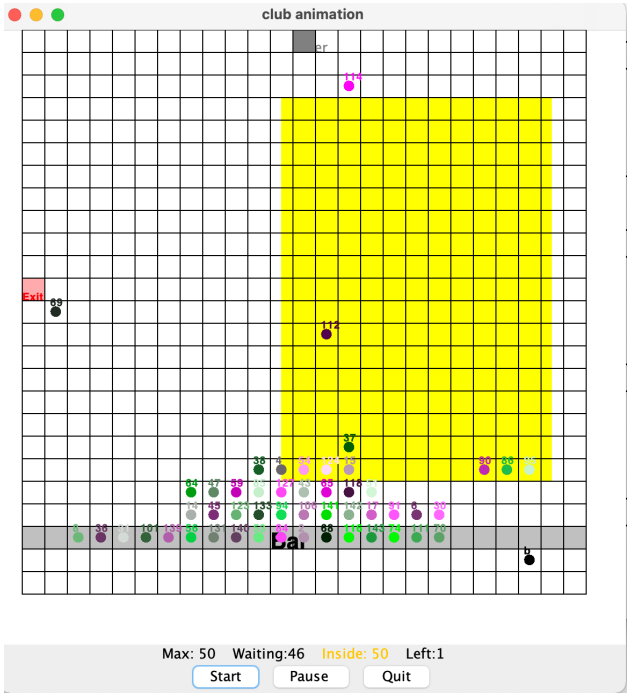
CSC2002S 2023 ASSIGNMENT
MULTITHREADED CONCURRENT CLUB SIMULATION
Club Rules (Version 1)



1. Objective:

In this assignment, your objective 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.

2. Club simulation

The assignment entails producing a Java simulation of a club, ensuring that it follows defined behavioral rules.

In this simulation, a club is represented as a **grid** of a specified size**,** containing an **entrance door**, an **exit door**, a **dance area** and a **bar**.

The simulation begins when the user presses the **Start button**. Patrons enter through the entrance door, proceed to the bar, then engage in various activities like dancing and wandering around. After a while, they decide to leave and head for the exit, leaving one by one.

Counters keep **accurate track** of the people waiting to enter the venue, those inside, and those who have left. The **pause button** pauses/resumes the simulation, and quite exits the program.

You do not need to implement the whole simulation – you are provided with a skeleton Java implementation. The simulation runs, but **a number of the rules for the simulation are violated**.

Your task is to fix the code, identifying and correcting all the concurrency issues so that all rules are all complied with all the time and the simulation suffers from no safety or liveness problems.

There is also the option to implement the **Andre the Barman extension** to the simulation.

3. The behaviour rules for the simulation

- The **Start button** initiates the simulation.
- The **Pause button** pauses/resumes the simulation.
- The Quit button terminates the simulation (it does).
- Patrons enter through the entrance door and exit through the exit doors.
- The entrance and exit doors are accessed by **one patron at a time**.
- The maximum number of patrons inside the club must not exceed a specified limit.
- Patrons must wait if the club limit is reached or the entrance door is occupied.
- Inside the club, patrons maintain a realistic distance from each other (one per grid block).
- Patrons move block by block and simultaneously to ensure liveness.
- The simulation must be free from deadlocks.

4. Andre the Barman

Upon successfully completing the above requirements, you have the option to extend the simulation by implementing Andre the Barman. Andre serves drinks to patrons at the bar, patrons must not leave the bar before being served. Implement Andre as a separate thread.

5. Code Description

The assignment provides a skeleton Java implementation with eight classes:

ClubSimulation.java: Main class for setting up and starting the simulation.
ClubView.java: JPanel class for visualization.
CounterDisplay.java: Class for displaying/updating counters.
Clubgoer.java: Class representing each patron as a thread.
PeopleLocation.java: Class for storing patron locations.
PeopleCounter.java: Class for tracking people inside and outside the club.
ClubGrid.java: Class representing the club grid.
GridBlock.java: Class representing grid blocks.

6. Assignment Submission:

Submit an assignment archive named with your student number and the assignment number (e.g., KTTMIC004_CSC2002S_PCP2). The archive should include:

- A concise report in PDF format.
- All classes mentioned above, including Andre the Barman's class.
- A Makefile for compilation.
- A readme file explaining how to run the program (input parameters cannot be changed).

- A GIT usage log in .txt format (use git log –all to display all commits and save).

Short Report

In the report, explain how you enforced simulation rules, your design approach for Andre, challenges faced, and lessons learned. Detail the synchronization mechanisms you added to each class and their appropriateness. Explain how you ensured liveness and prevented deadlock.

Marking

Ensure compliance with all specified requirements. Zero marks will be given if uploaded classes do not run with existing classes. Stay updated with any additional marking information on MSTeam.