1. Write a class StackAccess that has a stack object of size 3 as its property. The class allows only one thread to read from (pop), write to (push), and retrieve without delete from (peek) the stack at one time.

   If the stack is empty, a thread wishes to pop from the stack will wait until it is not empty, or to discard the pop operation after 1 second of waiting. If the stack is full, a thread wishes to push to the stack will wait until it is not full, or to discard the push operation after 1 second of waiting. Use lock condition and its methods to accomplish this waiting mechanism that prevents any infinite waiting.

   Write 3 task (Runnable) classes ReadStack, WriteStack and PeekStack that pops from, pushes to and peeks from the stack respectively. Each task performs its respective operation 4 times, with a 20ms interval between one operation and the next. The WriteTask pushes to the stack an arbitrary number between 0 and 99 in each of its operation.

   Then, write a driver class with a thread pool of 3 to test your implementation of StackAccess, ReadStack, WriteStack and PeekStack. Test your implementation with different test data, where your test should cover different scenarios such as:

   - There is only push/pop/peek task.
   - There are two types of task: push & pop, push & peek, pop & peek.
   - There are three types of task.
   - Produce more data than it is consumed.
   - Consume more data than it is produced.
   - Others