- CompletionService Interface
- ThreadLocal
- ThreadLocalRandom
- CountDownLatch
- CyclicBarrier
- Concurrent Collections
- ConcurrentHashMap
- ConcurrentModificationException

Practice Mock Interview

Exchanger

Guide to using the Exchanger<V> class.

*If you are interviewing, consider buying our **number#1** course for **Java Multithreading Interviews**.*

Overview

As the name indicates, `Exchanger` is a construct that can be used to make bidirectional transfers of objects between two threads. Each thread invokes the `exchange()` method with an item the thread wants to exchange with the other thread. A thread blocks when making the `exchange()` call until the other thread invokes the `exchange()` method.

In case of multiple threads, it is not possible for a thread to choose its partner to exchange objects with. Note that the `exchange()` must be invoked an even number of times during the course of a program to ensure no thread remains blocked when the program exits.

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.

Example

In the simple example below, we have two threads that exchange `String` objects with each other. Each thread shares its name with the other thread. The output of the program shows the string that each thread receives from the other thread.

course of a program to ensure no thread remains blocked when the program exits.

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.

Example

In the simple example below, we have two threads that exchange `String` objects with each other. Each thread shares its name with the other thread. The output of the program shows the string that each thread receives from the other thread.

Java

main.java

Saved

```java
import java.util.concurrent.*;

class Demonstration {
    public static void main( String args[] ) throws InterruptedException {
        // create an exchanger object
        Exchanger<String> exchanger = new Exchanger<>();

        // create an executor service
        ExecutorService executorService = Executors.newFixedThreadPool(5);

        try {
            // submit the first task
```

Run

In the second example, we have ten threads exchange their names. The very first thread blocks on the invocation to `exchange()`. Next, the second thread comes along and is paired with the first blocked thread and the two exchange strings and move on. This pattern continues with the third thread being paired with the fourth, the fifth with the sixth, so on and so forth.

Note that we can't have the first thread to pair with, say the seventh thread, i.e. the developer can't influence how threads are paired.

Java

Press + to interact

main.java

Saved

```java
import java.util.concurrent.*;

class Demonstration {

    static Exchanger<String> exchanger = new Exchanger<>();

    public static void main( String args[] ) throws InterruptedException {
        ExecutorService executorService = Executors.newFixedThreadPool(10);

        try {
            for (int i = 0; i < 10; i++) {
                executorService.submit(new Runnable() {
```

Run

The output of the above widget shows which thread gets paired with which other thread for the exchange of objects.

← Back lesson

AtomicMarkableReference

Mark As Completed Next →

Phaser