TDT4186 Operating systems
Assignment 2
Group 205: Péter Gombos and Audun Skjervold

We implemented the buffer as an array, using two counters; one to keep track of where the producer (doorman) should write to next, and one to keep track of where the consumer (barber) should read from next. We implemented if-tests to use the array as a circular buffer.

The Doorman and Barber classes both extend the Java class Thread, allowing us to use multi-thread functionality. When one thread is executing a synchronized method for an object, all other threads that invoke synchronized methods for the same object suspend execution until the first thread is done with the object. Therefore, using synchronized methods, we are able to use the following logic:

The doorman thread will try to add a customer, but if incrementing the write counter (adding a customer) would bring the counter past the read counter, this means the queue is full, and the doorman thread will instead sleep for a semi-random amount of time before trying again. Because the methods in both threads are synchronized, we know that the read counter will not change while the doorman/producer is trying to add a new customer, but will rather wait until the doorman/producer is finished.

Similarily, the barber thread will attempt to take a customer from the queue, but if incrementing the read counter (the equivalent of taking a customer) would bring it past the write counter, this would mean that the queue is empty, and the barber thread will sleep for a semi-random amount of time, allowing the doorman/producer to produce new data for the queue.

In the original deliverable we had implemented the barber thread without proper wait/notify logic, and rather letting the barber thread sleep for a semi random amount of time and then polling again to see if there are customers ready to be served. This is highly inefficient as there is no guarantee that there will be available customers when the thread wakes up, and in the worst case scenario, a barber could spend all his time sleeping and not see any customers coming in.

The correct way to implement this is to let the customerqueue tell the barber threads to wait if there are no customers in the queue. While waiting, the thread will be sleeping, only to be waken up again by a notify call from the queue if there now is customers there. The same has been done for the doorman, with the addCustomer-method.