Alex Szeto

CS 4348 Project 2

Project Objective

This project was to help facilitate the learning of using semaphores to direct threads that compete for memory resources so that a sort of sequential type of form is seen.

The synopsis of the project is the following:

*The clinic to be simulated has doctors, each of which has their own nurse. Each doctor has an office of his or her own in which to visit patients. Patients will enter the clinic to see a doctor, which should be randomly assigned. Initially, a patient enters the waiting room and waits to register with the receptionist. Once registered, the patient sits in the waiting room until the nurse calls. The receptionist lets the nurse know a patient is waiting. The nurse directs the patient to the doctor’s office and tells the doctor that a patient is waiting. The doctor visits the patient and listens to the patient’s symptoms. The doctor advises the patient on the action to take. The patient then leaves.*

Beginning the project, it was hard to get my head wrapped around the use of semaphores and their placements and implementations to effectively use them in a project. Although there were many hardships trying to get my mind wrapped around the idea of using semaphores, the constant reading of example materials flushed out the ordering and syntax used in the java util semaphore package.

I tried putting each subclass in their own source code file, but I quickly realized that that would make the shared global variables harder to get without implementing a getter and setter for each of the variables and the semaphores. That is why the subclasses are under the Project2 main class, keeping the global variables as private.

The ending of the project was also a bit confusing, where initially the program did not exit. I then realized that my functions were put in an infinite loop in order to make sure that all the patients were addressed. I attempted to write an exit function to force exit the loop, but I could not figure out an exit condition so I used a forced interrupt in the main method.

At that point, the program exited before any of the threads finished running (or started in some of the cases). Then, I learned of the .join() function that ensured the completion of that thread before progressing so I put a join on every single one of the patients to ensure that all patients were addressed before the program exited.