CSCD 467/567 Lab3

Fix the Faulty Solution one to Alternating Threads

**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*CSCD467Lab3.zip. For example, if your legal name is Will Smith, you should name your zip file as wSmithCSCD467Lab3.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. An example of ReadMe file should look like the following:

**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 AssignmentsLab3 Submit Assignment to upload your single zip file.**

**Problem Description:**

You are required to achieve these features in this Lab. **Please use the techniques that we have covered so far. (interrupt and wai()/notify are not an option here.)**

1, we have to use two threads T1 and T2 in the main thread.

2, starting with thread T1, T1 outputs a message, then thread T2 outputs.

3, two threads alternate till each finishes 25 message outputs.

A faulty solution was given and discussed in the last lecture. You are required to work on the faulty solution and fix it in this lab. After you fix the solution, please also answer the following questions about your work.

1, Why the provided solution does NOT work? Which statement(s) cause the program to hang up?

T1 Gets the lock, prints, gets the lock again, and then waits in the while loop for the condition to change, but it never will.

2, How did you fix that problem?

By making another class with a boolean saying which thread's turn it is with a getter and setter.

3, How does your changes fix the problem? ( hints: with regard to the lock )

By making whose turn it is dependant on a syncronized getter and setter in a seperate class the lock is retreaved and released in every iteration of the while loop.

**Correct Program Output**