

## Project 2: Yard

*Professor: Maurice Herlihy**TA: James Scherick*

## 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`

Note that a portion of this assignment's total score is reserved for testing. Passing basic functionality tests like what starts out in `TestYard.java` will reward you with some points; however we also expect you to write any additional tests you deem necessary.

Failing basic functionality tests does not necessarily imply that you will receive no credit for the assignment. At the same time, the course staff will be unable to conduct a rigorous inspection of non-functional code to award partial credit.

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!