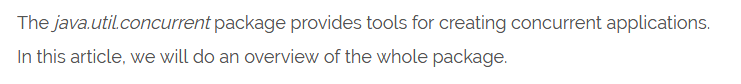
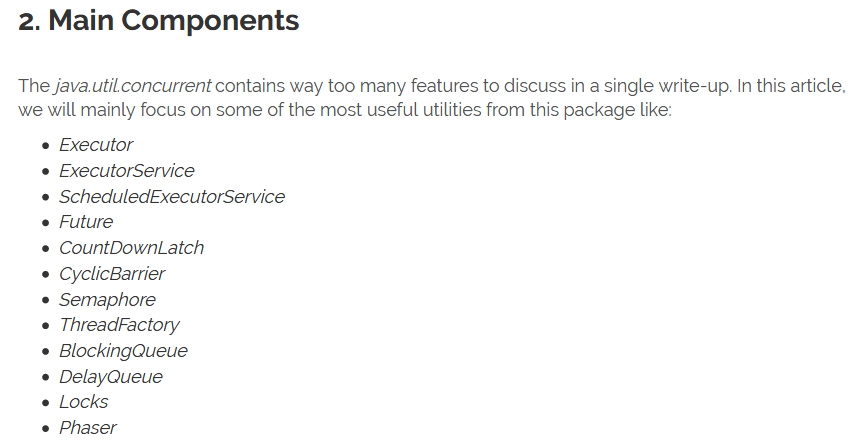
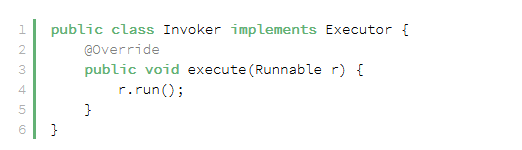
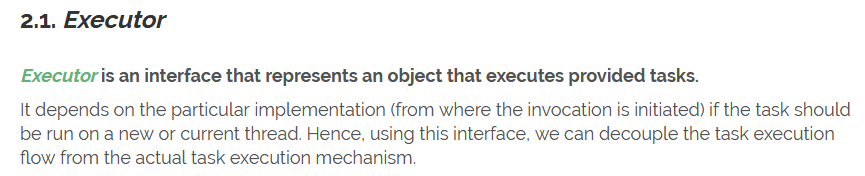
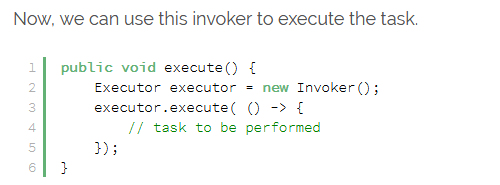
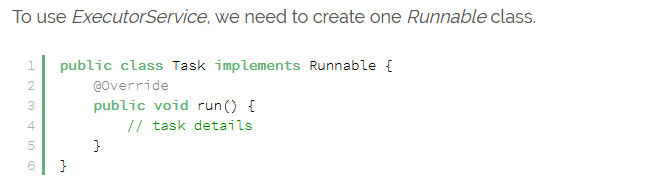
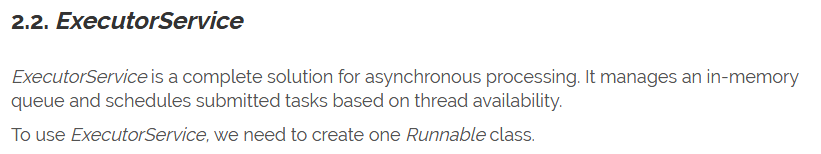
# Java Concurrency:

