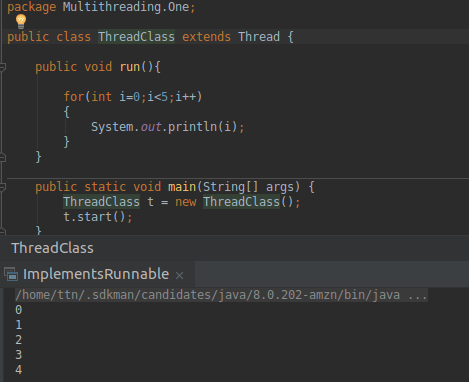
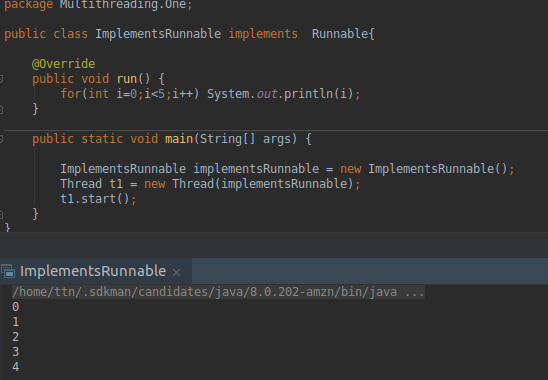
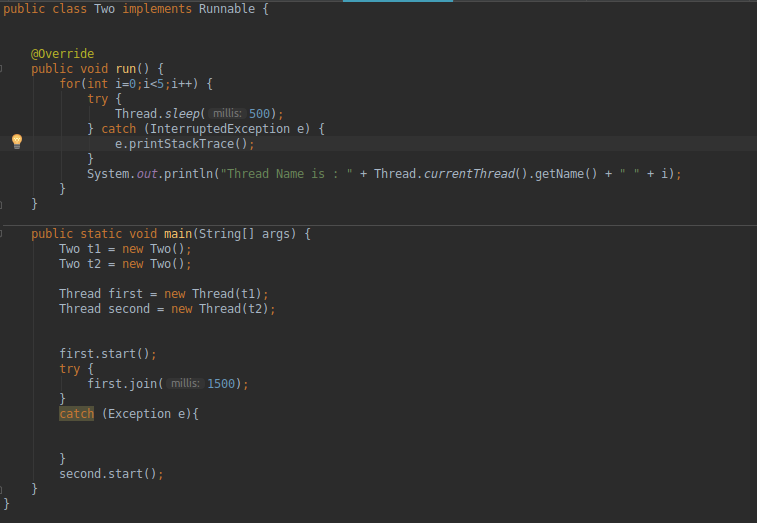
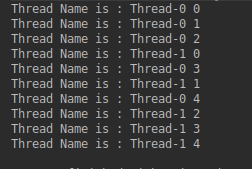
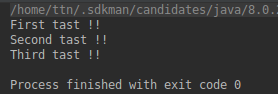
1. **Create and Run a Thread using Runnable Interface and Thread class.**

****

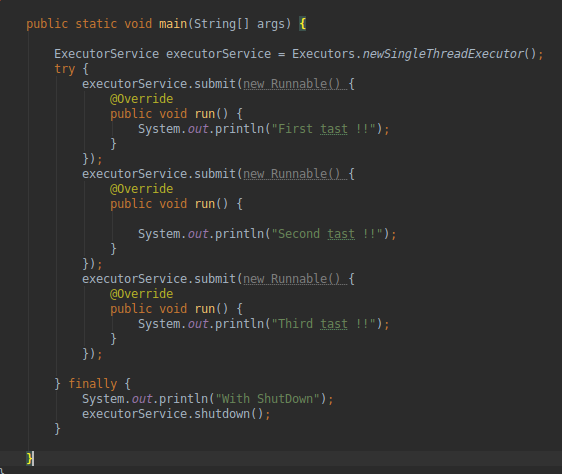
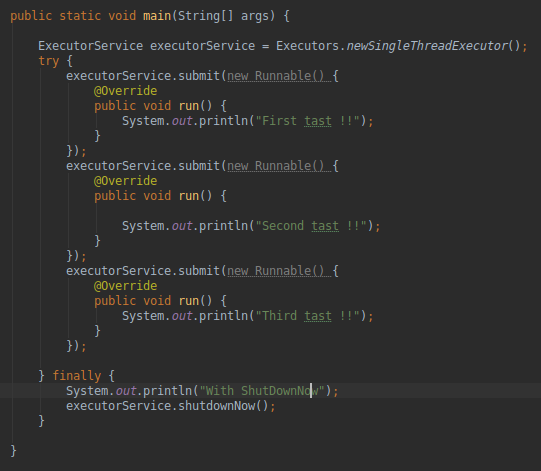
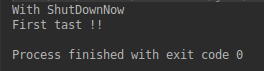
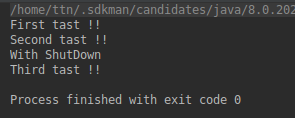
1. **Use sleep and join methods with thread.**

