Module java.base

Package java.util.concurrent

Class Exchanger<V>
java.util.concurrent.Exchanger<V>

Type Parameters:

V - The type of objects that may be exchanged

Since: 1.5

public class **Exchanger<V>**

An Exchanger allows two threads to exchange objects at a rendezvous point, and is useful in several pipeline designs.

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.

What is Exchanger?

Answer: - Exchanger is designed to simplify the exchange of data between two threads.

The operation of an Exchanger is astoundingly simple: it simply waits until two separate threads call its exchange () method. When that occurs, it exchanges the data supplied by the threads. This mechanism is both elegant and easy to use.

Uses for Exchanger are easy to imagine. For example, one thread might prepare a buffer for receiving information over a network connection. Another thread might fill that buffer with the information from the connection. The two threads work together so that each time a new buffer is needed, an exchange is made.

Methods of Exchanger

- **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.

What is the difference between exchange (V x) and exchange (V x, long timeout, TimeUnit unit)?

Answer: -

V exchange (V x) throws InterruptedException	**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), and then transfers the given object to it, receiving its object in return.	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 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	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.	• 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 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.