***Third partial assignment for hw2.***

***Due date: November 20th by 11:59pm***

Grade: 10% of the total grade (which is one third of the total 30% for hw2)

In this part of hw 2 the students should implement the deposit and withdraw agents as threads, using

condition synchronization to properly block a withdraw agent if the balance is not enough

for the amount of a withdrawal.

Also, AgentView and AgentController should be implemented, placed in view and controller packages.

The agent classes should be placed into model package.

Classes for the agents should implement Runnable interface and extend AbstractModel to be able to send

events to their AgentViews.

They can be started as threads either via low level API of the Thread class or thread pools in library

java.util.concurrent

Account class should be modified to include a specialized "autoWithdraw" (or appropriately named in some other way) method that

is invoked by Withdraw agent and that implements condition synchronization instead of throwing an exception.

Also "deposit" method should be edited to call "notifyAll" once the balance has been updated.

Please refer to the main homework 2 file linked off the course website for the description of

the agent behavior and AgentView layout.

Two simplifications for the agent behavior:

1. You do not need to implement manual pausing for the agents.

They will only have "Running" and "Blocked" states.

This also means that AgentView does not need "Pause" and "Resume" buttons.

2. You do not need to implement regulated frequency of agent operations. Instead you can put "Thread.sleep(300)" into

the loop inside of agents' run methods to make them go slower (this makes them sleep for 300 milliseconds on every iteration;

you can use different reasonable duration).

This also means the AgentView does not need “Operations per second” text field.

For unit testing:

1. Write a JUnit test class AccountThreadSafetyTest that creates an instance of Account class, instances of DepositAgent and

WithdrawAgent and starts the agents as threads. The agent threads should run their loop inside the "run" method

a predefined number of times.

The threads should be "joined" (method join on Thread class) before checking the end result (to be sure they finished).

The end result should show that the balance has the expected amount in it after running the agent threads.

2. Write a JUnit test class AutoWithdrawBlockTest that tests that "autoWithdraw" method in Account class properly blocks

if the withdraw amount exceeds the balance.

For acceptance testing:

1. Check that several deposit or withdraw agent windows can be open and functioning on a single account.

2. Check that withdraw agents properly block if the balance is not enough for the withdraw amount.

3. Check that withdraw agents properly wake up from blocking if a sufficient amount is deposited into the balance on which they were blocked.