****

1. **Use a singleThreadExecutor to submit multiple threads.**

****

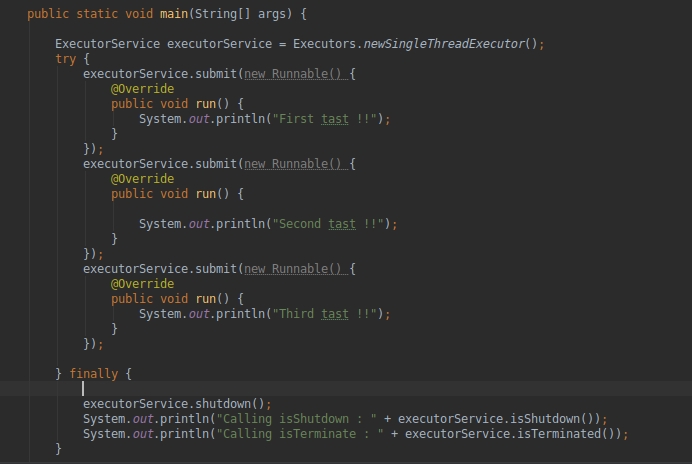
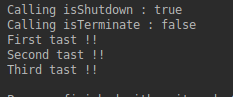
1. **Try shutdown() and shutdownNow() and observe the difference.**

****

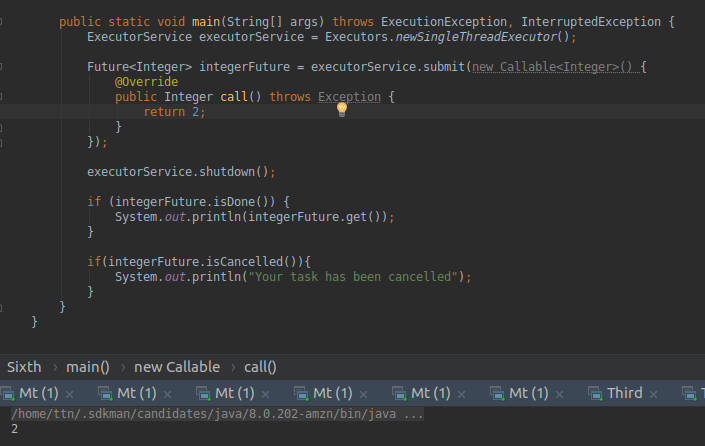
**shutdown() : Rejects New Task submitted to the thread executor while continuing to execute any previously submitted task.**

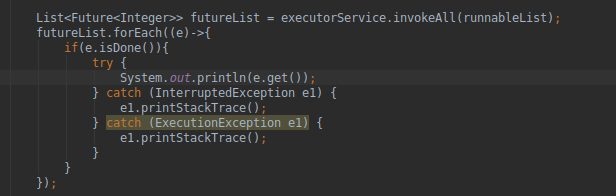
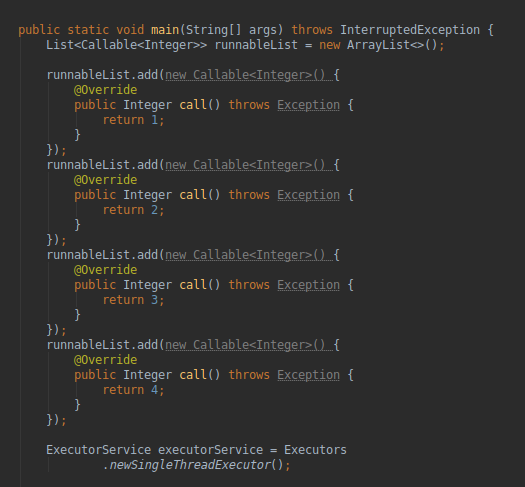
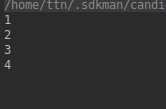
**shutdownNow() : shutdownNow will do the same as shutdown AND will try to cancel the already submitted tasks by interrupting the relevant threads. Note that if your tasks ignore the interruption, shutdownNow will behave exactly the same way as shutdown**

