Elizabeth Oyebade

MET CS526-O2

Process Priority Queue Simulation Term Project

04/26/2022

Description of Data Structures:

The first (main) data structure of this project is the Priority Queue *processPriorityQueue* which is utilizing a comparator to compare the priorities of the process. Within the program, processes are being brought from its holding area which can also be known as the *dataProcess* into the queue based on the arrival time of each process. To prevent a process from lacking, a maximum wait time is being implemented but if a process has waited longer than the predetermined maximum wait time (30) then the process's priority will be decreased by 1. The second data structure used in this program is the *dataProcess* which is an array list for storing the process's objects that's been read, it's storing the information about a process that's needed to be executed in the *processPriorityQueue*. A process is a user-defined class that holds a process and each process has an *ID*, *priority*, *arrival time*, *duration*, the *waiting time*, and overall, the *total wait time*. The duration of the process is the amount of time it takes to completely execute the process. The *dataProcess* is moved from the *processPriorityQueue* when the process's arrival time has reached the program. When this happens, if there are no other processes currently running in the queue, the process arrival time is either being executed immediately, it's waiting till both the currently running process is completed and there are no other processes with a low priority in the queue. The last data structure used in this program is the *waitTimeArray* which is used to keep track of the wait time and average wait time. This data structure's main purpose is to keep track of each process's wait time in the queue. The wait time of each process that's stored in the waitTimeArray will be averaged out to find the average wait time of each process when both the dataProcess and processPriorityQueue are empty.

Observations and Learnings:

The main thing I learned from this project is to research what you do not understand, ask the professor or facilitator questions, and not code in one sitting but instead, spend up to 2-4hrs coding parts of the problems to reach a solution. Whenever I encountered a bug within the code, I looked through the code from the start to the end to find the solution to the problem I encountered and to

make sure that nothing is broken with the code whenever I added a new code. An interesting observation I had was when I manipulated the original process input file. When I added a new process id, priority, duration, and arrival time with the same maximum wait time of 30, the total wait time increased from 391.0 to 453.0 and the average wait time increased from 39.1 to 41.18 but process 10 was still the last process to be removed from the queue. When I increased the max wait time to 80, the process order that was being executed changed, and with that, the total wait time increased from 391.0 to 402.0 and average wait time increased from 39.1 to 40.2 and the last process to be removed from the queue was processed 5, not 10. This shows that a process with a higher maximum wait time can be “starved” if a process with a lower priority keeps arriving and process 5 shows that the arrival time was 40 but was not executed till 170 with a higher maximum wait time.