UCT CSC3002F 2019

Operating Systems Assignment 2: Synchronization
Part I (of III)

(version 1)

Lecturer: Michelle Kuttel

Due dates:

Tue 23 April 2019, 9am **(Part I)**

Tues 30 April 2019, 9am (Part II) – for later release

Thurs 9 May 2019, 9am (Part III) – for later release

Aim

This assignment aims to give you experience in thread (and, by extension, process) synchronization using **semaphores**. In this project, you will write multithreaded programs in Java to solve **three classic synchronization problems**, using only semaphores. A secondary aim is to give you experience in reading, understanding and correcting existing code. The assignment is in three parts, increasing in difficulty. The first part is described below.

1 Part I: Simple barrier with semaphores

In this first part of the assignment, you will implement a simple barrier in Java, using only **semaphores**.

A barrier is a synchronization construct that forces threads to wait until a fixed number of threads (*n)* have arrived at the barrier, whereupon they may all continue with their next step.

The **synchronization requirement** is that no thread may proceed until *n* threads have reached the barrier. When the first *n-1* threads arrive, they should block until the *n*th thread arrives, whereupon all threads may proceed.

Note that for this task **you must only implement a simple barrier**, not a reusable barrier. The barrier implemented **must be deadlock free**.

1.1 Code skeleton: BarrierS

You **must use** and extend the `BarrierS` **package provided**, correctly implementing the `Barrier` class (and the `b_wait()` method), so that the `BarrierTest` class will execute correctly, **always**. This will not require many lines of code. **Do not alter** the `BarrierTest` or the `BThread` classes in the package (although you must submit them). You will need to inspect the various classes to see how they work – this is part of the assignment and no explanation other than the code will be given.

An example of a correct execution of `BarrierTest` is:

- java BarrierS.BarrierTest 5 5

```
Starting simulation with 5 threads, barrier size 5

Parent thread completed

Thread 1 waiting at barrier.

Thread 3 waiting at barrier.

Thread 0 waiting at barrier.

Thread 2 waiting at barrier.

Thread 4 waiting at barrier.

Thread 4 passed barrier.

Thread 1 passed barrier.

Thread 2 passed barrier.

Thread 0 passed barrier.

Thread 3 passed barrier.
```

1 Code requirements

You will code your solutions in Java, adding to the skeleton code provided.

You may only use the **Semaphore** class in the `java.util.concurrent` library: **no other Java synchronization constructs at all**.

All code must be **deadlock free.**

The output must comply with the stated synchronization constraints and with the examples shown.

2 Assignment submission requirements and assessment rules

- You will need to create, **regularly update**, and submit a GIT archive of your code for each separate part (i.e. one archive for Part I, one for Part II and one for Part III).
- You are required to **extend the provided code templates**. If you fail to do this (i.e. submit alternative code), you will obtain a mark of 0 for the assignment.
- Each submission archive must include a **Makefile and README file** (including running/installation instructions) for compilation/running.
- Label your assignment archive with your student number and the assignment number e.g. KTTMIC004_CSC3002S_OS2_PartI
- Upload the files and **then check that they are uploaded.** It is your responsibility to check that the uploaded file is correct, as mistakes cannot be corrected after the due date.

- The usual late penalties of **10% a day (or part thereof)** apply to this assignment.
- The **deadline for marking queries** on your assignment is **one week after the return of your mark.** After this time, you may not query your mark.
- Note well: submitted code that does not run or does not pass standard test cases will result in a **mark of zero**.
- **Any plagiarism, academic dishonesty, or falsifying of results reported will get a mark of 0 and be submitted to the university court**.

.