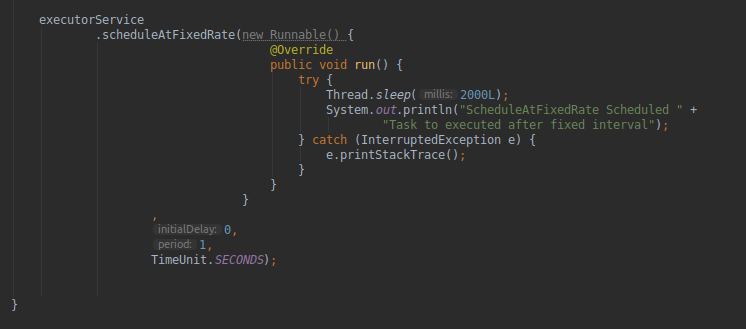
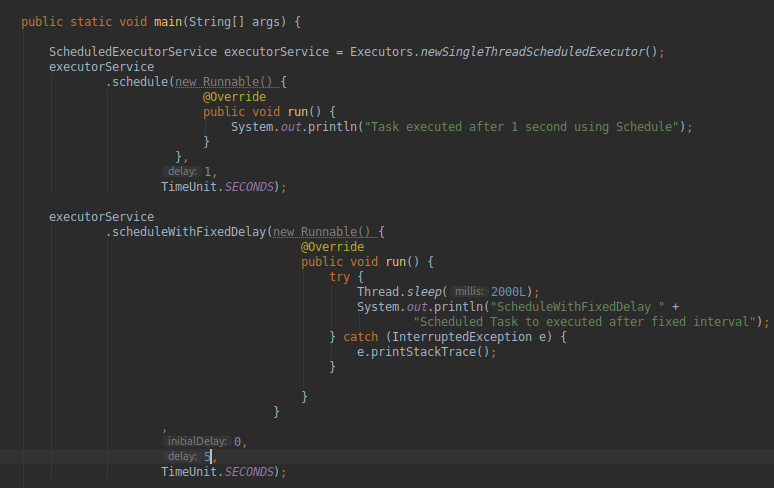
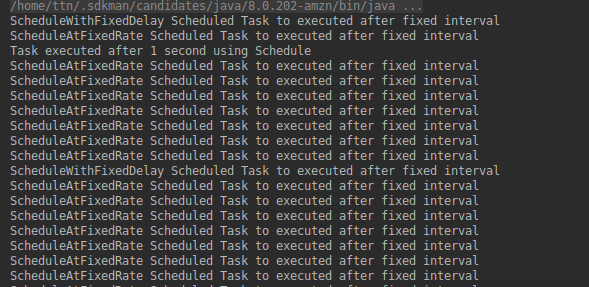
1. **Use isShutDown() and isTerminate() with ExecutorService.**

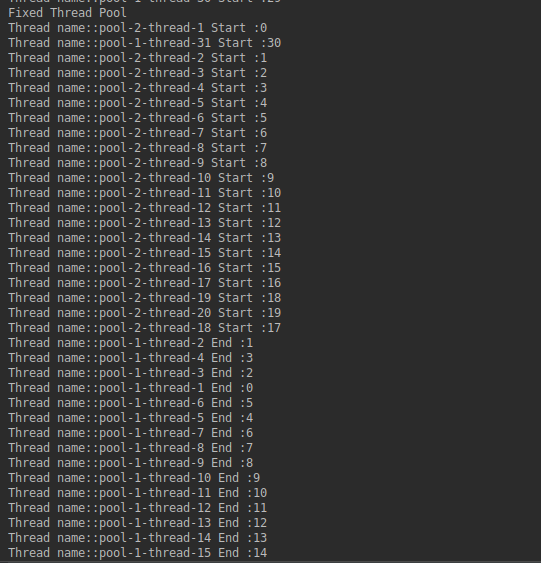
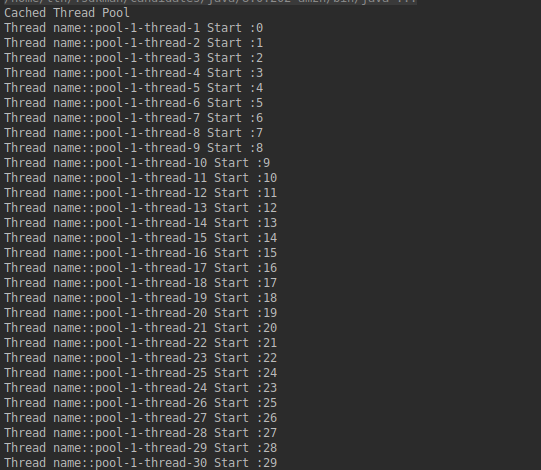
****

