UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Department of Computer Science
COS 226 - Concurrent Systems

Practical 4

- **Date issued:** 30 August 2023
- **Deadline:** 06 September 2023 (Midnight)
- This practical consists of 2 task. Read each task **carefully!**

1 Introduction

1.1 Objectives and Outcomes

This practical aims to explore Spin Locks and Contention.

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

1.2 Submission and Demo Bookings

You will are provided with some skeleton code, consisting of the following Java classes: **Main.java, Node.java, Printer.java, MCSQueue.java** and **Timeout.java**

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 apart from those highlighted in the textbook.
- 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
Task 1	5
Task 2	5
Total	10

2 Practical Requirements

Your task is to simulate printing in a bus network using concurrent programming. A bus network is a local area network (LAN) topology in which each **node**, a workstation or other device, is connected to a main cable or link called a bus. All connected stations on the bus can communicate with all others on the singular network segment. This bus network follows a specific procedure to ensure the integrity of the communication between nodes.

2.1 Task 1 - MCS Queue Lock

In order to mitigate the issue of mixed print-work by the printer in the network, only one other node on the network is allowed to broadcast a print request to the printer at any given time. The process of printing is as follow:

- All nodes have to stand in a queue to print. There are 5 different nodes in the network excluding the printer. Each node must send at least 5 print request to the printer.
- Each node can only send a single print request at a time.
- When a node attempts to print some material it stands in a queue if there is one, otherwise it can just submit the request.

Note:

- A thread will simulate a node and the printer will be the critical section.
- Once a nodes' print request is first in the queue, the thread will sleep for a randomly selected amount of time between 200 and 1000 milliseconds before exiting.

2.1.1 Output

The following output is expected:

- When a node ATTEMPTS to print some material, the following will need to be output:
  [Thread-Name][Request-Number] printing request.
- When a nodes' request has been RECEIVED by the printer, the following will need to be output:
  [Thread-Name][Request-Number] printing [random-message].

- When a nodes' printing is DONE, the following will need to be output:

  QUEUE: {[Thread-Name]:[Request-Number]} -> ....
  Example: QUEUE: {Thread-1:Request 1} -> {Thread-2:Request 1}
  Note: This is the print request queue

Note that you will have to be creative in printing the queue. Hint: consider using the qnode to store the information.

2.2 Task 2 - Time-out Lock

The following needs to be completed:

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

Note: Select a timeout based on the acceptable time a node should wait/spend in printing request.