java.util.concurrent

Class Exchanger<V>

java.lang.Object
    java.util.concurrent.Exchanger<V>

Type Parameters:

`V` - The type of objects that may be exchanged

```
public class Exchanger<V>
extends Object
```

A synchronization point at which threads can pair and swap elements within pairs. Each thread presents some object on entry to the `exchange` method, matches with a partner thread, and receives its partner's object on return. An Exchanger may be viewed as a bidirectional form of a `SynchronousQueue`. Exchangers may be useful in applications such as genetic algorithms and pipeline designs.

Sample Usage: Here are the highlights of a class that uses an `Exchanger` to swap buffers between threads so that the thread filling the buffer gets a freshly emptied one when it needs it, handing off the filled one to the thread emptying the buffer.

```
class FillAndEmpty {
   Exchanger<DataBuffer> exchanger = new Exchanger<DataBuffer>();
   DataBuffer initialEmptyBuffer = ... a made-up type
   DataBuffer initialFullBuffer = ...

   class FillingLoop implements Runnable {
     public void run() {
       DataBuffer currentBuffer = initialEmptyBuffer;
       try {
         while (currentBuffer != null) {
           addToBuffer(currentBuffer);
           if (currentBuffer.isFull())
             currentBuffer = exchanger.exchange(currentBuffer);
         }
       } catch (InterruptedException ex) { ... handle ... }
     }
   }

   class EmptyingLoop implements Runnable {
     public void run() {
       DataBuffer currentBuffer = initialFullBuffer;
       try {
         while (currentBuffer != null) {
           takeFromBuffer(currentBuffer);
           if (currentBuffer.isEmpty())
             currentBuffer = exchanger.exchange(currentBuffer);
         }
       } catch (InterruptedException ex) { ... handle ...}
     }
   }

   void start() {
     new Thread(new FillingLoop()).start();
     new Thread(new EmptyingLoop()).start();
```

```
    }
}
```

Memory consistency effects: For each pair of threads that successfully exchange objects via an `Exchanger`, actions prior to the `exchange()` in each thread *happen-before* those subsequent to a return from the corresponding `exchange()` in the other thread.

Since:

1.5

Constructor Summary

Constructors

Constructor and Description
`Exchanger``()` Creates a new Exchanger.

Method Summary

Methods

Modifier and Type	Method and Description
`V`	**`exchange`**`(V x)` Waits for another thread to arrive at this exchange point (unless the current thread is **interrupted**), and then transfers the given object to it, receiving its object in return.
`V`	**`exchange`**`(V x, long timeout, TimeUnit unit)` Waits for another thread to arrive at this exchange point (unless the current thread is **interrupted** or the specified waiting time elapses), and then transfers the given object to it, receiving its object in return.

Methods inherited from class java.lang.Object

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

Constructor Detail

Exchanger

```
public Exchanger()
```

Creates a new Exchanger.

Method Detail

exchange

```
public V exchange(V x)
           throws InterruptedException
```

Waits for another thread to arrive at this exchange point (unless the current thread is interrupted), and then transfers the given object to it, receiving its object in return.

If another thread is already waiting at the exchange point then it is resumed for thread scheduling purposes and receives the object passed in by the current thread. The current thread returns immediately, receiving the object passed to the exchange by that other thread.

If no other thread is already waiting at the exchange then the current thread is disabled for thread scheduling purposes and lies dormant until one of two things happens:

- Some other thread enters the exchange; or
- Some other thread interrupts the current thread.

If the current thread:

- has its interrupted status set on entry to this method; or
- is interrupted while waiting for the exchange,

then `InterruptedException` is thrown and the current thread's interrupted status is cleared.

Parameters:

`x` - the object to exchange

Returns:

the object provided by the other thread

Throws:

`InterruptedException` - if the current thread was interrupted while waiting

exchange

```
public V exchange(V x,
         long timeout,
         TimeUnit unit)
           throws InterruptedException,
                  TimeoutException
```

Waits for another thread to arrive at this exchange point (unless the current thread is interrupted or the specified waiting time elapses), and then transfers the given object to it, receiving its object in return.

If another thread is already waiting at the exchange point then it is resumed for thread scheduling purposes and receives the object passed in by the current thread. The current thread returns immediately, receiving the object passed to the exchange by that other thread.

If no other thread is already waiting at the exchange then the current thread is disabled for thread scheduling purposes and lies dormant until one of three things happens:

- Some other thread enters the exchange; or
- Some other thread interrupts the current thread; or
- The specified waiting time elapses.

If the current thread:

- has its interrupted status set on entry to this method; or
- is interrupted while waiting for the exchange,

then `InterruptedException` is thrown and the current thread's interrupted status is cleared.

If the specified waiting time elapses then `TimeoutException` is thrown. If the time is less than or equal to zero, the method will not wait at all.

Parameters:

`x` - the object to exchange

`timeout` - the maximum time to wait

`unit` - the time unit of the `timeout` argument

Returns:

the object provided by the other thread

Throws: