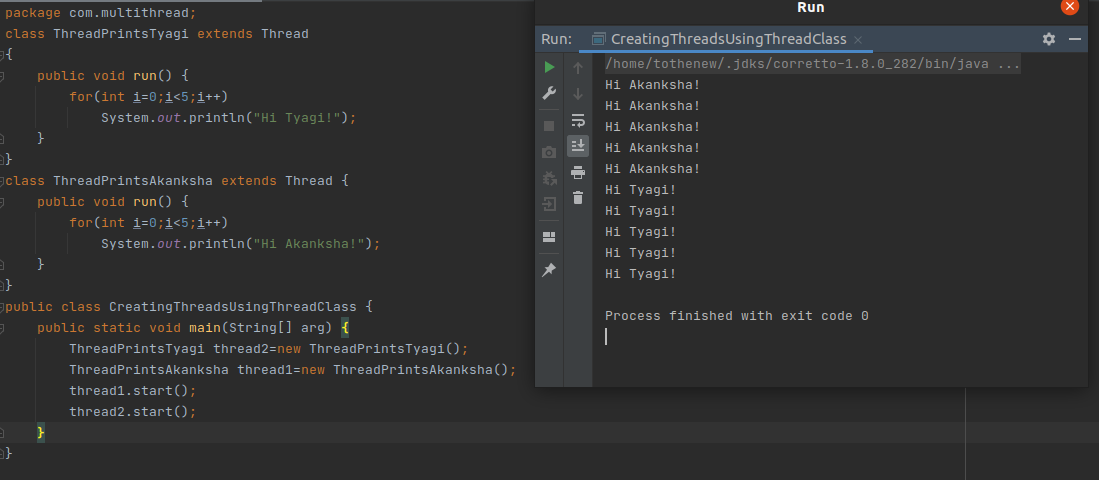
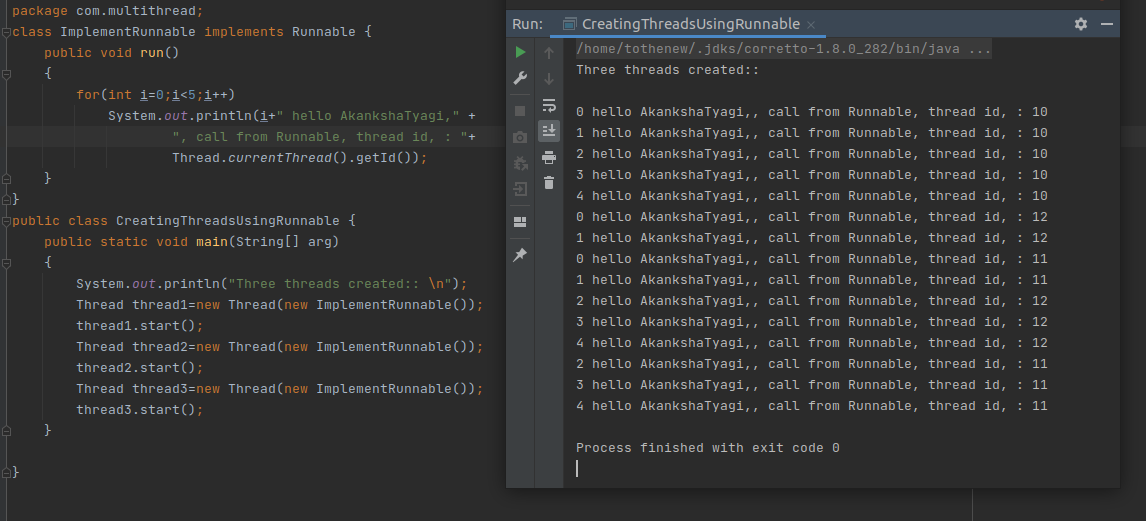
MultiThreading

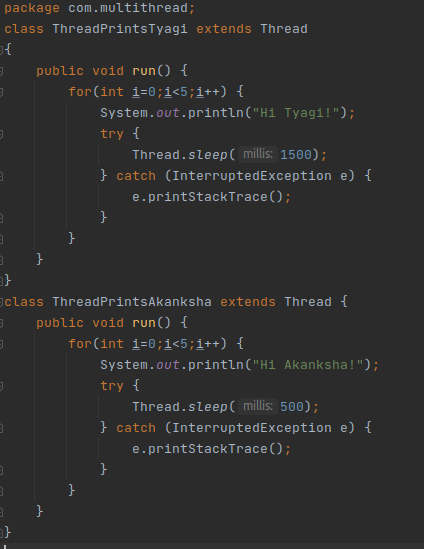
Q-1 Create and Run a Thread using Runnable Interface and Thread class.

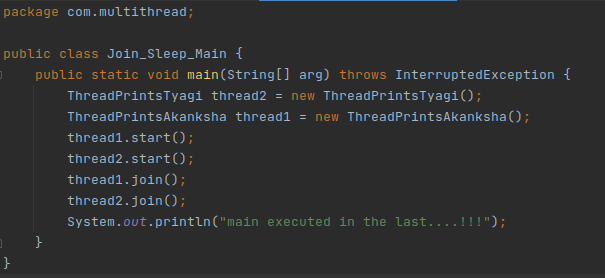
Extending Thread class

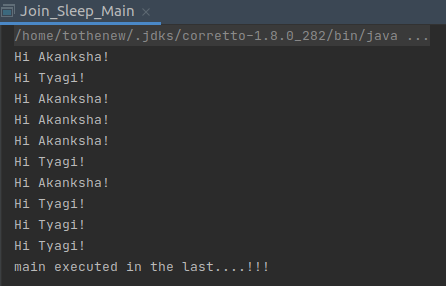


Implementing Runnable Interface

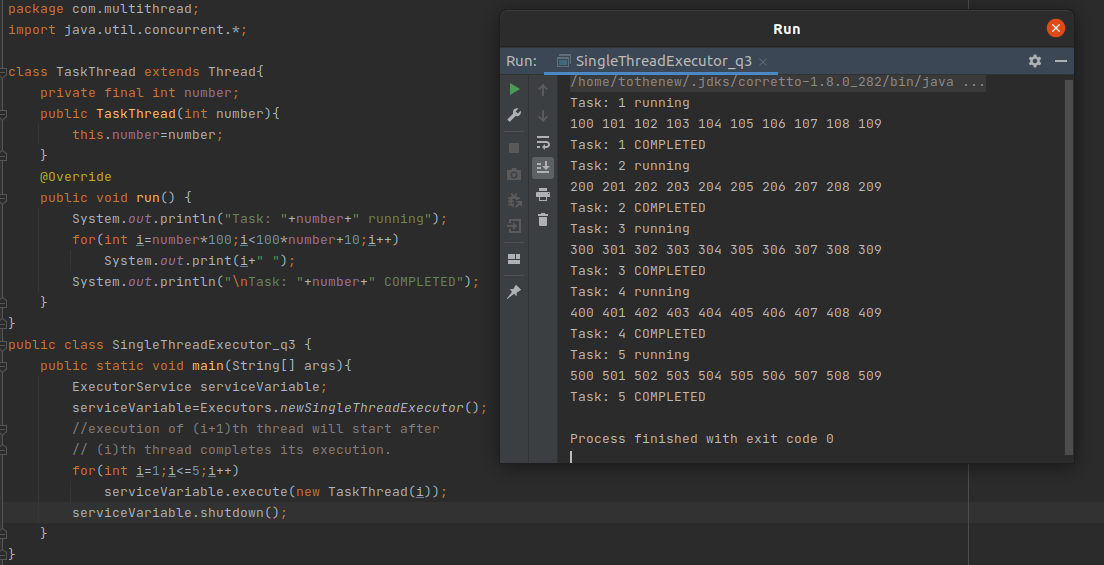
Q-2. Use sleep and join methods with thread.





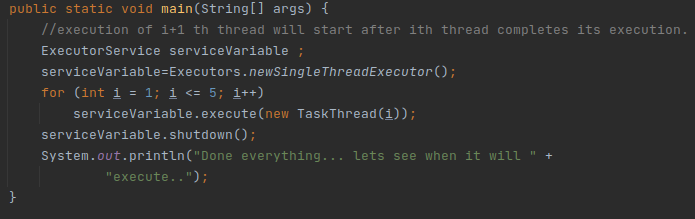


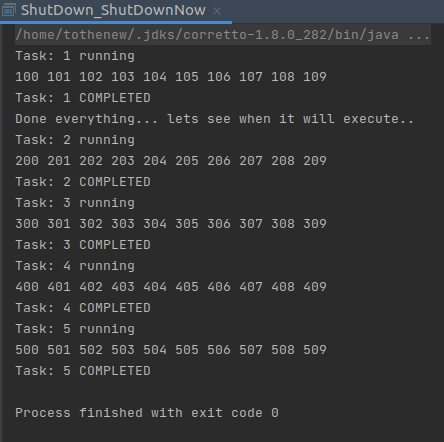
Q-3. Use a singleThreadExecutor to submit multiple threads.



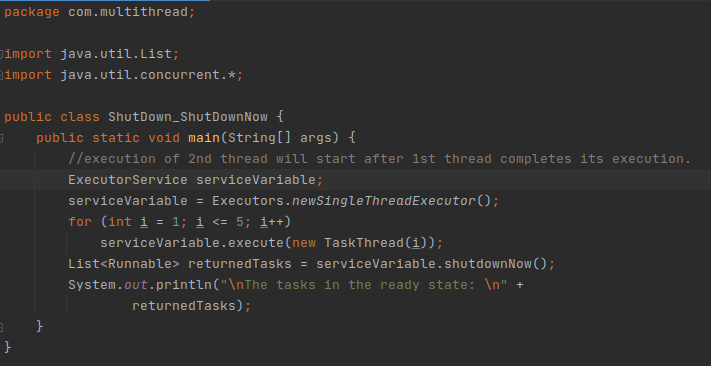
Q-4. Try shutdown() and shutdownNow() and observe the difference.

