UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Department of Computer Science
COS 226 - Concurrent Systems

Assignment 1

- **Date issued:** 18 August 2023
- **Deadline:** 17 September 2023 (Midnight)
- This assignment consists of a single task. Read **carefully!**

1 Introduction

1.1 Objectives and Outcomes

This assignment aims to test all the concepts that have been learned so far regarding mutual exclusion and locking via practical implementation without assistance.

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. Booking slots will be made available for the practical demo.

1.3 Mark Allocation

For the assignment, 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 for **locking** apart from those already provided.
- You must be able to explain your code to the tutors and answer any questions asked.

The mark allocation is as follows:

Task	**Marks**
Implementation	10
Explanation	10
Total	20

2 Assignment

For this assignment you are tasked with simulating a CRUD (Create, Read, Update, Delete) API requests.

2.1 CRUD API

You are part of a database development team using multi-threading to handle database request. An API request could be performing any of the following operations on the database:

- CREATE: Add a new record to the database.
- READ: Returns a copy of the all records in the database.
- UPDATE: Update a specific record in the database given partial information, id and name, about the record and the new information, practicals and assignments, to update. Updates first match.
- DELETE: Delete a specific record from the database matching the given information, partial/full. Delete first match.

The various requests to be made are stored in different queues based on the operation type. For each operation:

- There are n number of threads that handle the requests for that specific operation.
- Requests perform operations on the shared database.
- Only one thread may operate on the database at any given time.
- The different operations are handled concurrently.
- A request fails if the is no match, for UPDATE and DELETE, and the attempt attribute gets incremented.
- If a request exceeds 2 attempts the request is deleted otherwise its added back to the operations queue.
- Threads sleeps for random amount of time between 50 and 100ms after handling a request.
- Threads will operate until all request are handled i.e. the respective operation queue is empty.

2.2 Output

The following output must be produced:

- When a thread is ready to handle a request (they are not sleeping):
  [Thread-Name] [Operator-Name] is waiting for request.
- If a thread completes a request successfully:
  [Thread-Name] [Operator-Name] success [request-info]
- If the request is a READ output the database records:
  [Thread-Name] [Operator-Name]

  [record-1-info]
  [record-2-info]
  ...
  [record-N-info]

- If a thread does not complete a request successfully:
  [Thread-Name] [Operator-Name] failed [request-info]
- When a thread sleeps:
  [Thread-Name] [Operator-Name] is sleeping.
- 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 'waiting' output, there may not be another one until a 'success' or 'failed' and a 'sleeping' output by the same thread.
  - A request must receive a 'success' output before the effect reflects in the read operator output.

2.3 Note

- 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 javas pre-built locks.
- You may use any of Javas pre-built data structures.
- Any locks used must be FAIR