

48% completed

Search Course

Java Concurrency Reference

- Setting-up Threads
- Basic Thread Handling
- Executor Framework
- Executor Implementations
- Thread Pools
- Types of Thread Pools
- An Example: Timer vs ScheduledThreadPool
- ThreadPoolExecutor
- Callable Interface
- Future Interface
- CompletionService Interface
- ThreadLocal
- ThreadLocalRandom
- CountDownLatch
- CyclicBarrier
- Concurrent Collections
- ConcurrentHashMap
- ConcurrentModificationException

Practice Mock Interview

Java Multithreading for Senior Engineering Interviews / ... / CompletionService Interface

# CompletionService Interface

This lesson talks about how to batch multiple tasks together

## CompletionService Interface

In the previous lesson we discussed how tasks can be submitted to executors but imagine a scenario where you want to submit hundreds or thousands of tasks. You'll retrieve the future objects returned from the submit calls and then poll all of them in a loop to check which one is done and then take appropriate action. Java offers a better way to address this use case through the `CompletionService` interface. You can use the `ExecutorCompletionService` as a concrete implementation of the interface.

The completion service is a combination of a blocking queue and an executor. Tasks are submitted to the queue and then the queue can be polled for completed tasks. The service exposes two methods, one `poll` which returns null if no task is completed or none were submitted and two `take` which blocks till a completed task is available.

Below is an example program that demonstrates the use of completion service.

Java

Press `⌘` + `⌘` to interact

main.java

```
1 import java.util.Random;
2 import java.util.concurrent.ExecutorCompletionService;
3 import java.util.concurrent.ExecutorService;
4 import java.util.concurrent.Executors;
5 import java.util.concurrent.Future;
6
7
8 class Demonstration {
9
10     static Random random = new Random(System.currentTimeMillis());
11
12     public static void main( String args[] ) throws Exception {
```

Run

Back lesson

☒ Mark As Completed

Next

Future Interface

ThreadLocal