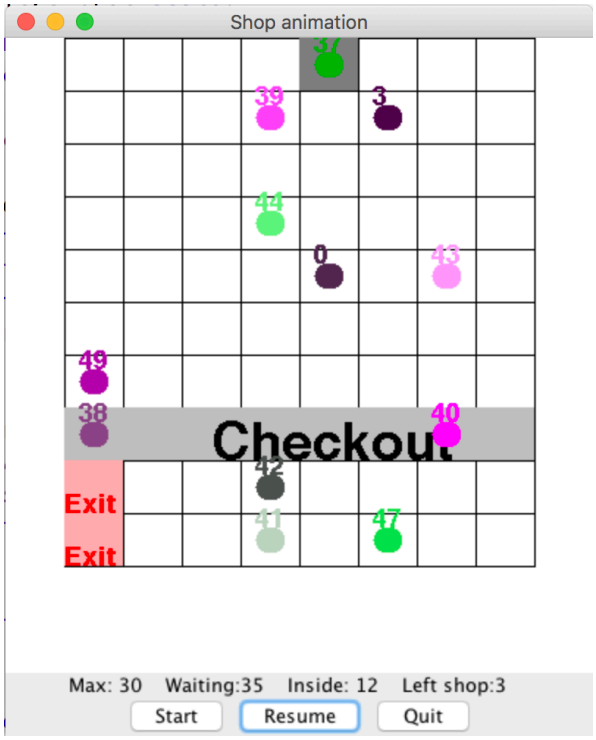
CSC3002F OS PART 2
ASSIGNMENT - CONCURRENCY

ENFORCING PANDEMIC LOCKDOWN LEVEL 3 RULES

DUE DATE: 22ND JUNE 2020, 10AM

In this assignment, you will correct the code for a multithreaded Java simulation of customers entering a shop under Covid19 pandemic level three regulations, using synchronization mechanisms to ensure that it operates within the specified synchronization constraints.

1. Social distancing shop simulation



The aim of this assignment is to produce a simple Java simulation of a shop of a specified size. In this simulation (an example screenshot is shown on the left) a shop is represented as a grid and customers (each have different colours and unique id numbers) arrive at random times through a single entrance door. On entering the shop, customers first browse (some customers move faster than others) and then head for the checkout. After checkout they leave.

The counters at the bottom of the simulation window 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.

You need to update this simulation to implement Covid19 pandemic lockdown Level 3 rules, as follows.

1.1. The rules for the simulation

- Customers enter through the entrance door and exit through the exit doors.
- Only one customer is allowed in through the entrance door at a time.
- There is a **specified limit** to the **maximum number of customers** allowed into the shop at one time. **This must not be exceeded.**
- Customers must **wait to enter the shop** if the limit is reached.
- **Customers should enter the shop in the order in which they arrived**.
- **Once in the shop,** customers must keep a distance from other customers – **only one customer allowed per grid block at any time**.
- Customers **may never "jump" blocks** – they move block by block.

- Customers must move **simultaneously.** (You need to **ensure liveness** in the simulation)**.**
- The simulation **must be deadlock free.**

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 are violated** – there are some messages in the text output to indicate some (not all) of the rule violations.
Your task is to fix the code, identifying and correcting all the concurrency issues so that all rules are all complied with and the simulation suffers from no safety or liveness problems.

2. Code Description

One of the aims of this assignment is to give you valuable experience in reading and understanding existing code, as well as and making changes to it without breaking it. So we will not be explaining the code in detail. It is sparsely commented.

The simulation comprises eight classes, which are listed below.
`SocialDistancingShop.java` – the main class.
`ShopView.java` – JPanel class updates the canvas and buttons for visualizing the simulation.
`CounterDisplay.java` – class to display/update the counters on the JPanel.
`Customer.java` – class for each customer.
`Inspector.java` – class for inspector who reports on some violations.
`CustomerLocation.java` – class where locations of each customer are stored.
`PeopleCounter.java` – counter class to keep track of people inside and outside shop.
`ShopGrid.java` – class representing the shop.
`GridBlock.java` – class representing blocks for the shop.

You **can only make changes to these four classes**:
`CustomerLocation.java`,
`PeopleCounter.java`
`GridBlock.java` and
`ShopGrid.java`.
These are the classes you will submit for the assignment. **You may not make ANY changes to ANY other classes.**

1.2. Input parameters

The program takes as command line parameters (in order, there are defaults if these are not supplied):

- the number of customers who will arrive
- the length of the shop in grid squares
- the breadth of the shop in grid squares
- the number of people allowed into the shop at any point

The location of entrance, exit doors and the checkout counter are hard-coded.

3. Assignment submission

3.1. Code

Submit these classes as part of your zipped archive:
`CustomerLocation.java`,
`PeopleCounter.java`
`GridBlock.java` and
`ShopGrid.java`.

You must annotate these classes with clear comments to explain what you have done and why: highlight any changes you made and why you made those specific changes. Also comment on sections of the code that did not need changes made and why.

1.1. Short report

You need to write a concise (short) report**, in pdf format**, that answers the following questions:

- Which threads run in the program?
- Which classes are shared amongst threads?
- Explain the synchronization mechanisms you added to each class and why they were appropriate.
- How did you ensure liveness in the code?
- How did you protect against deadlock? Was this necessary?

1.2. Assignment submission requirements

- Your submission archive must contain the report (**pdf format**) and your solution code.
- Upload the file and **then check that it is uploaded.** It is your responsibility to check that the uploaded file is correct, as mistakes cannot be corrected after the due date.
- The deadline for marking **queries** on your assignment is **one week after the return of your mark.** After this time, you may not query your mark.
- You may not ask your tutor for extensions or concessions – **all queries must be directed to the course lecturer.**

1.3. Marking

- Pay attention to all the requirements listed in this document. The marking template will be based on this.
- **If the classes you upload do not run with the existing classes, you will get zero for the assignment.**