**Laboratory 5**

1.a. The purpose of the synchronized methods in the ProducerConsumer class is to lock the instance of the ProducerConsumer in order to prevent any other thread interference.

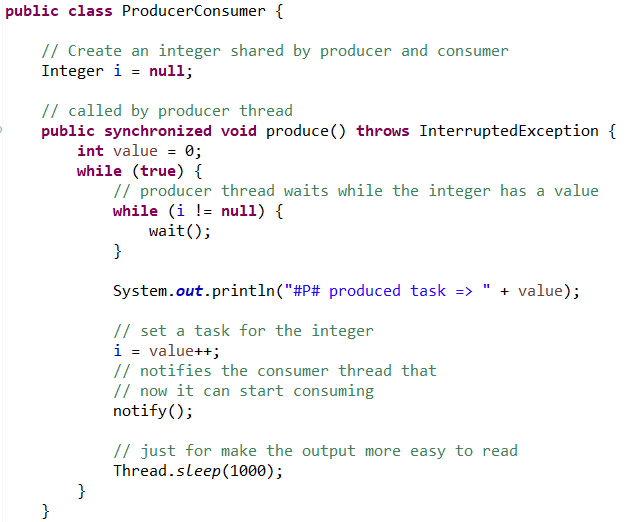
When a thread invokes any of the 2 synchronized methods, it automatically locks the ProducerConsumer instance, setting it inaccessible for any other threads, and only releasing this object when the thread ends execution for that specific method.

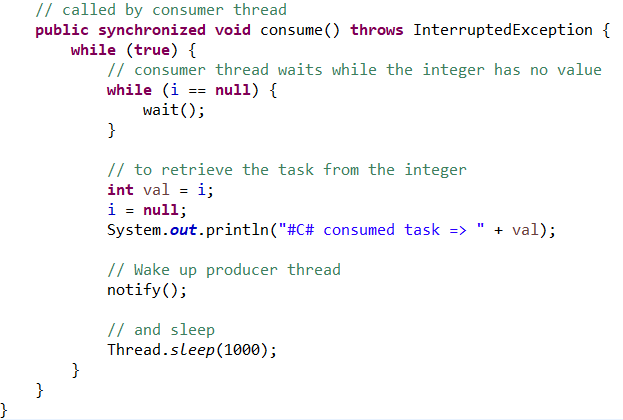
1.b. The method notify() wakes up one thread chosen arbitrary from the ones that are waiting on this object’s monitor. The awakened thread will not be able to proceed until the current thread unlocks the object.

The method wait() causes the current thread to wait until it is awakened, usually by being notified or interrupted. In order to use this method, the current thread must own the object’s monitor lock. Also, this method relinquish all the synchronization claims only on the object where this method was called, any other objects on which the current thread may be synchronized remain locked.

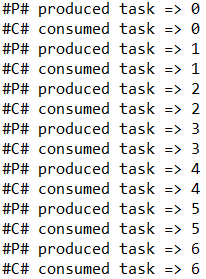
After a thread is awakened from the wait faze, it is re-enabled to compete with other threads for the right to synchronize on the object. Once it regains control, all its synchronization claims on the object are restored and returns to the situation where the wait() method was invoked.

2. In order for the program to produce one item and immedeiately consume it, then repeat this procedure, I replaced the list with an integer and modified the code accordingly in the ProducerConsumer class:





The result of running this code is:



Can be observed that the action of producing and consuming of an item alternates.