UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Department of Computer Science
COS 226 - Concurrent Systems Date Issued: 08 September 2021

Practical Assignment 2

- **Due Date:** 16 September 2021
- **Assessment:** The practical will be assessed offline.
- This practical consists of 2 tasks. Read each task **carefully**!

1 Information

1.1 Objective

This practical aims to further explore locking via the implementation of new locks.

You must complete this assignment individually.

1.2 Provided Code

You will be provided with some skeleton code to aid in the assignment, consisting of the following Java classes:

- Main.java
- Queue.java
- Store.java
- Filter.java
- Bakery.java

1.3 Mark Allocation

For each task in this practical, in order to achieve any marks, the following must hold:

- Your code must produce console output. (As this is not marked by fitchfork, formatting is not that strict)
- Your code must not contain any errors. (No exceptions must be thrown)
- Your code may not use any external libraries.
- Your name and student number MUST appear in EVERY file you upload.

The mark allocation is as follows:

Task Number	Marks
Task 1	10
Task 2	5
Total	15

2 Assignment

Due to the Covid-19 restrictions, a grocery store has decided to only allow one customer inside it's doors at a time. The store has 4 entrances and therefore 4 queues of people wishing to get inside. Each queue of people has 5 people in it. You are tasked to simulate this scenario and ensure that only one person gains access to the store at a time.

2.1 Task 1 - Filter Lock

For this task you will need to implement the simulation of the above mentioned scenario as well as implement a FilterLock to enforce mutual exclusion.

The following needs to be completed:

- The **run()** method of the **Queue** class needs to simulate 5 people accessing the store through the **enterStore()** method of the **Store** class.
- The **enterStore()** method needs to simulate a person purchasing items from the store. To do this, once inside the store, the thread representing that person will need to sleep for a randomly selected amount of time between 200 and 1000 milliseconds. Remember only one person is allowed in the store at any time!
- A FilterLock will need to be implemented inside the **Filter** class. i.e. A **lock()** and **unlock()** method.
- The following output is expected:
  - When a person ATTEMPTS to enter the store, the following will need to be output: [Thread-Name] Person: [Person-Number] is trying to get inside.
    Example: Thread-1 Person 2 is trying to get inside.

  - When a person ENTERS the store, the following will need to be output: [Thread-Name][Person-Number] has entered the store.
  - When a person EXITS the store, the following will need to be output: [Thread-Name][Person-Number] has left the store.

2.2 Task 2 - Bakery Lock

Some of the store patrons are complaining that the method of allowing people inside the store is unfair. To solve this problem we will devise a new method of allowing people inside.

The following needs to be completed:

- The FilterLock from the previous task needs to be replaced by a BakeryLock.
- Implement your BakeryLock inside the **Bakery** class.
- Change the simulation you have created to make use of the BakeryLock instead of the FilterLock.

2.3 Submission

Submission of this assignment will be via click-up, please zip all of your source code from each task SEPERATELY and upload each zip folder to the click-up submission.
Ensure your name and student number are present in all of your classes!

2.4 Notes

- Each queue into the store is represented as a thread, therefore to simulate multiple people per queue you will need to make use of a loop inside the **run()** method of the **Queue** class.
- Remember to create copies of your code after completing task 1 as you will need to upload two separate zip folders.
- Tutors will be available on the 9th of September for queries.