BlockingDeque阻塞双端队列

# BlockingDeque接口及其实现类

接口：BlockingDeque继承于**BlockingQueue**接口。

**BlockingQueue及其实现类和BlockingDeque及其实现类**都存在于java.util.concurrent并发包中，属于并发编程中的常用类。而Queue和Deque接口存在于java.util包中。

实现类：**LinkedBlockingDeque**

# BlockingDeque阻塞双端队列接口

## 简单介绍

public interface BlockingDeque<E> extends **BlockingQueue**<E>, **Deque**<E>

BlockingDeque接口直接继承了两个接口BlockingQueue和Deque。

All Superinterfaces:**BlockingQueue**<E>, Collection<E>, Deque<E>, Iterable<E>, Queue<E>

All Known Implementing Classes: **LinkedBlockingDeque**

This interface is a member of the Java Collections Framework. **Since:1.6**

## 特殊功能

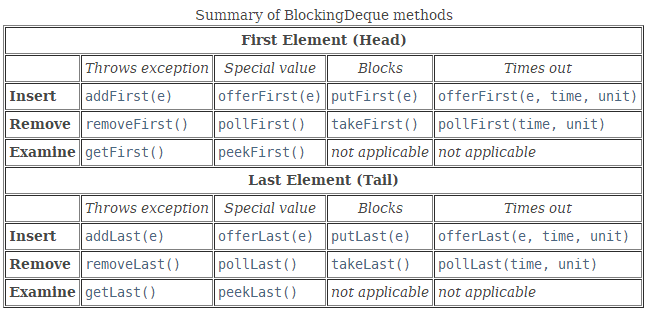
### 阻塞式双端队列

阻塞体现在两方面：1. 获取元素时，若队列为空，则一直等待；2. 存储元素时，若队列已满，一直等待。

A **Deque** that additionally supports blocking operations that wait for the deque to become non-empty when retrieving an element, and wait for space to become available in the deque when storing an element.

### 四种处理操作方式

**BlockingDeque** methods come **in four forms**, with different ways of handling operations that cannot be satisfied immediately, but may be satisfied at some point in the future: ①**one** throws an exception, ②the **second** returns a special value (either null or false, depending on the operation), ③the **third** blocks the current thread indefinitely until the operation can succeed, and ④ the **fourth** blocks for only a given maximum time limit before giving up. These methods are summarized in the following table:



双端队里相对于普通队列，增加了对**First和Last**的区别。

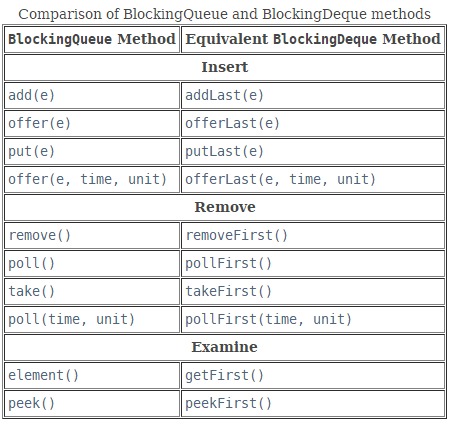
### 线程安全，不允许null元素、容量可能限定或不限定

Like any **BlockingQueue**, a BlockingDeque is thread safe, does not permit null elements, and may (or may not) be capacity-constrained.

### **directly as a FIFO BlockingQueue**

A BlockingDeque implementation may be used **directly as a FIFO BlockingQueue**. The methods inherited from the BlockingQueue interface are precisely equivalent to BlockingDeque methods as indicated in the following table:

BlockingQueue与BlockingDeque方法的对应关系。下面这个表就是两个接口的对应方法是等价的。



### Memory consistency effects

Memory consistency effects: As with other **concurrent** collections, actions in a thread prior to placing an object into a **BlockingDeque** happen-before actions subsequent to the access or removal of that element from the BlockingDeque in another thread.

