CS 526 Final Project Documentation Hoon Kim

**Data Structures Used for the Project**

* ArrayLists (of ints, int arrays, and Process Objects)
  + Int ArrayList used to store wait times of process when they were executed
  + Int Array Arraylists were used to store process attribute values read from file
  + Process Object ArrayLists were used to:
    - For initial storage of process objects as database “D”
    - temporarily store process objects when their priorities got updated
* Arrays (of ints and strings)
  + String arrays: text data read from the process scheduling input text file was initially stored as string
  + Int arrays: process data stored in the string array mentioned above was cast to ints that were used in constructors for Process Objects
* PriorityQueue (or Process Objects)
  + Used to store processes that in their priority order so the program would execute them by priority, not first-in-first-out

**Discussion**

* Observations & Learnings
  + I really felt the usefulness of priority queues in this application, where the program needed to be designed to run according to logic/order different from time variables
    - The key advantage of the priority queue data structure was evident during process additions to Q and priority edits.
    - As arrival time order did not necessarily correlate with priority order, the way priority queues self-sort upon an addition was very helpful. Without it, every addition to Q would have required a separate algorithm to remove all process, re-sort them, and re-insert them.
    - The self sorting of priority queues was also convenient when priorities of some processes needed to be decremented as there could potentially be cases where sort order would change.
  + The self sorting feature did make some initial attempts to edit priorities in a priority queue without a helper Arraylist challenging
    - I realized that using poll(), updating it and simply adding it back into the priority list would not work because the process that I just updated would be placed at the head of the priority queue. So the loop I wrote would iterate on 1 element of the PQ instead of every element.
    - The way around this was to use helper ArrayLists which use more space.
  + I also believe that using custom comparators for priority queues of objects is very convenient and makes adapting PQ implementations to different sorting needs very easy. Instead of editing many lines of sorting algorithm code to change the way a queue sorts itself (e.g. sort with arrival time instead of priority), I can simply edit the comparator to use a different variable, which will help decrease bugs/inconsistencies within programs.