接口。

Deque implementations generally do not define **element-based versions of the equals and hashCode methods**, but instead inherit the **identity-based versions** from class Object.

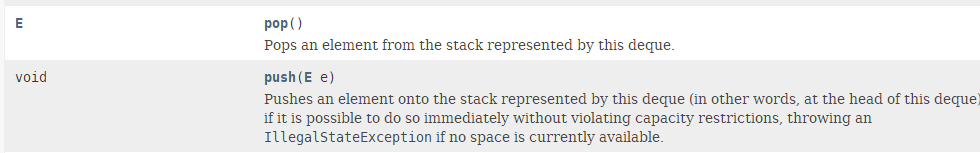
## peek、peekFirst、peekLast



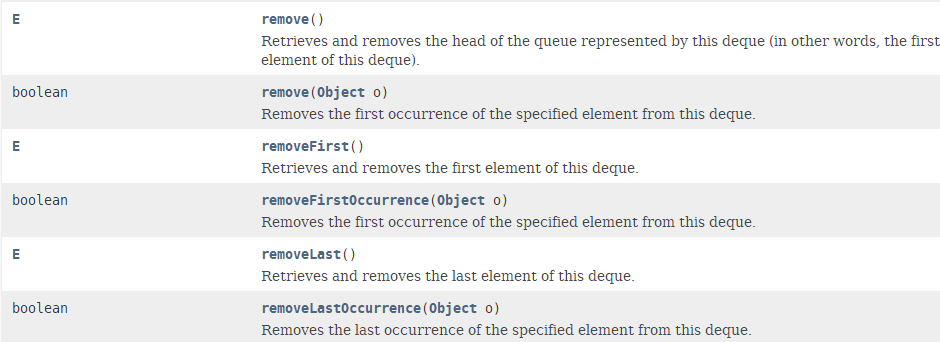
## poll、pollFirst、pollLast



## push、pop：stack的操作方法



## remove、removeFirst、removeLast、removeFirstOccurrence、removeLastOccurrence



## size



## contains



## iterator、descendingIterator：迭代器





# Deque对null元素

While Deque implementations **are not strictly required to prohibit the insertion of null elements**, they are strongly encouraged to do so. Users of any Deque implementations that do allow **null** elements are strongly encouraged not to take advantage of the ability to insert nulls. This is so because null is used **as a special return value** by various methods to indicated that the deque is empty.

尽管Deque没有严格禁止null元素的插入，但是强烈建议不要这么做。因为Deque的很多方法都是通过返回null值表明该Deque是空的。