**CPU Scheduling**

## This lab report satisfies all tasks given.

## Creation

I created my CPU scheduler simulation using C#. As an object-oriented language, the first step was to create the process class (Fig 1.). This class would contain all the variables for keeping track of the various process states, namely: the burst, arrival, completion, start, turnaround, waiting, and response time for each process.

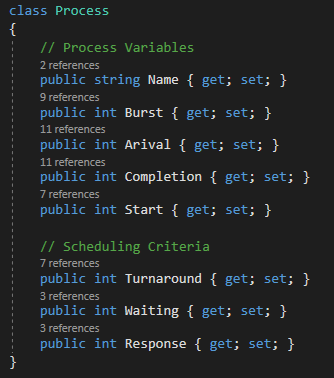


Figure 1: Process Class

Next, I created several helper methods for generating random numbers (Fig 2.) to use for the burst and arrival time during the creation of each process. As stated in the lab document, the minimum burst time was 2ms, and as such, our maximum time was 42ms. Thus in process creation, I used the random methods for each process creation for assigning the burst and arrival values, process creation happened up to N times (Fig. 3)

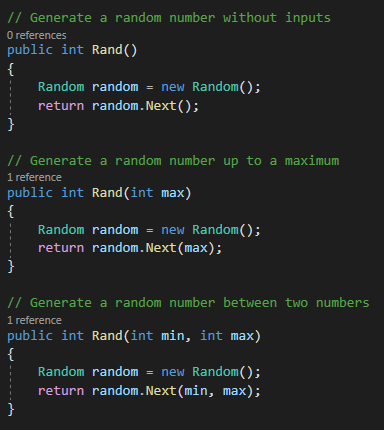


Figure 2: Random Number Helpers

The next step was to create a scheduling simulation for SJF and FCFS. To start since FCFS and SJF only differ in the ordering of when jobs are performed, I started by creating a Scheduler method (Fig. 4) which would handle all my calculations on any IEnumerable passed into the method. Then for each item in that IEnumerable, I calculated the correct values for each variable using the descriptions from class. Once they were calculated, I could use the numbers to find the averages for the turnaround time, throughput, and CPU utilization. Once that was done, I used a print method for printing each item from the sorted list and the resulting averages.

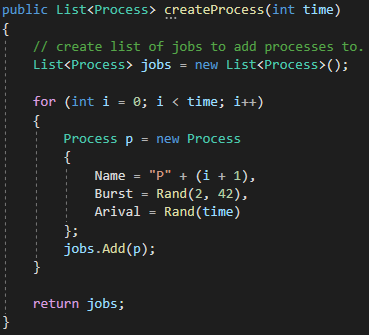


Figure 3: Create Processes

Using Linq, I sorted the Process list so they would be ordered by the Arrival variable for FCFS and created another IEnumerable in which I sorted the same list by Burst time. Once each was sorted, I could pass the resulting IEnumerable into the scheduler method. The results can be seen in Figure 5 & 6.

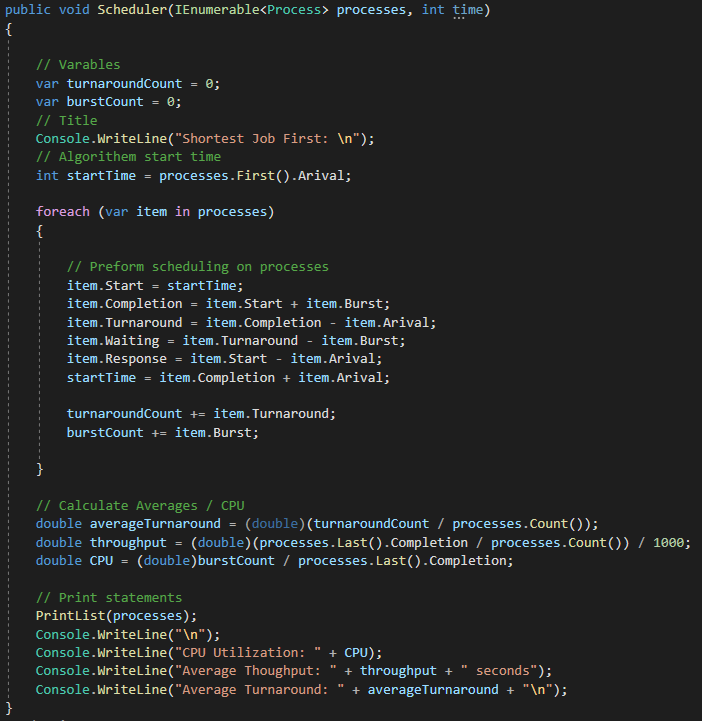
****

Figure 4: Scheduler Method

# Results

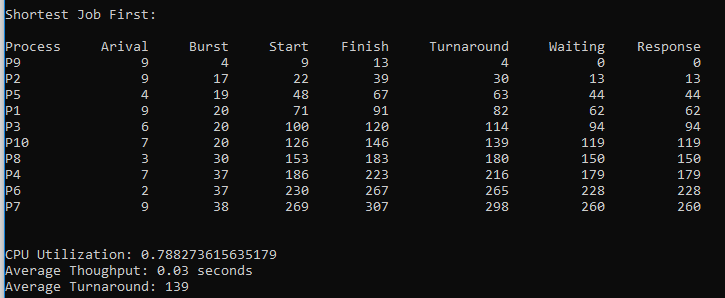


Figure 6: SJF Results

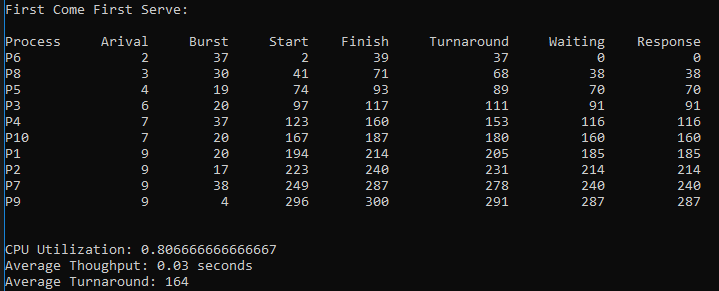


Figure 5: FCFS Results

By analyzing the results we can see that the shortest job first was better across the board, from CPU Utilization and Turnaround time both being shorter then FCFS. The throughputs where the same on this particular test and when running it with other processes Throughput was generally within a few ms of each other. As we know SJF is an provably the optimal solution, providing the shortest wait times which we can see by looking at the waiting column, The downsides of SJF being the time taken by a process must be known by the CPU beforehand, which is not possible, plus longer processes will have more waiting time, eventually leading to starvation of these processes.