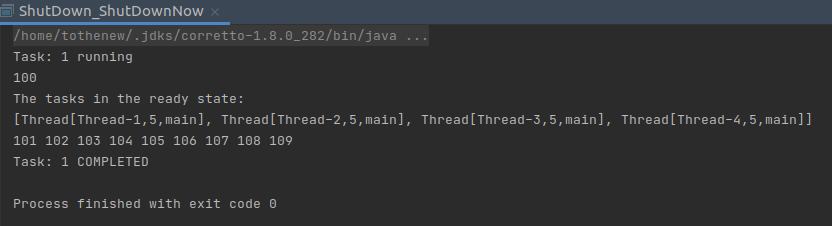
shutDown():



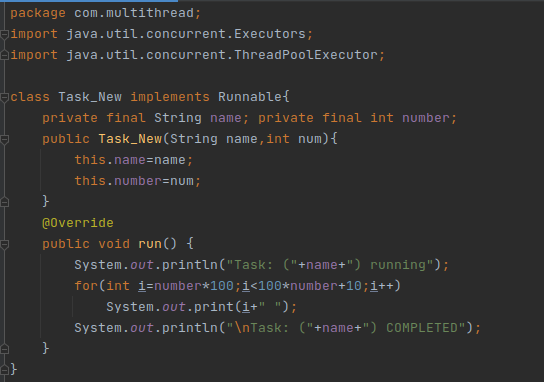


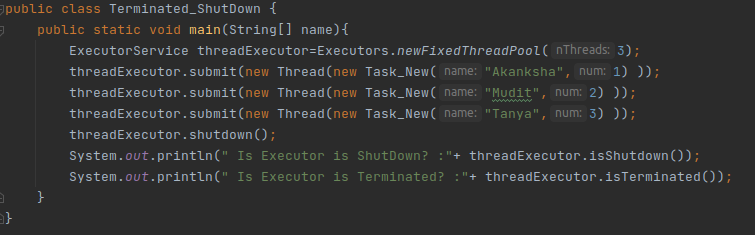
shutDownNow():

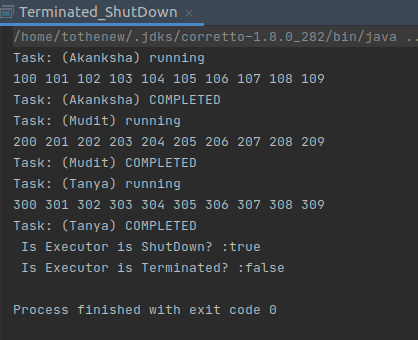




Q-5 Use isShutDown() and isTerminated() with ExecutorService.

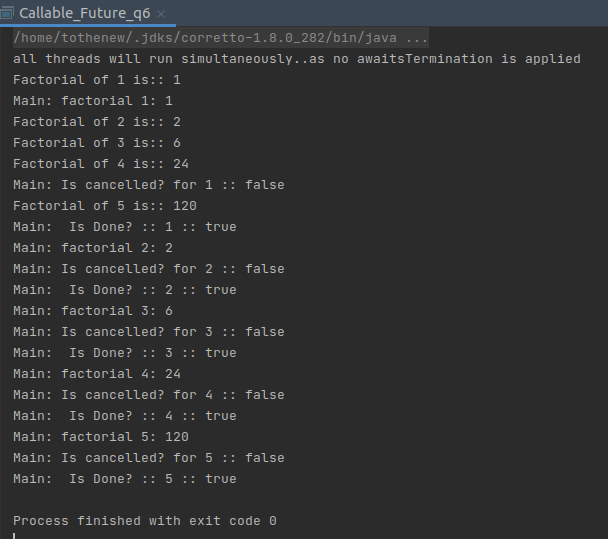
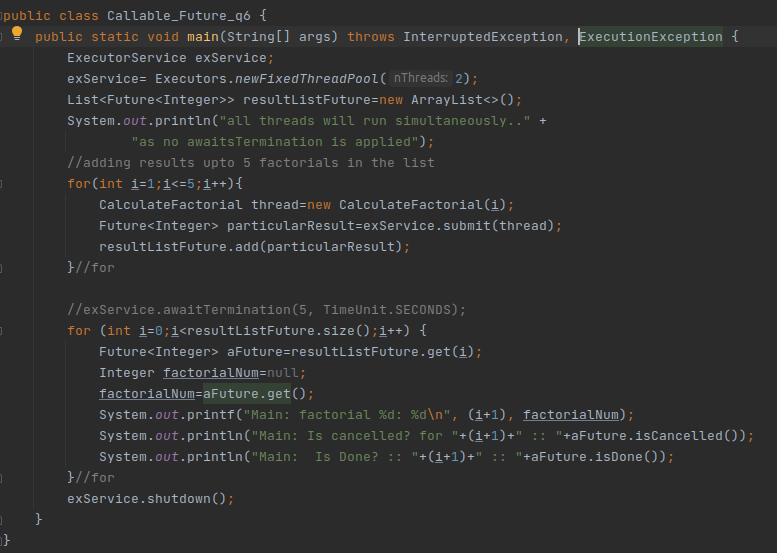






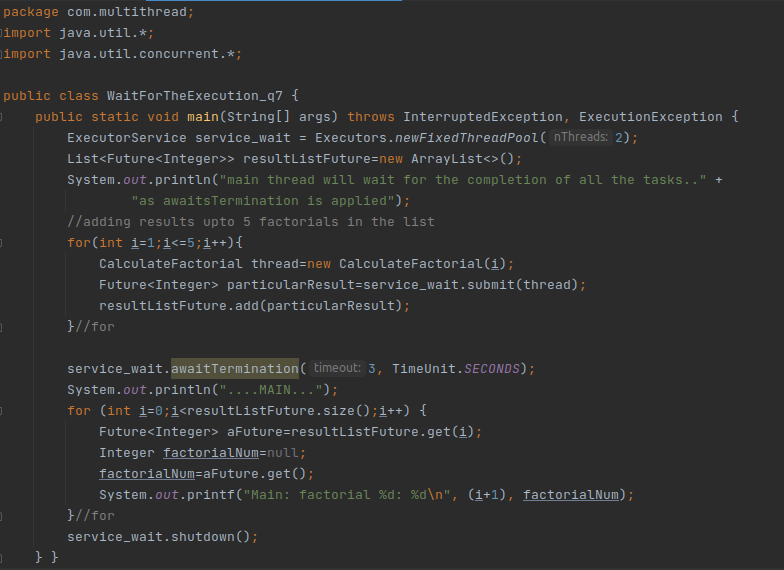
Q-6. Return a Future from ExecutorService by using callable and use get(), isDone(), isCancelled() with the Future object to know the status of task submitted.

