UCT CSC3002F 2019

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

(**version 2**)

Lecturer: Michelle Kuttel

Due dates:

Tues 30 April 2019, 9am (Part II)

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 second part is described below.

Part II: Dish washing with semaphores

In this simulation, you will correctly implement a simple simulation of dish-washing in a kitchen. There are two roles: washer and dryer, with a shared rack between them for the wet dishes. The washer washes dishes one-by-one, adding them to the rack for drying. The dryer removes the wet dishes from the rack one-by-one to dry them. There are several synchronization constraints to enforce to make this system work correctly:

- Only one person should access the shared rack at any point.
- When the rack is empty, the dryer must wait.
- When the rack is full, the washer must wait.
- The dryer cannot dry a dish before the washer has washed it.

1.1 Code skeleton: dishWashS

You **must use** and extend the `dishWashS` **package provided**, correctly implementing the `WetDishRack` class (and all the methods), so that the `CleaningDishes` class will execute correctly, **always**. **Do not alter** the `CleaningDishes` or the `Dryer` and `Washer` thread 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 `CleaningDishes` is:

- java dishWashS.CleaningDishes 6 3 100 100

```
---Washer is washing dish #1
---Washer added dish #1 to rack.
---Dryer removed dish #1 from rack
---Washer is washing dish #2
---Washer added dish #2 to rack.
---Washer is washing dish #3
---Washer added dish #3 to rack.
---Washer is washing dish #4
---Washer added dish #4 to rack.
---Washer is washing dish #5
---Dryer is done with dish #1
---Dryer removed dish #2 from rack
---Washer added dish #5 to rack.
---Washer is washing dish #6
---Dryer is done with dish #2
---Dryer removed dish #3 from rack
---Washer added dish #6 to rack.
---Washer is DONE.
---Dryer is done with dish #3
---Dryer removed dish #4 from rack
---Dryer is done with dish #4
---Dryer removed dish #5 from rack
---Dryer is done with dish #5
---Dryer removed dish #6 from rack
---Dryer is done with dish #6
---Dryer is DONE.
```

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**.

.