## 接口方法

### 添加元素

add、addFirst、addLast

offer、offerFirst、offerLast(一种返回特定值，一种timeout)

put、putFirst、putLast

### 获取元素

take、takeFirst、takeLast

poll、pollFirst、pollLast

peek

### E peek()

**Retrieves**, but does not remove, the head of the queue represented by this deque (in other words, the first element of this deque), or returns null if this deque is empty.

### element

E element()

Retrieves, but does not remove, the head of the queue represented by this deque (in other words, the first element of this deque).

### remove

E remove()

Retrieves and removes the head of the queue represented by this deque (in other words, the first element of this deque).

boolean remove(Object o)

Removes the first occurrence of the specified element from this deque.

boolean removeFirstOccurrence(Object o)

Removes the first occurrence of the specified element from this deque.

boolean removeLastOccurrence(Object o)

Removes the last occurrence of the specified element from this deque.

### size

int size()

Returns the number of elements in this deque.

### push

void push(E e)

Pushes an element onto the stack represented by this deque (in other words, at the head of this deque) if it is possible to do so immediately without violating capacity restrictions, throwing an IllegalStateException if no space is currently available.

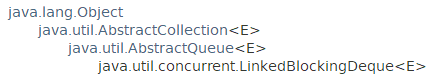
# LinkedBlockingDeque

## 继承实现关系介绍

public class **LinkedBlockingDeque**<E> extends **AbstractQueue**<E>

implements BlockingDeque<E>, Serializable

All Implemented Interfaces:Serializable, Iterable<E>, Collection<E>, **BlockingDeque**<E>, BlockingQueue<E>, Deque<E>, Queue<E>



This class is a member of the [Java Collections Framework](https://docs.oracle.com/javase/8/docs/technotes/guides/collections/index.html). Since: 1.6

## 功能介绍

### 是否容量限定可选

An **optionally-bounded blocking deque** based on linked nodes.

The optional capacity bound constructor argument serves as a way to **prevent excessive expansion**. The capacity, if unspecified, is equal to **Integer.MAX\_VALUE**. Linked nodes are dynamically created upon each insertion unless this would bring the deque above capacity.

### 操作时间一般固定

**Most operations run in constant time** (ignoring time spent blocking). Exceptions include remove, removeFirstOccurrence, removeLastOccurrence, contains, iterator.remove(), and the bulk operations, all of which run in linear time.

This class and its iterator implement all of the optional methods of the Collection and Iterator interfaces.

## 构造方法

### LinkedBlockingDeque()

Creates a LinkedBlockingDeque with a capacity of Integer.MAX\_VALUE.

### **LinkedBlockingDeque**(Collection<? extends E> c)

Creates a LinkedBlockingDeque with a capacity of Integer.MAX\_VALUE, initially containing the elements of the given collection, added in traversal order of the collection's iterator.

### LinkedBlockingDeque(int capacity)

Creates a LinkedBlockingDeque with the given (fixed) capacity.

## 一般方法

### 添加元素

add、addFirst、addLast

offer、offerFirst、offerLast

put、putFirst、putLast

### 获取元素

peek、peekFirst、peekLast

poll、pollFirst、pollLast

take、takeFirst、takeLast

element、getFirst、getLast

### drainTo

int drainTo(Collection<? super E> c)

Removes all available elements from this queue and adds them to the given collection.

int drainTo(Collection<? super E> c, int maxElements)

Removes at most the given number of available elements from this queue and adds them to the given collection.

### clear、remove、

remove、removeFist、removeLast

removeFirstOccurrence、removeLastOccurrence

### toArray

Object[] **toArray**()

Returns an array containing all of the elements in this deque, in proper sequence (from first to last element).

<T> T[] **toArray**(T[] a)

Returns an array containing all of the elements in this deque, in proper sequence; the runtime type of the returned array is that of the specified array.

### iterator、spliterator

### size