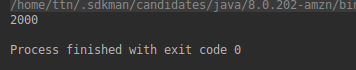
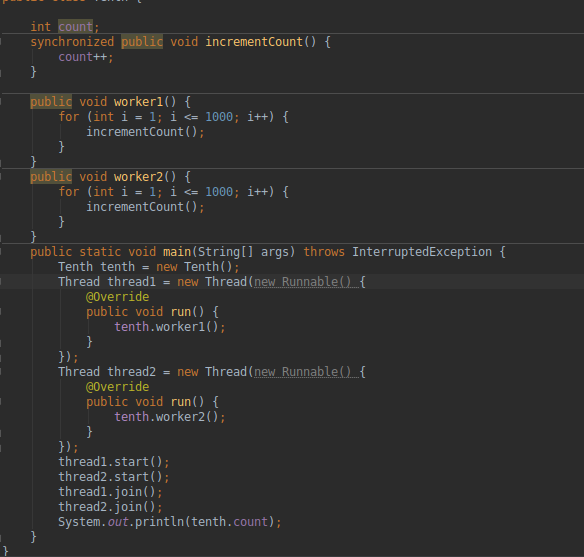
1. **Return a Future from ExecutorService by using callable and use get(), isDone(), isCancelled() with the Future object to know the status of task submitted.**

****

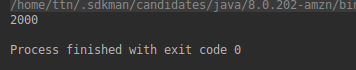
1. **Submit List of tasks to ExecutorService and wait for the completion of all the tasks. **
2. **Schedule task using schedule(), scheduleAtFixedRate() and scheduleAtFixedDelay()**

****

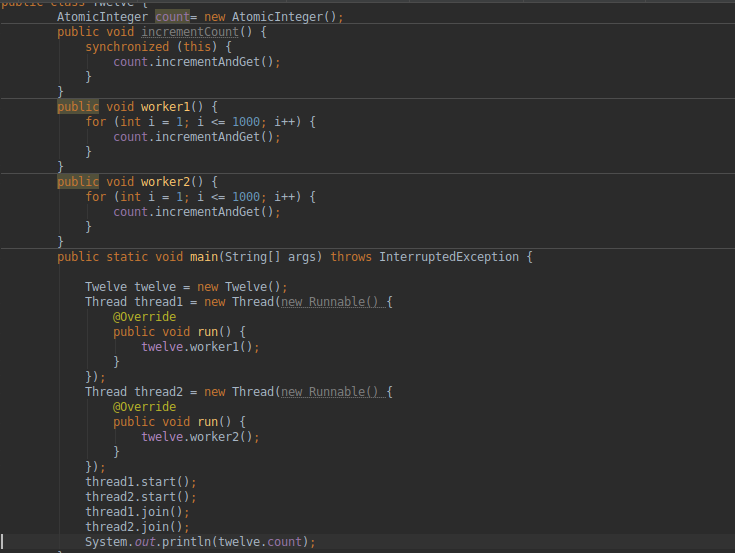
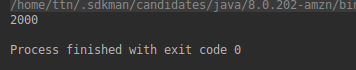
1. **Increase concurrency with Thread pools using newCachedThreadPool() and newFixedThreadPool().**
2. **Use Synchronize method to enable synchronization between multiple threads trying to access method at same time.**

****

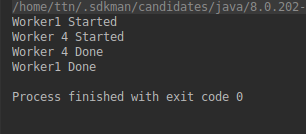
1. **Use Synchronize block to enable synchronization between multiple threads trying to access method at same time.**

****

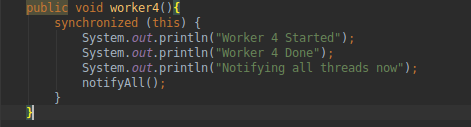
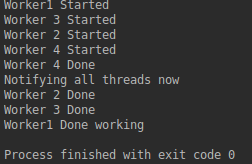
1. **Use Atomic Classes instead of Synchronize method and blocks.**

