Maneesh Budda
mrb204

Constructive Review of Andrew Megeath's Final Project

Andrew's Final Project focuses on understanding how the Peterson's Algorithm works and has written code for it using the Java programming language. What Andrew was attempting to understand and was testing by writing Java code for Peterson's Algorithm was to see how the algorithm for 2 processes would not function correctly for 3 processes. The Perterson's algorithm is a concurrent programming algorithm for when two processes are in the critical section at a point in time, which allows for the two processes to share a resource without conflicting with each other by only communicating through shared memory.

Andrew splits the classes associated with simulating CPU processing and those associated with running experiments with Peterson's Algorithm into two packages that being the Processing package and Demos package respectively. The Processing package includes several files: CPU.java, WaitStep.java, ProcessStep.java, Process.java, and ActionStep.java, and the Demos package includes files that test various process scenarios. This is well organized and structured on his part when trying to properly execute this algorithm.

In his project, Andrew shows us the outputs of the Perterson's Algorithm in several different scenarios/cases. The scenarios that Andrew covers include No Locks 2 Processes, Peterson's 2 Processes, Peterson's 3 Processes, Peterson's 3 Processes Incorrect Fix, and Peterson's 3 Processes Incorrect Fix Failing, which are all stored in the Demos package. In the No Locks 2 Processes demo, Andrew has shown that when 2 processes with critical sections run simultaneously without having a locking system, the sections overlap, which can cause issues with the program. In the Perterson's 2

Processes simulation, Andrew was able to show that 2 processes with critical sections run simultaneously without having a locking system, which results in the critical sections of each process to be separate as seen by the critical section output of each process being isolated. For Perterson's 3 Processes, Andrew demonstrated that when 3 processes are run and enter the critical sections. Only process A gets finished, and the other 2 are seized as a result of deadlocking after termination of process A. In the next situation, Peterson's 3 Processes Incorrect Fix, Andrew fixes the deadlocking that occurs in the previous scenario by keeping processes locked until they are unlocked. In the final simulation, Peterson's 3 Processes Incorrect Fix Failing, Andrew shows us how the fix from the previous scenario fails resulting in neither A or B's second critical sections to be executed after C's first critical section approaches termination, which causes processes A and B to deadlock.

I believe that Andrew has done a great job of showing the Perterson's algorithm at work when each of the processes that being processes labeled A, B, and C have critical sections in a situation with locks and no locks and also with incorrect fixing and the fix failing. What Andrew can do to further improve his project is by setting up flags in his program to show us lock and unlock values and also to check which thread is entered to run the processes. Also another thing Andrew can do to enhance his project and outputted results would be to show us how the processes communicate between each other through shared memory and also show us two or more processes working on the same data.