Seneca College of Applied Arts & Technology **August 1, 2022**

SCHOOL OF INFORMATION & COMMUNICATION TECHNOLOGY

JAC444 **Due date: August 9, 2022**

Workshop 10

Description:

This assignment lets you practice Multi-Threaded programming in Java and includes concepts such as Threads, Guarded Blocks, and Synchronization.

Give a solution to the following problem:
You want to help your friend with some money. You and your friend have access to a shared account, but in a "Strange Bank."

The Strange Bank has the following rule for the accounts: once you deposit some money in some currency (to an account with an initial balance of zero,) you are not allowed to deposit money in another currency unless the account balance would be zero (again.)

You have one Dollar, two Euros, and three Pounds and you want to transfer these amounts to your friend. Write a Java program that simulates you depositing to and your friend withdrawing from the shared account.

Hints: You and your friends are Java `Thread`s that try to access the bank account concurrently (you to deposit, your friend to withdraw). You must communicate through the `wait` and `notify` methods.

*Make sure that **all the deposit and withdraw actions between you and your friend should happen to be in** amounts of one **(at a time)**. **There should be 6 transactions done by either you or your friend!***

***Also, there shouldn't (necessarily) be alternate ordering between you and your friend, depositing and withdrawing**! As an example**, one possible run** for (just) 3 Pounds could be:*

1. *You have the CPU you could deposit 1 pound,*
2. *You would continue by depositing another,*
3. *CPU could now switch to your friend; they could come and withdraw 1 pound,*
4. *CPU is (again) back to you; you deposit your last pound,*
5. *CPU is (again and eventually) back to your friend; they withdraw another 1 pound,*
6. *They continue by withdrawing the last, and then, that's done! (For the whole solution including 1 Dollar, 2 Euros, and 3 Pounds, it would be the same idea as stated here.)*

For a full mark, you need to provide me with at least two different runs of your code regarding the screenshots you attach.

This is a classical problem in thread theory called "Producer/Consumer" in which there is a container that accepts only one object at a time. The producer wants to put in the container as many objects as it produces but must wait until the consumer consumes the already-shared object from the container (you can implement your solution based on the last example/video discussed last week!)

Marking Criteria and Tasks:

Please note that you should:

a- have an appropriate indentation.
b- have proper file structures and modularization.
c- follow Java naming conventions.
d- document all the classes properly.
e- not have debug/useless code and/or file(s) left in assignment.
f- have good intra and/or inter-class designs.

in your code!

- Task: Developing and running the desired solution: **(you should submit your source code - just individual .java files, and screenshots which demonstrate all the different/possible ways your code runs): 5 marks.**

Deliverables and Important Notes:

- You are supposed to **submit your solution online on Bb by the end of the day on Tuesday, 9th of August 2022.)**

- Please note that you would be allowed to **submit just once**, so please **be super careful and double-check before you hit submit.**

- **There would be a 20% penalty for each day (or part of it,) in case you submit late!**

- Remember that you are encouraged to talk to each other, the instructor, or anyone else about any of the assignments, **but the final solution may not be copied from any sources**.