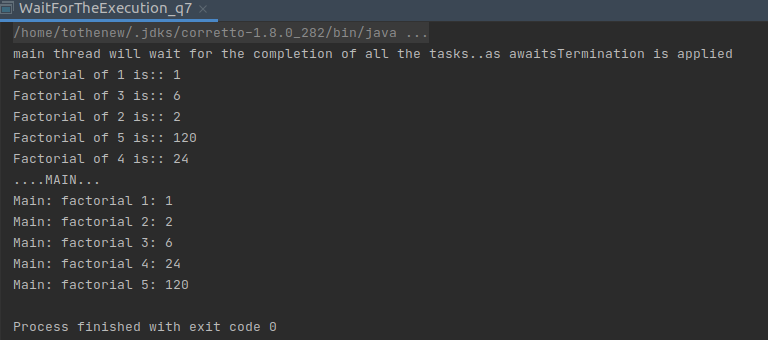




Q-7: Submit List of tasks to ExecutorService and wait for the completion of all the tasks.

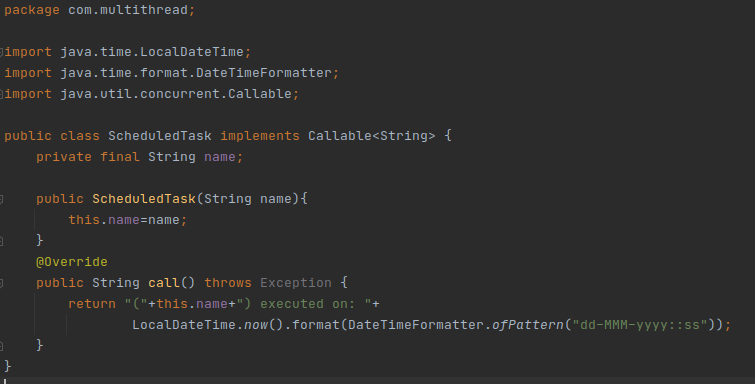


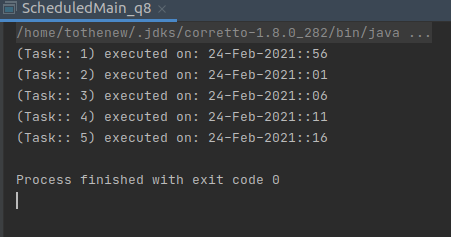
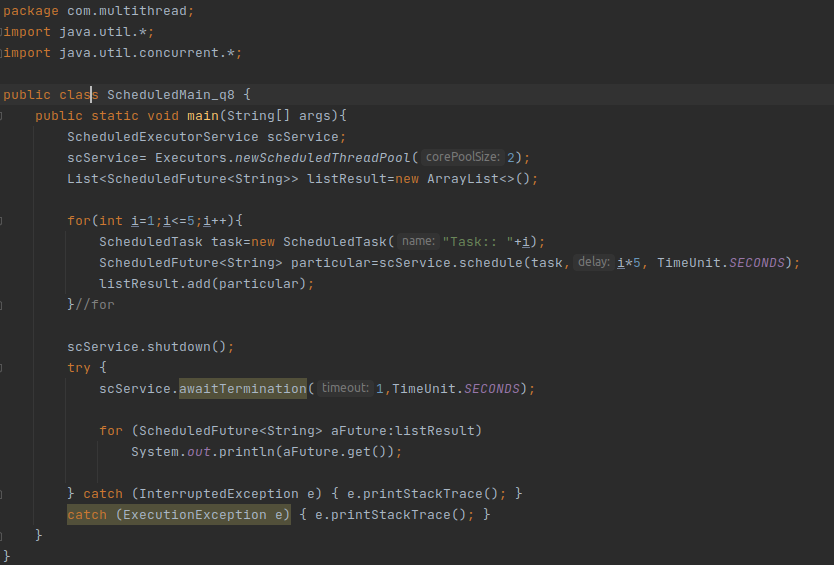


