PCP2

Concurrency Report

MDSOMO001

**Start, Pause, and Quit Buttons**

Start Button: I put the latch.await() in the startSim() method so that threads wait there when the latch is not open yet. Then in the start button, I release the latch when start button is pressed which allows threads to begin and patrons start entering the club.

Pause Button: Pauses and resumes the simulation by toggling the isPaused atomic boolean. When isPaused is true, sleep() is called and simulation pauses and the pause button reads “Resume”. When pressed again, it changes back to “Pause” and sleep() is no longer being called.

Quit Button: Terminates the simulation by exiting the application.

**Entrance and Exit**

Entrance and Exit Locks: Entrance and exit doors are synchronized to ensure that only one patron can access them at a time.

**Maximum Patrons Limit**

PeopleCounter: Keeps track of number of patrons inside and outside the club, along with the maximum limit.

Waiting and Capacity Checks: There is a counter variable that keeps track of all patrons, if counter.overCapacity is true, (club is full) then wait() is called and patrons wait outside the club. Patrons still arrive when capacity is reached though, they just wait outside.

**Movement and Distancing**

GridBlock and Clubgoer: Patrons' movement is governed by the GridBlock class, ensuring that they move one block at a time and maintain a realistic distance from each other.

Simultaneous Movement: Patrons move simultaneously in the grid to avoid deadlocks and ensure liveness.

**Deadlock Prevention**

Order of Lock Acquisition: Proper order of acquiring locks is maintained to prevent circular waiting and deadlock situations.

Resource Allocation: Adequate synchronization mechanisms are in place to ensure that threads don't get stuck due to resource contention.

**Andre**

I was able to implement Andre in the GUI but I struggled to get him to move or serve drinks.