CLASSICAL AND QUANTUM COMPUTING
EXERCISE X

Consider the producer-consumer problem. A number of *producers* create objects which are used by *consumers*. All the objects are stored in the same place until they are needed. If the storage place is full, a producer must wait until there is place. Similarly, if a consumer requires an object and the storage place is empty, the consumer must wait until a producer places an object in the storage place.

An ideal data structure for this task is a *queue*. A queue is a first in first out structure. Data can only be added to the back of queue, and removed only from the front of the queue. Implement a queue data type.

Implement a producer and consumer using the `Thread` class of Java. Use `synchronized` methods and the methods

```
public final void wait() throws InterruptedException
public final void notify()
public final void notifyAll()
```

to ensure that the queue works correctly in a multi-threaded environment. `wait` causes a thread to wait on the associated object. Similarly `notify` and `notifyAll` notifies a thread or all threads respectively, which are waiting on the associated object, to resume execution.