73% completed

Search Course

Java Concurrency Reference

Practice Mock Interview →

CountDownLatch
CyclicBarrier
Concurrent Collections
ConcurrentHashMap
ConcurrentModificationException

Practice Mock Interview →

RejectedExecutionException

Learn the causes of RejectedExecutionException being thrown.

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

Overview

Rather than creating individual threads for executing jobs in parallel, we can leverage Java's executor classes that abstract away thread scheduling and management from the user and leave the user to focus on submitting `Runnable` or `Callable` tasks/commands. Some of the classes implementing the `Executor` and `ExecutorService` interfaces are as follows:

- `AbstractExecutorService`
- `ForkJoinPool`
- `ScheduledThreadPoolExecutor`
- `ThreadPoolExecutor`

The `Executor` interface defines a method `execute` while the `ExecutorService` defines overloaded versions of the `submit` method. Both these methods throw the runtime exception `RejectedExecutionException` when implemented by the various executor classes. The `RejectedExecutionException` exception is thrown when a task can't be accepted by the executor for execution. Generally there are two scenarios when a task is rejected for execution: The executor service has already been shut down. When the executor service

The `Executor` interface defines a method `execute` while the `ExecutorService` defines overloaded versions of the `submit` method. Both these methods throw the runtime exception `RejectedExecutionException` when implemented by the various executor classes. The `RejectedExecutionException` exception is thrown when a task can't be accepted by the executor for execution. Generally there are two scenarios when a task is rejected for execution: The executor service has already been shut down. When the executor service
has exhausted resources and can't take on any more task submissions. For instance in the case of the `ThreadPoolExecutor`, the `RejectedExecutionException` is thrown when the executor service uses a queue with a defined maximum capacity and a defined maximum number of threads for the thread pool and both resources have reached capacity.

Example

Consider the program below that attempts to submit a task for execution to a fixed-size thread pool executor, after `shutdown()` has already been invoked. The executor throws the `RejectedExecutionException`.

Java

main.java Saved

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

class Demonstration {
    public static void main( String args[] ) {

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

        // shutdown the executor
        executor.shutdown();

        // attempt to execute a Runnable after the ExecutorService has been shutdown
```

Run

As a second example, consider the program below that uses an instance of `ThreadPoolExecutor` as the executor service to submit tasks for execution. We instantiate the instance with a thread pool size of 10 and a queue to hold submitted tasks of size 10 too. The first 10 tasks submitted are all executed by the 10 available threads and the queue remains empty. However, the next ten tasks submitted fill-up the queue and on submitting the 21st task the executor throws `RejectedExecutionException`.

Java Press ⇧ + ↵ to interact

main.java Saved

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

class Demonstration {

    public static void main( String args[] ) {

        // Create a ThreadPoolExecutor with maximum of 10 threads and a queue that can hold
        ExecutorService executorService = new ThreadPoolExecutor(5, 10, 1, TimeUnit.HOURS,
                new LinkedBlockingQueue<Runnable>(10));

```

```
11        // declar outside the try loop to determine which task gets rejected.
12        int i = 0;
```

Run

In the above program if you change the for loop limit from 21 to 20 on line#16, the program would run without experiencing any exceptions.

Back lesson
ExecutionException

Mark As Completed

Next
CompletionException