UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Department of Computer Science
COS 226 - Concurrent Systems

Assignment 1

- **Date issued:** 08 August 2022
- **Deadline:** 25 August 2022, 8:00 PM
- This practical consists of 2 task. Read each task **carefully!**

1 Introduction

1.1 Objectives and Outcomes

This assignment aims to explore locking via implementation of new locks when 3 or more threads are involved.

You must complete this assignment individually. Copying will not be tolerated.

1.2 Submission and Demo Bookings

A basic class setup for the simulation has been provided. You will have to create some classes from scratch.

Submit your code to **clickup** before the deadline.

You will have to demonstrate each task of this practical during the **physical** practical lab session. So be sure to create copies of your source code for each task separately. Booking slots will be made available for the practical demo.

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. (Note section 3 regarding data structures)
- You must be able to explain your code to a tutor and answer any questions asked.

The mark allocation is as follows:

Task Number	**Marks**
Implementation	10
Explanation of Code	10
Total	20

2 Assignment: Practical Requirements

Simulate the software development life-cycle from the start-to-finish of the following scenario.

2.1 Software Engineering

A software engineering team is trying to manage their project efficiently by dividing the software into deliverable components. Various team members will be responsible for developing different components within the project based on their expertise.

Team members:

- Software Developers
- Software Testers

The various components to be developed are stored in a queue:

- Develop: Contain components that will be developed by software developers.
- Testing: Contain components that needs to be tested before being integrated into the system.

2.2 Team Rules

- A developer may only develop components and a tester may only test components.
- Each team member will work for a random amount of time between 100ms and 500ms before they require a break.
- The break must be a random amount of time between 50 and 100ms.
- Team members will work until all components are developed and tested.
- The team has a variable amount of developers and testers.
- Only one developer may be developing a component at a given time, likewise only one tester may be testing a component at a time.
- A developers and a tester may work at the same time.
- **NB!** After a component has been developed, it must be added to the 'testing' queue.

2.3 Output

The following output must be produced:

- When a team member is ready to develop/test a component (they are not on break):
  [Thread-Name] is ready to develop/test a component.
- If a team member finishes a component during their shift:
  [Thread-Name] finished developing/testing [Component-Name]
- If a team member does not finish developing/testing a component:
  [Thread-Name] developed/tested [Component-Name] for [Time] ms. Time remaining: [Time-Remaining on component]
- When a team member goes on break:
  [Thread-Name] is taking a break.
- Due to the concurrent nature of the program, many outputs may be interleaved, however there are a couple of basic rules to follow:
  - If there has been a 'developing' output, there may not be another one until a 'break' output by the same thread has been output. Likewise with 'testing' outputs.
  - A thread must output 'ready to develop/test' before their corresponding 'developing/testing'output.

2.4 Example Output

```
...
Thread-3 tested Timetable for 250 ms. Time remaining: 1750
Thread-3 is taking a break.
Thread-3 is ready to test a component.
Thread-1 finished testing Calendar
Thread-1 is ready to test a component.
Thread-5 finished testing Phonebook
Thread-5 is ready to test a component.
Thread-3 tested User Manager for 189 ms. Time remaining: 1311
Thread-3 is taking a break.
Thread-1 tested Timetable for 500 ms. Time remaining: 1250
Thread-1 is taking a break.
Thread-3 is ready to test a component.
Thread-1 is ready to test a component.
Thread-5 tested User Manager for 342 ms. Time remaining: 969
Thread-5 is taking a break.
Thread-3 tested Timetable for 117 ms. Time remaining: 1133
Thread-3 is taking a break.
...
```

3 Notes

- You will have to get creative with handling threads, queues and locks.

- A general rule of thumb is that each queue should have their OWN lock when accessed.
- You may not REMOVE any given code.
- You may add as much code and as many classes as you deem necessary.
- Any locks used in the program will have to be written from scratch. i.e NO using java's pre-built locks.
- You may use any of Java's pre-built data structures.
- Any locks used must be FAIR.