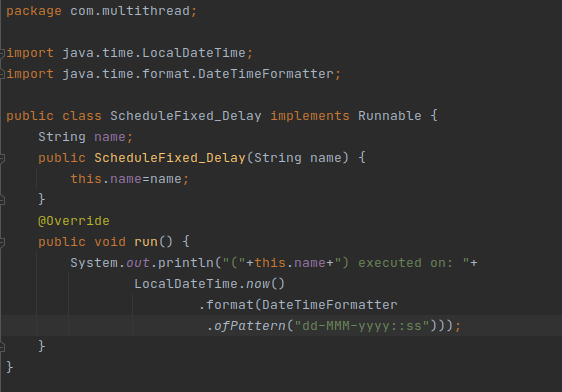


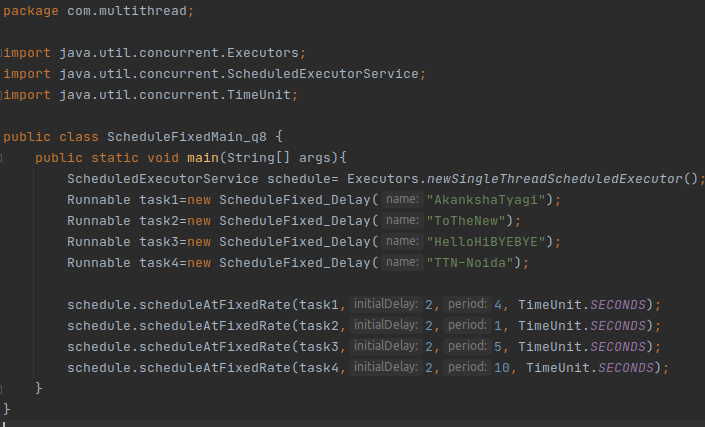
Q-8 Schedule task using schedule(), scheduleAtFixedRate() and scheduleAtFixedDelay()

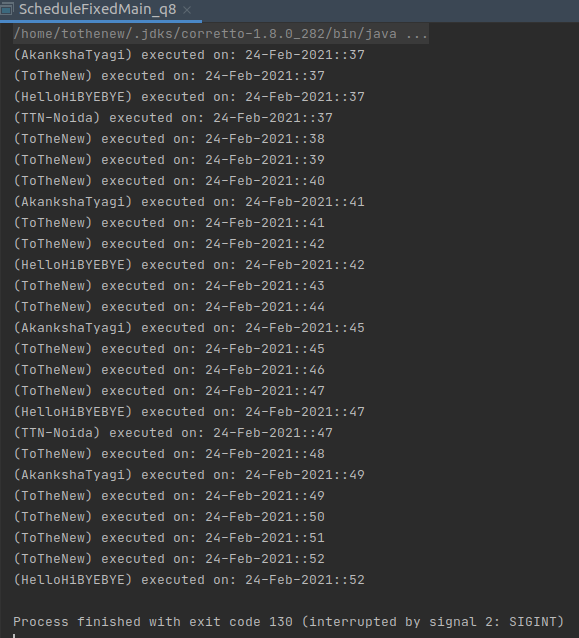
\_schedule\_

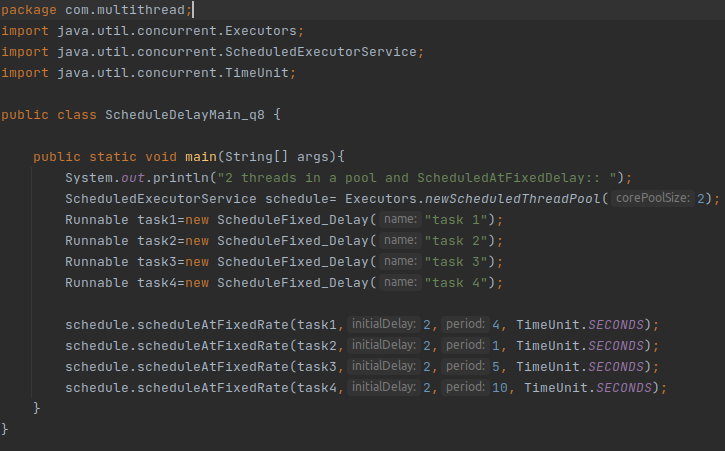


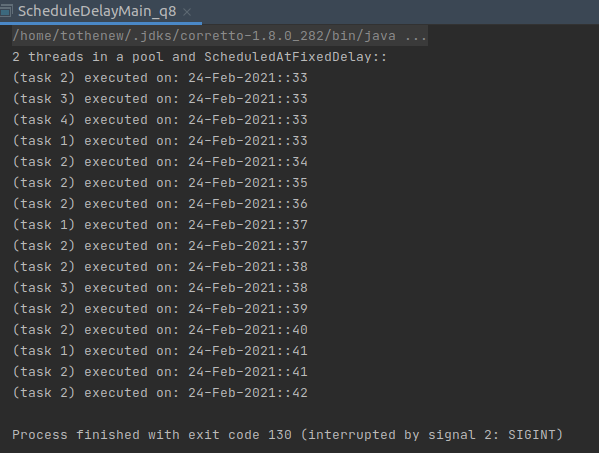


scheduleAtFixedRate():: 



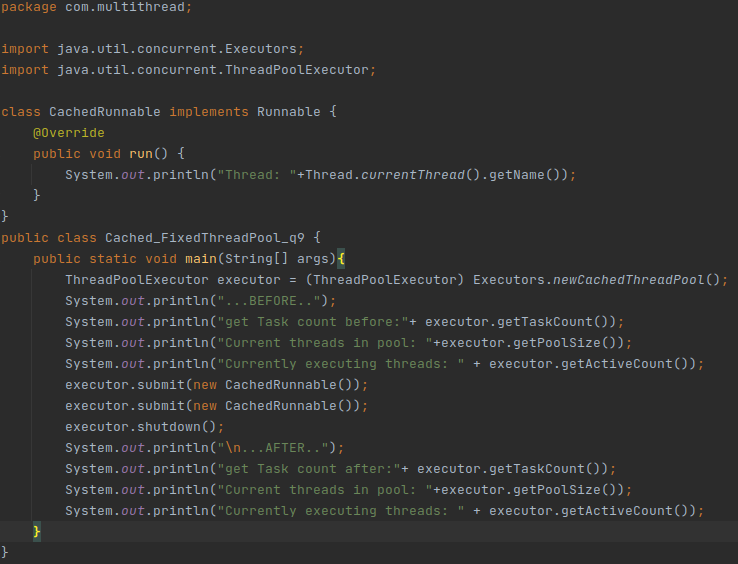


scheduleAtFixedDelay(): 