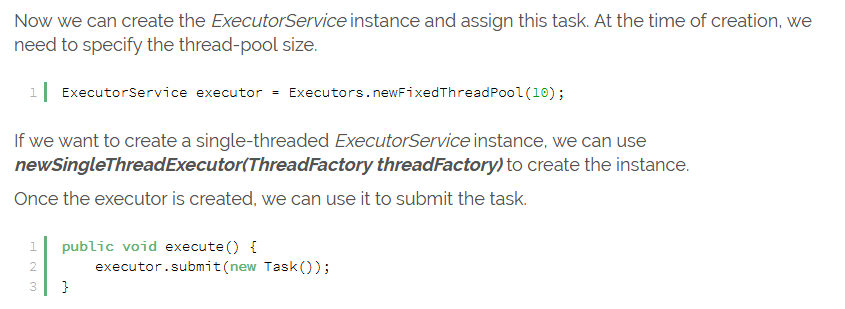






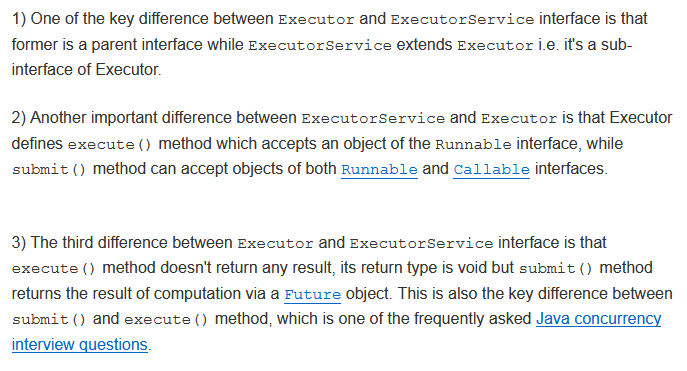


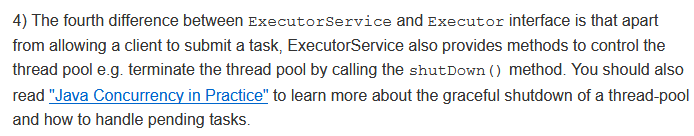


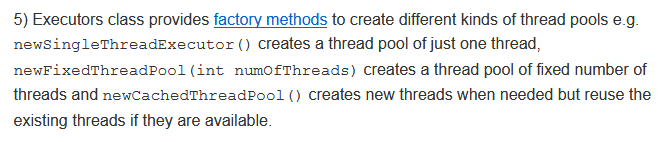


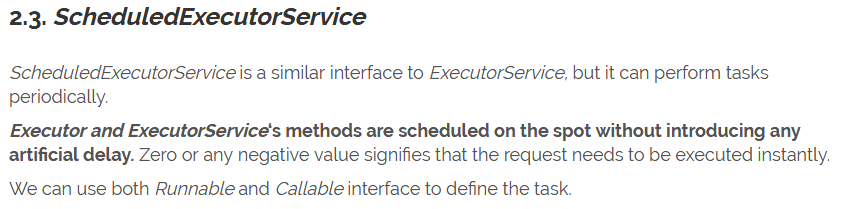
It also comes with two out-of-the-box execution termination methods. The first one is ***shutdown()***; it waits till the all submitted task finish executing. The other method is ***shutdownNow()*** whic*h* immediately terminates all the pending/executing tasks.

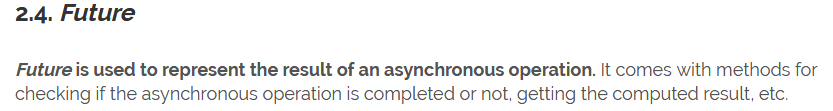
What is the difference between Executor and Executor Service?

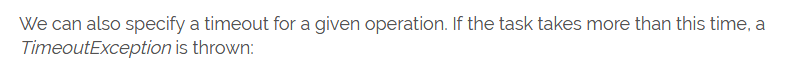


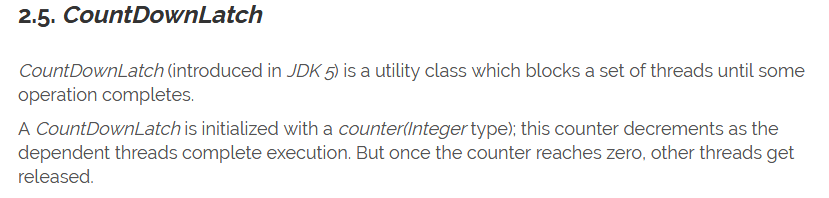












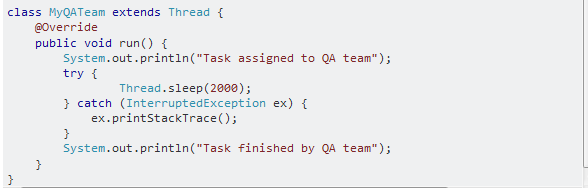
**Note:** It is similar to join in threads.

* **Use Case of CountDownLatch:**

Consider a scenario where manager divided modules between development teams (A and B) and he wants to assign it to QA team for testing only when both the teams completes their task.







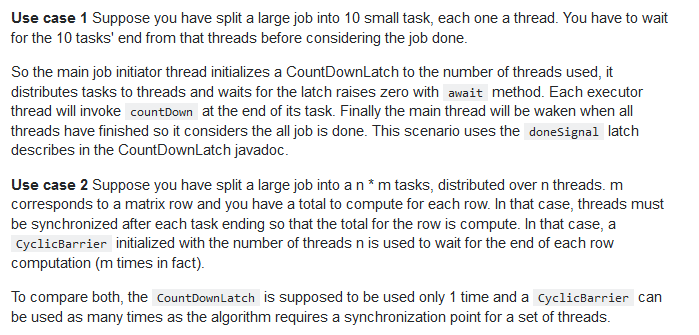
* **Disadvantage of CountDownLatch:**

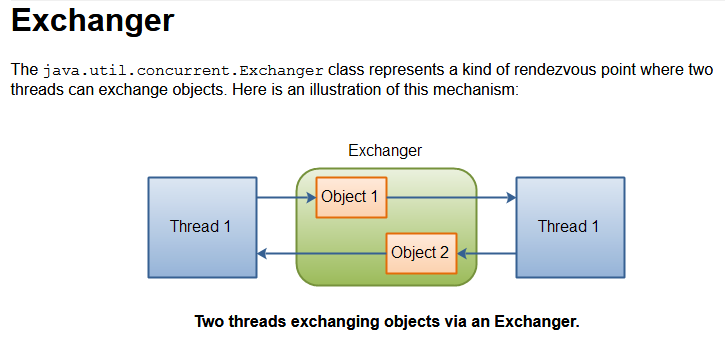
The disadvantage of CountDownLatch is that it's not reusable: once the count become zero it is no longer usable.

Difference between CountDownLatch and CyclicBarrier in Java Concurrency?

Both CyclicBarrier and CountDownLatch are used to implement a scenario where one Thread waits for one or more Thread to complete their job before starts processing but there is one Difference between CountDownLatch and CyclicBarrier in Java which separates them apart and that is, you cannot reuse same CountDownLatch instance once count reaches to zero and latch is open, on the other hand CyclicBarrier can be reused by resetting Barrier, Once barrier is broken.

* **Use Case of CyclicBarrier:**





* **Use Case of Exchanger:**

The Exchanger class is useful for passing data back and forth between two threads. e.g. Producer/Consumer. It is one of the least used Concurrency classes.