CSC2002S 2023 ASSIGNMENT MULTITHREADED CONCURRENT CLUB SIMULATION REPORT

Lwazi Mavumbuka
MVMLWA003

The start button

To fix the start button for the simulation, a static CountDownLatch variable named startlatch is created in the Clubgoer class and wait is called under the startSim method to block all the threads so that the simulation does not begin. This CountdownLatch is then initialised with a value of 1 in the main method of ClubSimulation class. When the start button is clicked the startlatch begins to decrement and the simulation begins.

The pause button

To make sure that the pause button works, an AtomicBoolean named paused is created in the Clubgoer class. The checkPause method keeps checking if the boolean paused is set to true, and if it is it sleeps the threads for 100 milliseconds and checks again. On the ClubSimulation class, paused is initialized to false. When the paused button is pressed, Clubgoer/paused is then set to true which will set the state of the simulation. Clubgoer is synchronized to ensure safety since it is a shared variable among all the threads. If the game is paused the text of the button is set to “resume” and if not the button remains as “pause”

Entrance

To ensure that one patron enters at a time, in the enterClub method of the Gridblock class, a while loop is created, the while loop continues while the counter is still over capacity, inside the while loop counter is synchronized and calls wait do that all the other threads cannot enter while the counter is over capacity.

LeaveClub

In the leaveClub method, the variables currentBlock and entrance are synchronised to ensure thread safety. When a patron leaves the notifyAll method is called to notify all the threads that are waiting outside of the club. A while loop is created, this loop checks if counter is not over capacity, and if it is not, notifyAll is called to notify other threads to enter the club

Deadlocks

To prevent deadlocks in my code all the shared variables and methods such as counter and entrance have been synchronized. All the getters and setters are synchronized to ensure thread safety