

72% 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 / ... / ExecutionException

ExecutionException

Guide to working with ExecutionException.

If you are interviewing, consider buying our number#1 course for [Java Multithreading Interviews](#).

Overview

`ExecutionException` is a checked exception that extends the `Exception` class. This exception is thrown by an instance of `FutureTask` that encounters an `Exception` or `Error` (both derives of `Throwable`) that remain unhandled by user code and, subsequently an attempt is made to retrieve the result of such a task.

Example

Consider the program below that has a task submitted to the `ExecutorService`. The task simply throws a runtime exception to simulate failure. When we attempt to retrieve the result of the task, the snippet `futureTask.get()` throws `ExecutionException`.

```
1 import java.util.concurrent.*;
2
3 class Demonstration {
4     public static void main( String args[] ) {
5         ExecutorService es = Executors.newFixedThreadPool(5);
6
7         // Create a FutureTask, which takes in an instance of Callable
8         FutureTask<Integer> futureTask = new FutureTask<>(new Callable<Integer>() {
9             @Override
10             public Integer call() throws Exception {
11                 // The task simulates encountering an exception by throwing one.
12                 throw new RuntimeException("runtime issue");
13             }
14         });
15
16         try {
17             Integer result = futureTask.get();
18         } catch (ExecutionException e) {
19             e.printStackTrace();
20         }
21     }
22 }
```

In the above example, if you comment out **line#23** and the associated catch clause for `ExecutionException`, upon re-run of the program you'll notice that it doesn't throw `ExecutionException` even though the task throws a runtime exception. A programming oversight to retrieve the result of submitted task that fail can falsely lead the program to exit successfully.

Finally, if we replace **line#12** with the snippet `throw new Error();`, we'll still observe the `ExecutionException` being throw upon retrieving the result of the program.

Back lesson

Mark As Completed

Next

CancellationException

RejectedExecutionException