Assignment 1 – COEN 346
Deadline: November 7 by 11:59pm
Weight: 7%

Submission instructions:

- Copy all Java files and your report file in a folder and put your student id(s)_(assignment number) as the folder name.
- Zip the folder and Submit it via Moodle.
- If you are working in pair put the student id of both students as the name of the Zip file.

Dining Philosophers

1) Source Code

There are five files that come with the assignment. A soft copy of the code is available to download from the course web site. This time the source code is barely implemented (though compiles and runs). You need to complete its implementation.

2) File Checklist

Files distributed with the assignment requirements:
common/BaseThread.java – unchanged
DiningPhilosophers.java - the main()
Philosopher.java - extends from BaseThread
Monitor.java - the monitor for the system

3) Background

This assignment is a slight extension of the classical problem of synchronization – the Dining Philosophers problem. You are going to solve it using the Monitor synchronization construct built on top of Java's synchronization primitives. The extension refers to the fact that sometimes philosophers would like to talk, but only one (any) philosopher can be talking at a time while they are not eating. Additionally, a philosopher can be talking for a limited time (please see below for details). While one philosopher is talking, none of the others philosophers can sleep; however, they can be eating or thinking.

4) Tasks to be done

Make sure you put comments for every task that involves coding to the changes that you've made. This will be considered in the grading process.

Task 1: The Philosopher Class

Complete the implementation of the Philosopher class, that is all its methods according to the comments in the code. Specifically, eat(), think(), talk(), and run() methods have to be implemented entirely.
Non-mandatory hints are provided within the code.

Task 2: The Monitor

Implement the Monitor class for the problem. Make sure it is correct, **deadlock- and starvation-free** implementation that uses Java's synchronization primitives, such as wait() and notifyAll(); no use of Semaphore objects is allowed. Implement the four methods of the Monitor class; specifically, pickUp(), putDown(), requestTalk(), and endTalk(). Add as many member variables and methods to monitor the conditions outlined below as needed:

1. A philosopher is allowed to pickup the chopsticks if they are both available. That implies having states of each philosopher as presented in your book. You might want to consider the order in which to pick the chopsticks up.
2. If a given philosopher has decided to make a statement, they can only do so if no one else is talking at the moment. The philosopher wishing to make the statement has to wait in that case.

Note: You need to implement a specific strategy to prove that your solution is starvation-free.

Task 3: Variable Number of Philosophers

Make the required changes such a way that your program accepts a positive integer number (>=3) from the input (Scanner), and spawns exactly that number of philosophers instead of the default one. If the input is not a positive integer, the program reports this fact to the user by printing a warning message as in the example below and continues prompting the user for a positive number (it should be at least 3):

"entered negative number" is not an acceptable number. Please enter a positive number (bigger than or equal 3)

Then the program continues its normal execution.

Task 4: Dynamic Modification of the Number of Philosophers

In this task, you are required to show whether Task 2 above can be modified so that the number of philosophers can be changed randomly in the middle of the task; that is philosophers should be allowed to leave in the middle or new philosophers can join the table. If this is feasible, implement the changes. If not, you should comment clearly why this is not possible.

Task 5: Additional Condition

Assume two pepper shakers are placed on the table for the philosophers to use while eating. You are required to add another mutual exclusion for the sharing of these two pepper shakers between only eating philosophers.

5) Submitted Documents

The assignment can be done in a group of two students. The deliverable consists of a **well-commented code** and at **least on page report specifying the high-level description of the code** (description of the methods/functions/threads and the flow of the program).You also need to explain the synchronization solution and the starvation-free method that you used and also discuss the rationale behind them. This assignment will take 7% of your total mark for programming assignments. Also for this assignment 80% of the mark is dedicated to your code and 20% to your report.