****

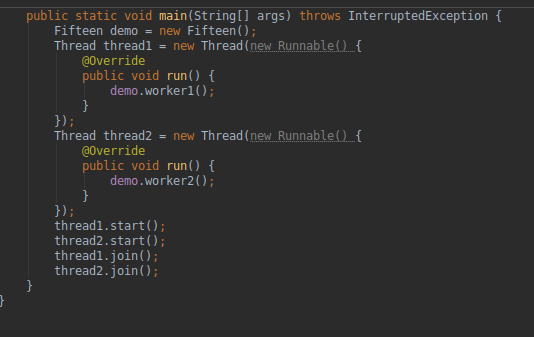
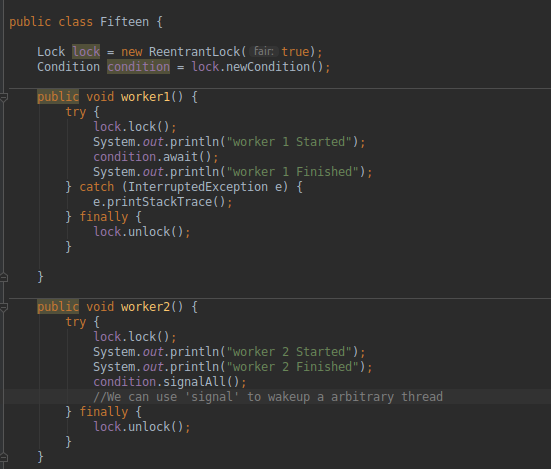
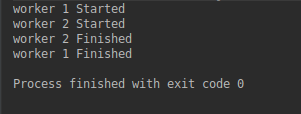
1. **Coordinate 2 threads using wait() and notify().**

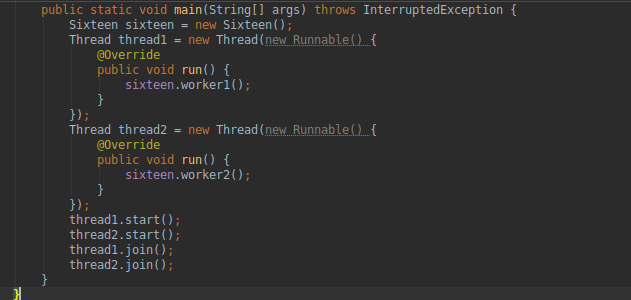
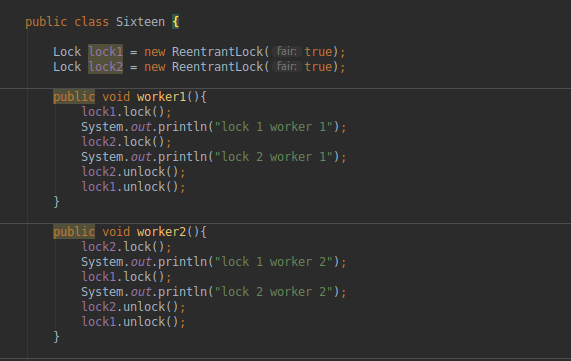
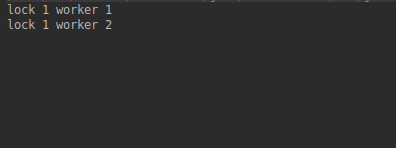
****

1. **Coordinate mulitple threads using wait() and notifyAll()**

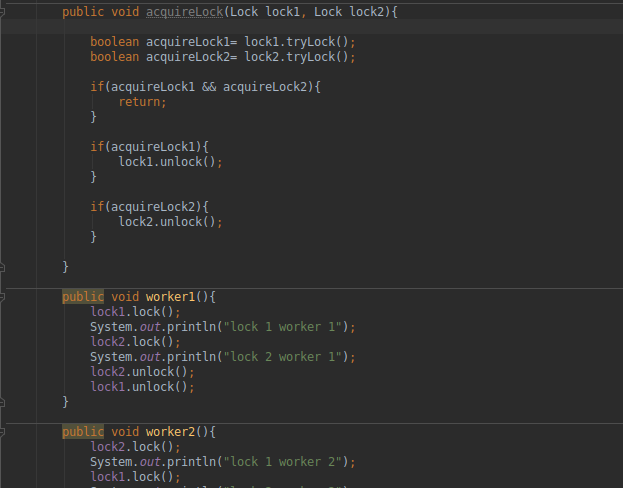
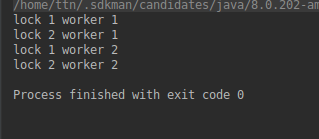
****

1. **Use Reentract lock for coordinating 2 threads with signal(), signalAll() and wait().**

****

1. **Create a deadlock and Resolve it using tryLock(). DEADLOCK**

**DEADLOCK RESOLVED ADDING “TRYLOCK”**

****