MULTITHREADED CONCURRENT CLUB SIMULATION.

The goal of this assignment is to correct and extend the existing code for a multithreaded Java simulation of patrons in a club. Utilize synchronization mechanisms to ensure that the simulation adheres to specified synchronization constraints and maintains safety and liveness.

Here are behaviour rules of the simulation and how I fixed them:

1. ### Start button.
   The Start button initiates the club simulation. Latch.await() and CountDownLatch are used in startSim() method in Clubgoer class to initiate the latch(make threads wait) and once the Start button on the club Simulation is clicked, the latch will be released and the simulation will start. There is AtomicBoolean 'first' flag that ensures this occurs only once, even if multiple threads call the startSim() method. The latch is set for each thread in ClubSimulation class using for loop and then threads can run(start()) when the start button is clicked.

2. ### Pause button.
   The Pause button pauses/resumes the simulation. The pause button set/unset the value of a shared AtomicBoolean "pause", notifying waiting threads when it sets pause to false. This flag is shared with the Clubgoer threads when they are constructed. The Clubgoer threads have a method "checkPause()" for pausing that synchronizes "pause" to make each Clubgoers wait when pause is true.

3. ### The Quit button terminates the simulation (it does).

4. ### Patrons enter through the entrance door and exit through the exit doors (they do).

5. ### The entrance and exit doors are accessed by one patron at a time.
   The method "enterClub" which takes in PeopleLocation and returns entrance Gridblock which is where patrons/Threads enter the club. Entrance is synchronized to ensure one patron can access the entrance door at a time. One thread can check if the block is occupied (while loop), if not then it can enter the club. getExit() method in ClubGrid is also synchronized to ensure one thread can exit the club at a time.

6. The maximum number of patrons inside the club must not exceed a specified limit.

   While loop is used inside enterClub() method in Club for threads to wait if getInside()==getMax() (when the club has reached limit), this is synchronized to ensure only one thread can check at a time (multiple threads check and delays may result in overcrowd).

7. Patrons must wait if the club limit is reached or the entrance door is occupied.

   Like 6. and also entrance is synchronized to ensure only one thread can check at a time. While loop is used to check if the entrance block is occupied so that threads can wait if it is occupied, else enter. Threads also wait 2 seconds before entering the club.

8. Inside the club, patrons maintain a realistic distance from each other (one per grid block).(they do)

9. Patrons move block by block and simultaneously to ensure liveness.

   I made changes maxWait to 2000 and minWait to 1000 (used to calculate moving speed), so that patrons are not moving fast/jumpy (seem like they move smoothly)

10. The simulation must be free from deadlocks.

    There are no deadlocks as I have been releasing locks correctly whenever a certain thread is done using the lock. I have synchronized my code where necessary such as (entering/exit the club, pause).

Rector Ratsaka

RTSREC001