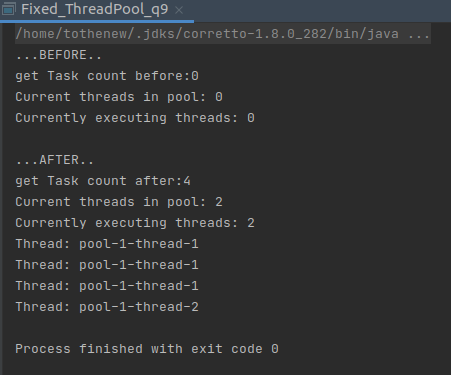
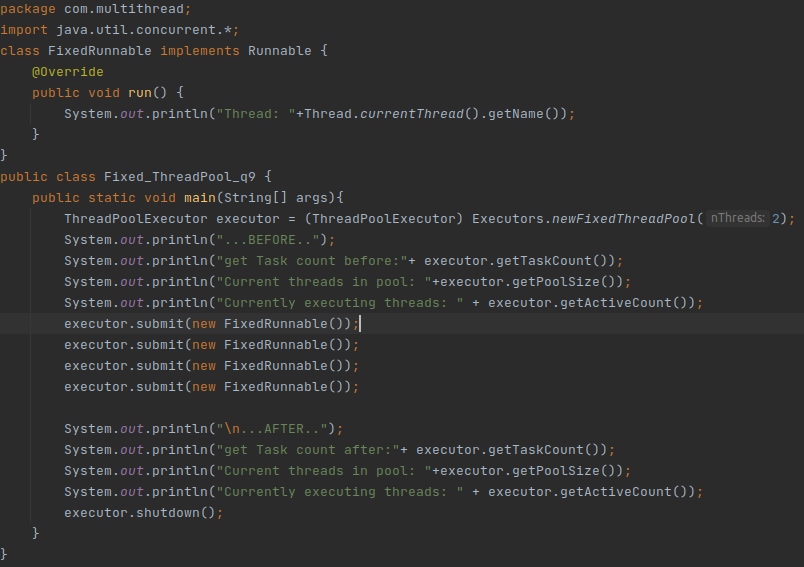
Q-9: Increase concurrency with Thread pools using newCachedThreadPool() and newFixedThreadPool().

CachedThreadpool():



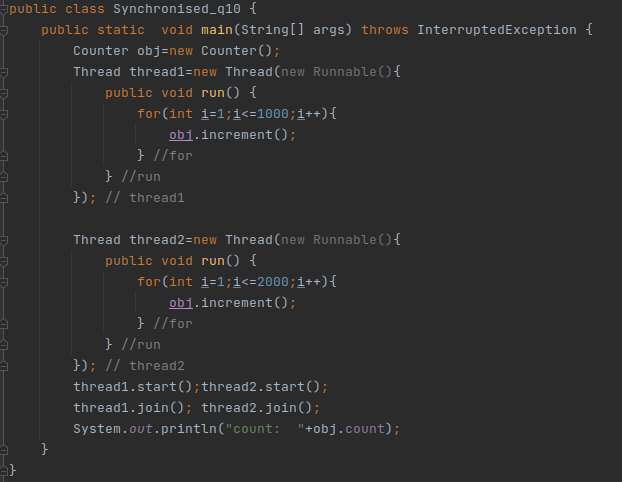
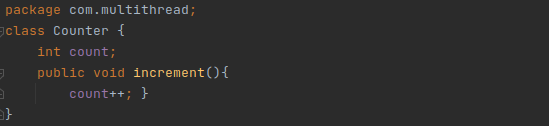


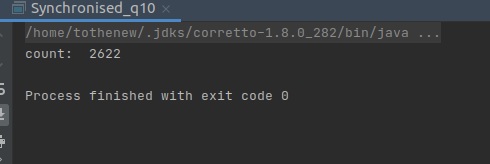
fixedThreadPool()



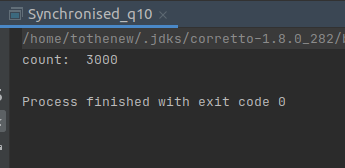
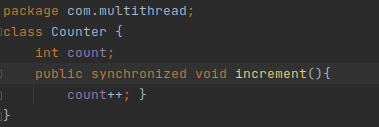
Q-10: Use Synchronize method to enable synchronization between multiple threads trying to access methods at same time.

