Practice Script 5

Summary:
- Object-Oriented Concurrent Programming
  (shared object communication model).
- Programming all aspects of object synchronization.

Exercise 5.1

Add a public interface for external synchronization (services `grab`, `release`, and `isGrabbedByMe`) to the shared generic queue developed in the previous lecture (exercise 4.1).

Implement this class with Java's native monitor support.

Exercise 5.2

Add the possibility for external synchronization to the class `Logger` (from exercise 4.2).

Test the class by a multi-threaded program, with threads that do a simple log (internal synchronization), and others that grab the logger for multiple logs (external synchronization). Use appropriate random waiting times is each thread to enhance the testing simulation.

Exercise 5.3

Implement the *dining philosophers* problem[1], considering that each fork is a shared object.

To make the problem most interesting, generalize it to any number of philosophers, have each fork to record the number of times it was used, and also register the number of meals for each philosopher.

Try different strategies to ensure the absence of *deadlocks* and *starvation* liveness problems. In particular, try a solution to ensure a fixed order in the reservation of the forks, and another that try to "catch" the two forks at once (all or nothing locking semantics).

[1]`http://en.wikipedia.org/wiki/Dining_philosophers_problem`