COS 226 Practical Assignment 2

- **Date Issued**: Monday, 29 July 2019
- **Date Due**: Sunday, 4 August 2019, 18:00
- **Submission Procedure**: Upload your tasks to the CS website
- **Assessment**: The practical will be marked in the week of 12 – 16 Aug 2019
- This assignment consists of **2 tasks** with a total of **15 marks** for the entire practical.

1. Introduction

From here on forward, the practical assignments assume that you know the basics of how threads work. You must complete this assignment individually.

You may ask the Teaching Assistants for help but they will not be able to give you the solutions. They will be able to help you with coding, debugging and understanding the concepts explained both in the textbook and during lectures.

2. Mark Allocation

This assignment is divided into **two tasks**. Your program must adhere to the following:

1. Your program must produce the expected output
2. Your program must not throw any exceptions
3. Your program must implement the correct mechanisms or algorithms, as specified by the assignment.

3. Source code

Before you begin, you should download the source code from the CS website. You are allowed to make changes to the files in the source code.

4. Task 1 – Filter Lock (10 marks)

In this task you must implement the filter lock algorithm that will provide the full implementation of the Lock interface and then test your lock on the Counter class. You are provided with **FilterLock.java, Counter.java, TThread.java and ThreadCounterDemo.java**

- The primary methods of the Lock interface which you must implement include the *constructor* which takes a single integer argument to indicate the number of threads, the *lock()* function and the *unlock()* function. You need to implement the Filter Lock as described in the textbook in the lock() and unlock() methods.

- To test your Filter lock, you will again make use of the shared counter from Practical 1. Make the necessary changes to the Counter.java file to ensure that mutual exclusion is enforced through the use of the Filter lock.

- You will have 4 threads running in this system as seen in ThreadCounterDemo.java. Each thread has to increment the shared counter 3 times and sleep for 500 milliseconds in between attempts. Each time the counter is incremented the new value of the counter as well as the ID of the thread that incremented it should be given as output as follows:

  ```
  Thread-0 1
  Thread-1 2…
  ```

- Be sure to have your thread IDs start at zero

- Once your threads have incremented the counter for a total of 12 times (3 times each), you will need to output the execution time of each thread. In other words, how long it took them to execute in milliseconds. The output for this should be in the form:

  ```
  Thread-0 executed for 1092 milliseconds…
  ```

- You are allowed to make changes to any or all of the given source code files as necessary.

Compress all your files into a single archive and upload the archive to the CS website to the **Practical2 Task1** submission box. Make sure that the uploaded file does not belong to any package.

5. Task 2 – Bakery Algorithm (5 marks)

In this task you have to implement the Bakery Lock algorithm as specified in the textbook. Use the same **Counter.java**, **TThread.java** and **ThreadCounterDemo.java** files from Task 1 and add your own **Bakery.java**.

- You have to implement the constructor, lock() and unlock() methods of the Bakery algorithm. The constructor should again take a single integer argument to indicate the maximum number of threads.

- Again test your lock with the shared counter class. Make appropriate changes to your code so that mutual exclusion to the critical section is now enforced through the Bakery algorithm.

- Use the same testing conditions as in Task 2 – 4 threads executing 3 times each and incrementing the counter after sleeping for 500 milliseconds.

- Again output the execution time of each thread after the thread has completed its run.

Compress all your files into a single archive and upload the archive to the CS website to the Practical2 Task2 submission box.