Without synchronised method:

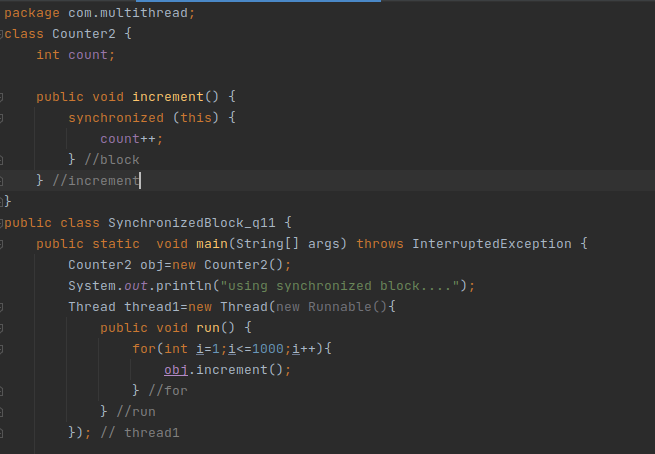


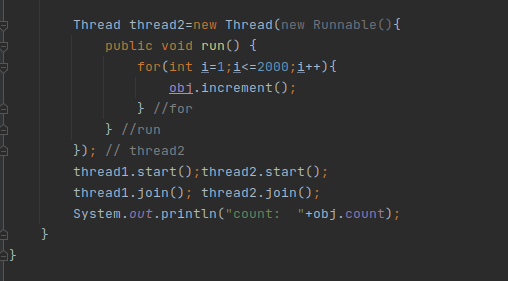
Output: 

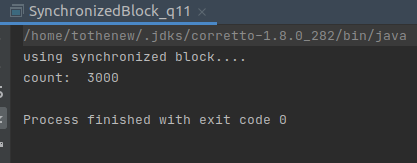
With synchronized method:

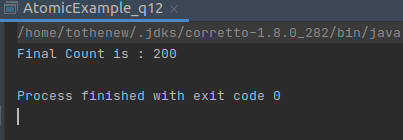
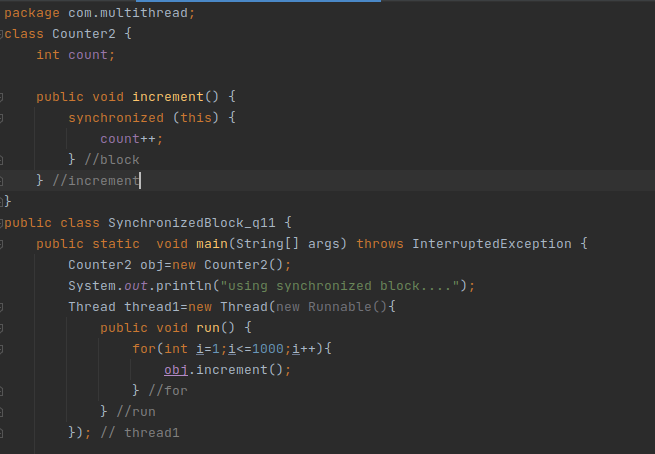


Using synchronized block:

