CSCD 467 Concurrent Systems Fall 2013 Lab4
Alternating N Threads Using Conditional Wait in Monitor

Rules: Your code must use Java Language. If your program shows a compilation error, you get a zero credit for this lab assignment. To avoid compatibility issues, I encourage you to upgrade your JRE to latest version of SE 1.7 or 1.8.

Submission: Wrap up all your java files and a ReadMe text file into a single zip file. Name your zip file as *FirstInitialYourLastName*CSCD467Lab4.zip. For example, if your legal name is Will Smith, you should name your zip file as wSmithCSCD467Lab4.zip.

You are required to submit the ReadMe text file along with all your java code. In the ReadMe file you should put your legal full name, description about how to compile and how to run your program on command line tool. An example of ReadMe file should look like the following:

```
(This only serve as an Example. Your ReadMe file should contain the similar content.)
Name: Will Smith
Description: unzip the submitted wSmithCSCD467Lab4.zip, you get a folder named
smithLab4.
To Compile:  cd into folder smithLab4,
        javac *.java
To Run
        java myLa5   4    1   1000000
```

Before you leave the laboratory, please show the TA or the instructor how your program works, they will give you a score for this Lab assignment.

For archive purpose, please also submit your single zip file on EWU Canvas by following CSCD467-01 Course → Assignments → Lab4 → Submit Assignment to upload your single zip file.

Problem Description:
You are required to design and implement a program that can alternate N threads in a round-robin fashion. You are required to use the conditional wait feature in a Monitor object in parallel programming. That is, all shared data and conditional variables are encapsulated in an object. Access to shared data is protected with mutual exclusion, namely using the synchronized keyword in Java. You have to implement the features listed in the following.

1, You have to specify the number of alternating threads (***N***) as a parameter using Java command-line arguments.

2, You have to create N threads in you main program. Each thread will print out a total of 10 messages before it terminates. But threads take turns to print for each round of message output. For example, if N=3, Thread 0 prints message 1 → Thread 1 prints message 1 → Thread 2 prints message 1 → Thread 0 prints message 2 → Thread 1 prints message 2 → Thread 2 prints message 2 → Thread 3 prints message 1 ..... and repeat this pattern.

3, You have to apply wait and notify (or notifyAll) methods within your monitor class.

Correct Program Output

A correct output looks like the following if N = 3.

```
1st Message from thread 0.
1st Message from thread 1.
1st Message from thread 2.
2nd Message from thread 0.
2nd Message from thread 1.
2nd Message from thread 2.
3rd Message from thread 0.
3rd Message from thread 1.
3rd Message from thread 2.
4th Message from thread 0.
4th Message from thread 1.
4th Message from thread 2.
5th Message from thread 0.
5th Message from thread 1.
5th Message from thread 2.
6th Message from thread 0.
6th Message from thread 1.
6th Message from thread 2.
7th Message from thread 0.
7th Message from thread 1.
7th Message from thread 2.
8th Message from thread 0.
8th Message from thread 1.
8th Message from thread 2.
9th Message from thread 0.
9th Message from thread 1.
9th Message from thread 2.
10th Message from thread 0.
10th Message from thread 1.
10th Message from thread 2.
```