ExecutionException

Guide to working with ExecutionException.

We'll cover the following

If you are interviewing, consider buying our number#1 course for Java Multithreading Interviews (https://bit.ly/2QfKXCK).

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 derivates 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`.

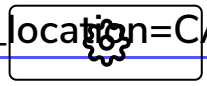
```
1  import java.util.concurrent.*;
2
```

```
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<Integ
9              @Override
10             public Integer call() throws Exception {
11                 // The task simulates encountering an exception by throwing
12                 throw new RuntimeException("runtime issue");
13             }
14         });
15
16         try {
17             es.submit(futureTask);
18
19             // wait a while for the task
20             Thread.sleep(300);
21
22             // Attempt to get the result of the task
23             futureTask.get();
24         } catch (ExecutionException e) {
25             // You should see the following line print in the console.
26             System.out.println("ExceutionException thrown by program.");
27         } catch (InterruptedException ie) {
28             // we can ignore InterruptedException for demo purposes
```

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.

interviewing soon? We've partnered with Hired so that companies apply to y
utm_source=educative&utm_medium=lesson&utm_location=CA&utm_campa



Back

Next

CancellationException

Quiz 1

Mark as Completed

Report an Issue

Ask a Question (https://discuss.educative.io/tag/executionexception__java-concurrency-reference__java-multithreading-for-senior-engineering-interviews)