Operating Systems **CSI3131** ***Assignment 2*** *Winter 2019*

Semaphores/inter-thread synchronization

You must submit your assignment on-line with Virtual Campus. This is the only method by which we accept assignment submissions. Do not send any assignment by email. We will not accept them. We are not able to enter a mark if the assignment is not submitted on Virtual Campus! The deadline date is firm since you cannot submit an assignment passed the deadline. You are responsible for the proper submission of your assignments and you cannot appeal for having failed to do so. A mark of 0 will be assigned to any missing assignment.

Assignments must be done individually. Any team work, and any work copied from a source external to the student (including solutions of past year assignments) will be considered as an academic fraud and will be forwarded to the Faculty of Engineering for imposition of sanctions. Hence, if you are judged to be guilty of an academic fraud, you should expect, at the very least, to obtain an F for this course. Note that we will use sophisticated software to compare your assignments (with other student's and with other sources...). This implies that you must take all the appropriate measures to make sure that others cannot copy your assignment (hence, do not leave your workstation unattended).

Goal: Practise semaphore usage.

Posted:
Due:

Description (Please read the complete assignment document before starting.)

SIMULATING AN HARBOUR

Consider an Harbour used for launching international tourist ships. The Harbour has 2 launching/arriving docks. There are 4 ships offering trips to 2 possible destinations. The ship destination is determined by the launching/arriving dock used. Each ship is capable of taking 2 passengers, and can go to any of the destinations. At the beginning, all ships start with zero passengers. There are 10 passengers. The behaviour of ships and passengers is captured in the following pseudo code:

Ship:

```
for (;;) {
        Wait until there is an empty launch/arriving dock.
        Land at the Harbour on an empty dock.
        The passengers currently in the ship (if any) leave the ship.
        Announce that the passengers can board to the destination
              determined by the dock used.
        The waiting passengers sailing to that destination board the ship.
        The ship is launched when it becomes full.
        Wait for random time 500..2000ms /* sailing in the water */
}
```

Passenger:

```
for(;;) {
        /* Makes some money to afford another ship. Note the ratio
              between work and leisure. */
        Wait for random time 0..700 ms
        Choose a random destination 0,1.
        Arrive to the Harbour and wait until there is an ship going to that
              destination, then board it.
        Enjoy the ship.
        Leave the ship.
}
```

Correctness specification

The following conditions must be satisfied in order for the simulation to run correctly:

- Each arriving/launching dock can at any time hold at most one ship. All other ships are in the water doing the ship or waiting for landing.

- At any moment, there are at most 2 passengers in an ship.

- A passenger can board only an ship sitting on the launch/arriving dock, after all passengers returning from a ride have already left, and after the ship has announced boarding.

- An ship launches into the water if and only if it is full of passengers (i.e. 2 passengers).

- A passenger can leave an ship only after the ship has landed back at the Harbour.

- The boarding starts only after the last passenger has left the ship.

- Other minor restrictions, which follow from the comments in the provided skeleton code.

Your goal:

Take the skeleton code and complete the classes defining the behaviour of the ships and the Harbour (the passenger class is provided complete). The provided code already creates and initializes the threads simulating the ships and the passengers, as well as the object representing the Harbour. Your goal is to provide proper synchronization between them, satisfying the conditions for correct simulation. Use only semaphores (use the Semaphore class) to achieve the needed synchronization.

Hints:

- Take a look here for many semaphore problems and solutions.

- Note that the number of passengers, destinations, ships and ship capacity has been chosen in such a way that a deadlock will not occur in a correct implementation – there are enough passengers to always fill at least one of the ships waiting for launching.

- On Unix/Linux systems (and on some Window systems as well), the following commands might help you with debugging/verifying proper execution:

    1. Launch the program with the output redirected to a log file, i.e. “java Assignment2 > assignment2.log”

    2. To see what passenger 5 did, type “cat assignment2.log | grep “Passenger 5”

    3. To see what ship 3 did, type “cat assignment2.log | grep “Ship 3”

    4. This way, you can rather easily see if your solution does not work yet.

5. You can figure out for yourself other ways to filter out useful information from the log file, or simply study the log file to see whether everything worked as specified

To submit your solution:

Submit single file assignment2.java, containing your solution. You will get partial marks for partial solutions. Be sure to include your name and student number at the start of the file.

Marking Criteria

Compiles and runs correctly by fulfilling the conditions below [100pts]. If any of the points below fail to be implemented; deduct 5 points for each non-implemented part.

If it does not compile and/or run correctly; check the conditions below one by one:

· Each arriving/launching dock can at any time hold at most one ship. All other ships are in the water doing the sailing or waiting for arriving. [15]

· At any moment, there are at most 2 passengers in an ship. [10]

· A passenger can board only a ship sitting on the launch/arriving dock after all passengers returning from a ride have already left, and after the ship has announced boarding. [15]

· An ship launches into the water if and only if it is full of passengers(i.e.2 passengers).[10]

· A passenger can leave a ship only after the ship has arrived back at the harbour. [10]

· The boarding starts only after the last passenger has left the ship.[10]