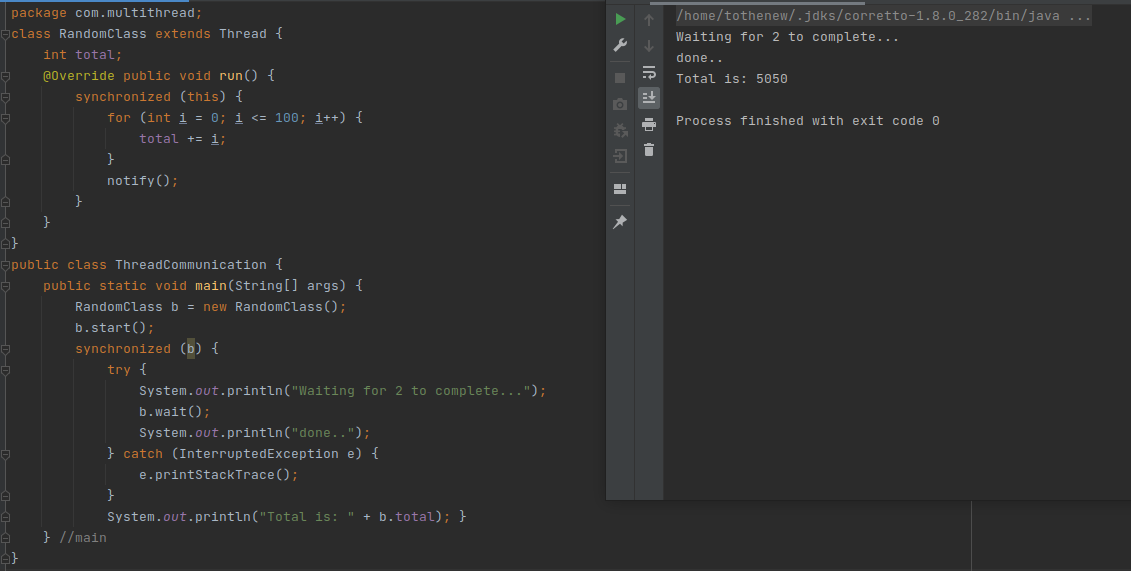




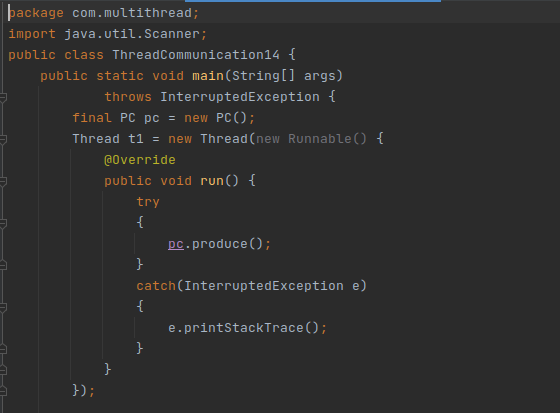


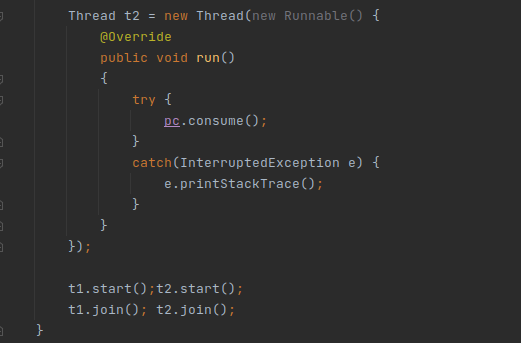
Q-12: Use Atomic Classes instead of Synchronize method and blocks.

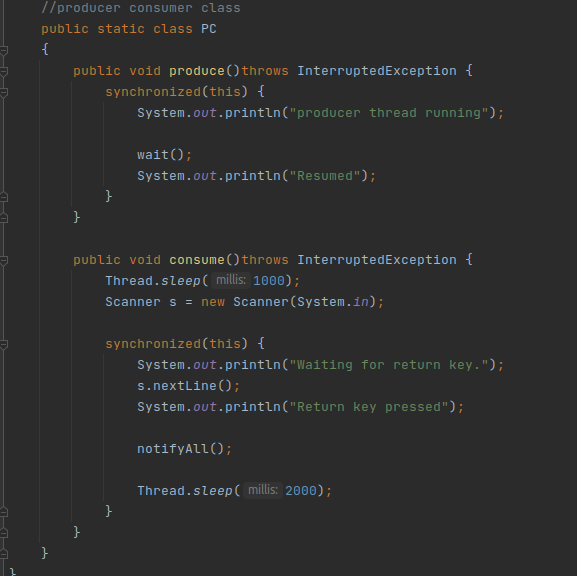
Q-13: Coordinate 2 threads using wait() and notify().

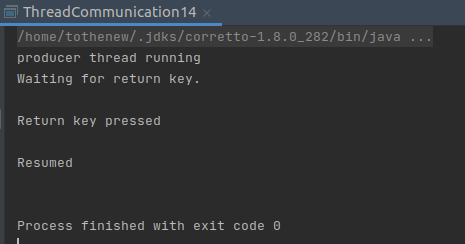


Q-14. Coordinate mulitple threads using wait() and notifyAll()







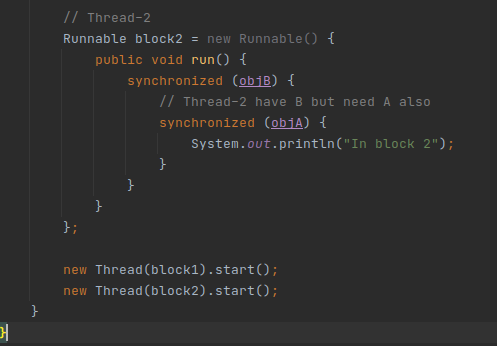
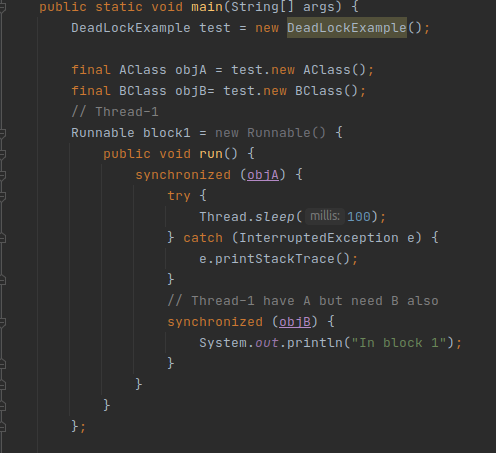
Output: 

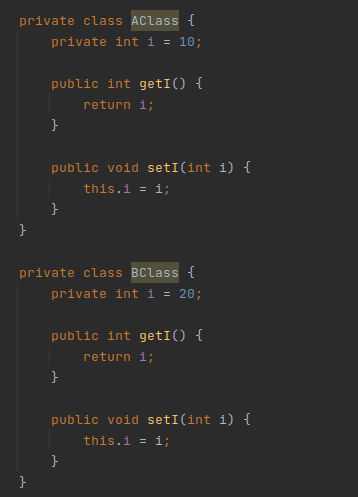
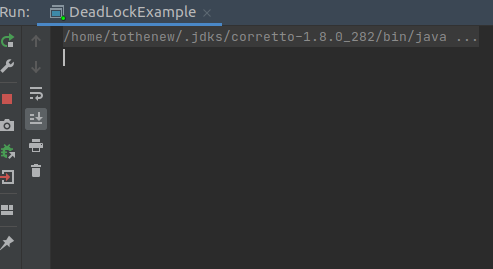
Q15: Use Reentrantlock for coordinating 2 threads with signal(), signalAll() and wait().

// not able to do this.

Q-16: Create a deadlock and Resolve it using tryLock().

// Deadlock



Resolving:

