TransferQueue传输队列

# TransferQueue传输队列接口

## 继承关系

public interface TransferQueue<E> extends BlockingQueue<E>

**TransferQueue接口**直接继承于BlockingQueue接口，存在于java.util.concurrent包中。

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

一个实现类：LinkedTransferQueue链表传输队列。

This interface is a member of the Java Collections Framework. Since:1.7(JDK1.7才出现的)

## 功能介绍

### await receipt of elements by consumers

**A BlockingQueue** in which **producers** may wait for **consumers** to receive elements. A TransferQueue may be useful for example ***in message passing applications*** in which **producers sometimes (using method transfer(E)) await receipt of elements by consumers invoking take or poll**, while at other times enqueue elements (via method put) without waiting for receipt. Non-blocking and time-out versions of tryTransfer are also available. A TransferQueue may also be queried, **via hasWaitingConsumer(),** whether there are any threads waiting for items, which is a converse analogy (类似，类推，类比) to a **peek** operation.

analogy 英[əˈnælədʒi] 美[əˈnælədʒi] n. 类推; 类似，相似; 比拟，类比;

### capacity bounded

Like other blocking queues, a TransferQueue may be **capacity bounded**. If so, an attempted transfer operation may initially block waiting for available space, and/or subsequently block waiting for reception by a consumer.

Note that in a queue with zero capacity, such as **SynchronousQueue**, put and transfer are effectively **synonymous**(同义的).

**synonymous** 英[sɪˈnɒnɪməs]美[sɪˈnɑ:nɪməs]adj. 同义词的; 同义的，类义的;

## 接口方法介绍

### tryTransfer方法

boolean **tryTransfer**(E e)

Transfers the element to a waiting **consumer** immediately, if possible.

boolean **tryTransfer**(E e, long timeout, TimeUnit unit)

Transfers the element to a **consumer** if it is possible to do so before the timeout elapses.

### transfer(E e)

void transfer(E e) Transfers the element to a **consumer**, waiting if necessary to do so.

### hasWaitingConsumer()

boolean hasWaitingConsumer()

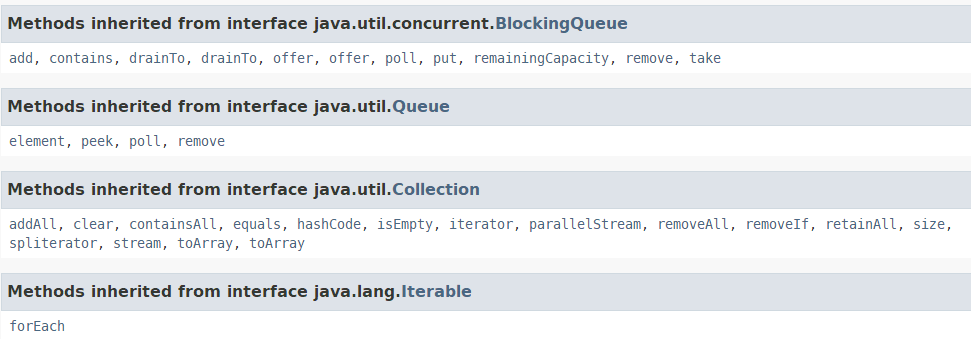
Returns true if there is at least one consumer waiting to receive an element via BlockingQueue.take() or timed poll.

### getWaitingConsumerCount()

int getWaitingConsumerCount()

Returns an estimate of the number of consumers waiting to receive elements via BlockingQueue.take() or timed poll.

### 其他接口方法



# LinkedTransferQueue链表传输队列

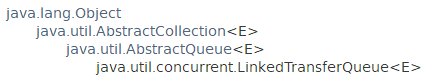
## 简单介绍

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

implements **TransferQueue**<E>, Serializable

**LinkedTransferQueue** 是TransferQueue接口目前惟一的一个实现类。存在java.util.concurrent包中。

实现的接口All Implemented Interfaces: Serializable, Iterable<E>, Collection<E>, **BlockingQueue**<E>, **TransferQueue**<E>, Queue<E>



This class is a member of the Java Collections Framework. Since: 1.7

## 特殊功能

### 基于链表的，无界传输队列，FIFO

**An unbounded TransferQueue based on linked nodes.** This queue orders elements FIFO (first-in-first-out) with respect to any given producer. The head of the queue is that element that has been on the queue the longest time for some producer. The tail of the queue is that element that has been on the queue the shortest time for some producer.

### 元素的遍历

Beware that, unlike in most collections, the **size** method is NOT a constant-time operation. Because of the asynchronous nature of these queues, determining the current number of elements requires **a traversal of the elements**, and so may report inaccurate results if this collection is modified during traversal. Additionally, the bulk operations addAll, removeAll, retainAll, containsAll, equals, and toArray are not guaranteed to be performed **atomically**. For example, an **iterator** operating concurrently with an addAll operation might view only some of the added elements.

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

### Memory consistency effects

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

## 构造方法

### LinkedTransferQueue()

Creates an initially empty LinkedTransferQueue.

### LinkedTransferQueue(Collection<? extends E> c)

Creates a **LinkedTransferQueue** initially containing the elements of the given collection, added **in traversal order** of the collection's iterator.

## 一般方法

(在尾部添加，头部获取，FIFO)

### 添加元素add、offer、put

boolean add(E e)

Inserts the specified element **at the tail of this queue**.

boolean offer(E e)

Inserts the specified element at the tail of this queue.

boolean offer(E e, long timeout, TimeUnit unit)

Inserts the specified element at the tail of this queue.

void **put**(E e)

Inserts the specified element at the tail of this queue.

### 获取元素

E peek()

Retrieves(检索获取), 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.

E poll(long timeout, TimeUnit unit)

Retrieves and removes the head of this queue, waiting up to the specified wait time if necessary for an element to become available.

E take()

Retrieves and removes the head of this queue, waiting if necessary until an element becomes available.

### remove

boolean remove(Object o)

Removes a single instance of the specified element from this queue, if it is present.

### size、remainingCapacity、isEmpty

int size()

Returns the number of elements in this queue.

int remainingCapacity()

Always returns Integer.MAX\_VALUE because a LinkedTransferQueue is not capacity constrained.

boolean isEmpty()

Returns true if this queue contains no elements.

### 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.

### getWaitingConsumerCount

int getWaitingConsumerCount()

Returns an estimate of the number of consumers waiting to receive elements via BlockingQueue.take() or timed poll.

### hasWaitingConsumer

boolean hasWaitingConsumer()

Returns true if there is at least one consumer waiting to receive an element via BlockingQueue.take() or timed poll.

### transfer 、tryTransfer

void transfer(E e)

Transfers the element to a consumer, waiting if necessary to do so.

boolean tryTransfer(E e)

Transfers the element to a waiting consumer immediately, if possible.

boolean tryTransfer(E e, long timeout, TimeUnit unit)

Transfers the element to a consumer if it is possible to do so before the timeout elapses.

### contains

boolean contains(Object o)

Returns true if this queue contains the specified element.

### Spliterator、iterator

Iterator<E> iterator()

Returns an iterator over the elements in this queue in proper sequence.

Spliterator<E> spliterator()

Returns a Spliterator over the elements in this queue.