UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Department of Computer Science
COS 226 - Concurrent Systems Date Issued: 07 November 2021

Practical Assignment 4

- **Due Date:** 11 November 2021
- **Assessment:** The practical will be assessed offline.
- This practical consists of 3 tasks. Read each task **carefully**!

1 Information

1.1 Objectives

This practical aims to explore various thread implementations and their relative performance.

You must complete this assignment individually.

1.2 Provided Code

You must use your code from Practical 2 to complete this practical.

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.
- You must be able to explain your code live to a tutor and answer any questions asked.

The mark allocation is as follows:

Task Number	**Marks**
Task 1	5
Task 2	5
Task 3	5
Total	15

2 Assignment

By now you should be familiar with mutual exclusion, this practical aims to highlight the differences between various lock implementations. This practical builds on practical 2 to explore more ways of providing mutual exclusion through locks, as well as allowing multiple threads into the critical section. You will have to code 3 different types of locks for this practical.
For this practical you must have 5 queues of 10 people in each queue.

2.1 Task 1 - Test and Test and Set Lock

In this task you will need to implement a TTAS Lock. A reminder of Practical 2 follows: 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 TTAS Lock will need to be implemented inside a **TTAS** 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 - MCS Queue Lock

The following needs to be completed:

- The TTAS Lock from the previous task needs to be replaced by a MCS Queue Lock.
- Implement your MCS Queue Lock inside an **MCS** class.
- Change the simulation you have created to make use of the MCSQueue instead of the TTAS.
- The output remains the same as Task 1.

2.3 Task 3 - Semaphore

The following needs to be completed:

- The lock from the previous task needs to be replaced by a Semaphore.
- Implement your Semaphore inside a **Semaphore** class.
- Change the simulation you have created to make use of the Semaphore.
- The number of people allowed in the store is now 3 at a time.
- The output remains the same as Task 1, with the exception of one thing:
  - When a person ENTERS the store, the following will need to be output:
    [Thread-Name][Person-Number] has entered the store. Spaces remaining: [Number of spaces left]
    Example: Thread-1 Person 3 has entered the store. Spaces remaining: 1

3 Final Notes

Upload each task in a separate zip folder inside the SAME submission.
You will only get one submission for this task so ensure you have uploaded ALL zip files to your submission.