REPORT

NDVSYE003 PCP2 – Concurrency

The Task:

A medley simulation program has been provided for this assignment as well as a working demo of how it should perform, however the program provided has many concurrency issues and race conditions that must be corrected in order to fix the code so that it run as intended.

The main goal of this Assignment is to learn how to find and protect a parallel program from concurrency and thread safety issues.

Major Concurrency Errors Identified:

- SwimTeam.run() method - all swimmers in a single swim team are started at once. Should be waiting for eachother.
- GridBlock.moveTowards Method needs the getters for getting the coordinates of each swimmer to be synchronized to avoid moving into the same block as another swimmer.
- The Swimmer.dive() method should wait until all backstroke swimmers are at the starting block
- Consecutive swimmers in the same team should wait until the previous stroke exits the pool

Other issues:

- Start button in *medleySimulation* is not implemented to start simulation.

Changes made and why:

- Constructed the functionality of the start button by using the notify() method and wait() method in MedleySimulation.
- Synchronised GridBlock get method to ensure that blocks are only accessed by one swimmer at a time preventing a data race.
- Corrected instances of spinning in StadiumGrid(While loops that process nothing continuously waiting for its looping variable to become false). This was corrected by simply adding a “wait()” inside the while loops and notifying the wait methods using a notify() in the GridBlock release method.
- A CyclicBarrier was used to ensure all back stroke (black) swimmers wait at the starting block until all other back stroke swimmers arrive. This is done so that all teams start the relay at the same time.

- To allow the swimmers to relay (current swimmer out and next swimmer in) and avoid missed signals stemming from conditional wait and notify methods, an array of CountDownLatch(s) were used for each team e.g latches[team] = new CountDownLatch(1). This was used to communicate to swimmer threads to dive() and begin swimming correctly

Bonus Elements

First, second and third:

New JLabels and tracking variables were created in the FinishingCounter and CounterDisplay Classes. The constructor for CounterDisplay was modified to allow for the displaying of second and third.

Conclusion

This Assignments goal has been achieved. The rules of the simulation are all maintained and the additional feature of first second and third has been added. The concepts of thread safety and Data races have become more familiar to me and parallel and concurrent programming is a technique which I am able to implement in my programming.