CSC2002S Assignment 2 Report

The way I enforced the simulation rules was by enforcing a latch that would indicate once the start button is pressed for all the threads to start try to enter the club. This was done in the Clubgoer class as this is the creation of each patron and controlled their actions. To implement the pause button I used a classic wait that when if the pause button was pressed, it would change a flag variable which in this case is a AtomicBoolean to true telling the threads/patrons to wait until that flag variable turned to false, after which it turns false all the threads continue as they were previously, this was done in the ClubGoer class with the atomicboolean being in the clubSimulation class as this Is where the button action resides. To enforce the max number of patrons for the club I used another lock that when a flag variable turned to true, the patron would have to wait till another thread left the club after which it would signal to the other for it to enter, this was done in the ClubGoer class as this is where the patrons were defined and is where their actions were controlled. To make sure the patrons keep a safe distance from each other I used a synchronization lock on a specific gridblock making it so that a thread can only access that gridblock once the other thread that was using it, is done with it preventing patrons from being in the same gridblock, this was the same case for making the patrons move block by block and simultaneously and both synchronization methods were performed in the ClubGrid. Majority of the synchronization was done In the ClubGrid class as this is the shared object that all the classes used.

The challenges I faced include the getting of the IllegalMonitorStateException error many times which after a few times I learned to fix. As it indicated to me that a thread tried to wait or notify other threads without owning the monitor making the need for that notify to be synchronized. It taught me a lesson to always double check my synchronization locks.

To prevent deadlocks in my program I made sure that the synchronized blocks I created were used in a certain order making it so threads access the information they need in the exact same order each time preventing threads from being in a lock and waiting for each other.