CS1760: Multiprocessor Synchronization Due 30 November 2023

Project 3: Yard

Professor: Maurice Herlihy *TA: cs1760tas@lists.brown.edu*

1 Introduction

For this programming assignment, you will need to implement a Yard protocol that satisfies certain conditions in order to allow Alice and Bob's pet to use it and explain why your solution satisfies these conditions. The Stencil code for the assignment is located at: HERE

Both Alice and Bob have now acquired multiple pets. Their pets still do not get along, but they must share a yard. We will build a monitor to help them out. There is a single Yard resource that must satisfy the following:

- Mutual exclusion: pets of different owners may not occupy the yard simultaneously
- Starvation-freedom: every pet who wants to enter the yard eventually enters, assuming a fair scheduler. (A fair scheduler is one that schedules longest-waiting threads to run first.)

The protocol is implemented via the following four procedures:

- eneterAlicePet() is called whenever Alice's pets want to enter
- leaveAlicePet() is called when one of Alice's pets leaves the yard
- enterBobPet() and leaveBobPet() do the same for Bob's pets

Implement this class using locks and condition variables in Yard.java. Additionally, we want a README file that explains why it satisfies mutual exclusion and starvation-freedom regardless of the number of threads used. Additionally, either prove that your design works even with an adversarial scheduler (one that deschedules threads in any way that you don't want it to) or propose a scenario in which an adversarial scheduler thwarts your starvation-freedom guarantee.

Once you copy the stencil code into your directory, you'll be ready to go! The files you need to modify are:

- Yard.java
- TestYard.java
- additional testing files, such as BasicTest.java if needed

Note that you are only been given a test template (in BasicTest.java) this time, and you are expected to add to it and even modify its API if needed. A portion of your grades us reserved for testing.

There will be an autograder for your implementation, and will manually grade your answers in README.md and your tests.

Here are some tips:

- Note that we expect you to submit a Yard using a strategy that runs with our setup - this means it has to run correctly with a finite number of threads. Depending on your implementation, this may cause a problem that you need to work around. Come to TA hours if you're not sure how to resolve the issue!