Assignment 4: Monitors in Java

I. Introduction

Remember from Unit 3.3 that monitors can be simulated in Java by having all methods which access to the resource synchronized, and methods exported (public).

Besides, by default Java only provides 1 Condition for queueing threads, but we can create objects of type Object and queue on them threads of the same type. Thus, using primitives in Java we can s simulate Monitors with:

- Synchronized blocks
- 1 implicit condition variable per object
- Operations to block and unblock threads on a given object: ***`wait, notify, notifyAll.`***

However, from Java 1.5 we can use class java.util.concurrent.locks.ReentrantLock, which makes it easier to create and use Condition variables.

http://download.oracle.com/javase/1.5.0/docs/api/java/util/concurrent/locks/reentrantlock.html

In this case, in order to simulate Monitors we need:

- Explicit blocking:
  - **final ReentrantLock** l **= new ReentrantLock(true);**
  - l.lock(); l.unlock();
- Condition variables:
  - As many as you want
    - **final Condition** condition1 **= l.newCondition();**
    - **final Condition** condition2 **= l.newCondition();**
- Operations to block and unblock threads.
  - condition1.await();
  - condition1.signal();
  - condition1.signalAll();

II. Solve the following tasks:

A. The one-lane bridge

Due to budget cuts, a road built upon a bridge has a single lane, although it is a two-ways road and cars may come from both directions. Therefore, some kind of synchronization is necessary to prevent cars from a collision. To illustrate the problem, we provide a solution in which no synchronization is performed. Your task is to add synchronization to cars.

The "unsafe" solution can be download from Campus Virtual. Run it. The user interface is simple, using the two buttons you can add cars going to the right or left. You will see as cars collide in the bridge when there are more than one in different direction.

It is not necessary to understand the graphics code to solve the problem. All you need to know is that red cars approach from left to right, and call the control.redEnters () method when approaching the bridge (for permission), and they call method. control.redExists () when it leaves. Blue cars do the homologue actions: blueEnters () and blueExits ().

The graphics files must be placed in the working directory in a folder named "image", or you may also change the path in the source code.

Here is an instance of the class that handles TrafficController synchronization, as you can see the supplied class has empty implementations of the four methods,

Your task is to change this class into a monitor to coordinate the cars so that:

Safe, Efficient and Fair Bridge

No cars waits forever to pass (Although there is heavy traffic in both directions) → fairness. Do not use basic synchronization tools provided by the Object (synchronized wait and notify) class, we recommend using java.util.concurrent utilities. In particular, use ReentrantLock. Using it, explicitly implement the lock of the object (do not use synchronize) and different condition variables.

The solution will establish a constant MAX which tells us that in case of cars waiting on both sides of the bridge, not over MAX cars will pass over without changing the turn.

You can create a private method CheckTurn within the monitor to be called by cars before entering the bridge and change the turn if MAX cars have already passed.

The solution must be efficient: the cars do not cross one by one; that is, several cars in the same direction can move across the bridge at the same time → efficiency

http://docs.oracle.com/javase/7/docs/api/java/util/concurrent/locks/ReadWriteLock.html

http://docs.oracle.com/javase/7/docs/api/java/util/concurrent/locks/Condition.html

B. The crosswalk

Traffic control is again within the TrafficControl class. You must create the synchronization necessary to avoid accidents:

1. Safe crosswalk

Add necessary synchronization to prevent accidents from occurring. You can do this by using the basic synchronization tools provided by the the Object class (synchronized, wait, notify, ...)

2. Priority for pedestrian (MONITOR)

In this case, besides avoiding accidents the program must give priority to pedestrians over cars. That is, if there are pedestrians crossing the streets and cars arrive, then cars must stop and give way to pedestrians.

3. Priority for pedestrians but avoiding cars starvation (MONITOR)

If the flow of pedestrians is high, the traffic can collapse since cars never go on. In order to solve that, only 5 cars should wait per each sequence of 10 pedestrians crossing the street.

III. Evaluation

We will use **3 lab sessions** to solve this assignment.

Remember you will be evaluated using this rubric:

Respect to the global interests of the group	The group is not well organized. Members did not share the work to carry on. ***0 points***	Some problems arised, and many were solved. ***1 points***	No problems have affected the performance of the group, or were successfully solved, ***2 points***	
Correctness of the assignment	It was not solved correctly at all. ***0 points***	Only a few points were right ***1 points***	Mostly correct ***3 points***	Perfect ***4 points***
Correctness of answers to questions about the code (Algorithms and structures used)	No answers/wrong answers ***0 points***	A few correct, or correct but only member answered ***1 points***	Both members gave mostly correct answers. ***2 points***	Both members answered correctly to all questions. ***4 points***