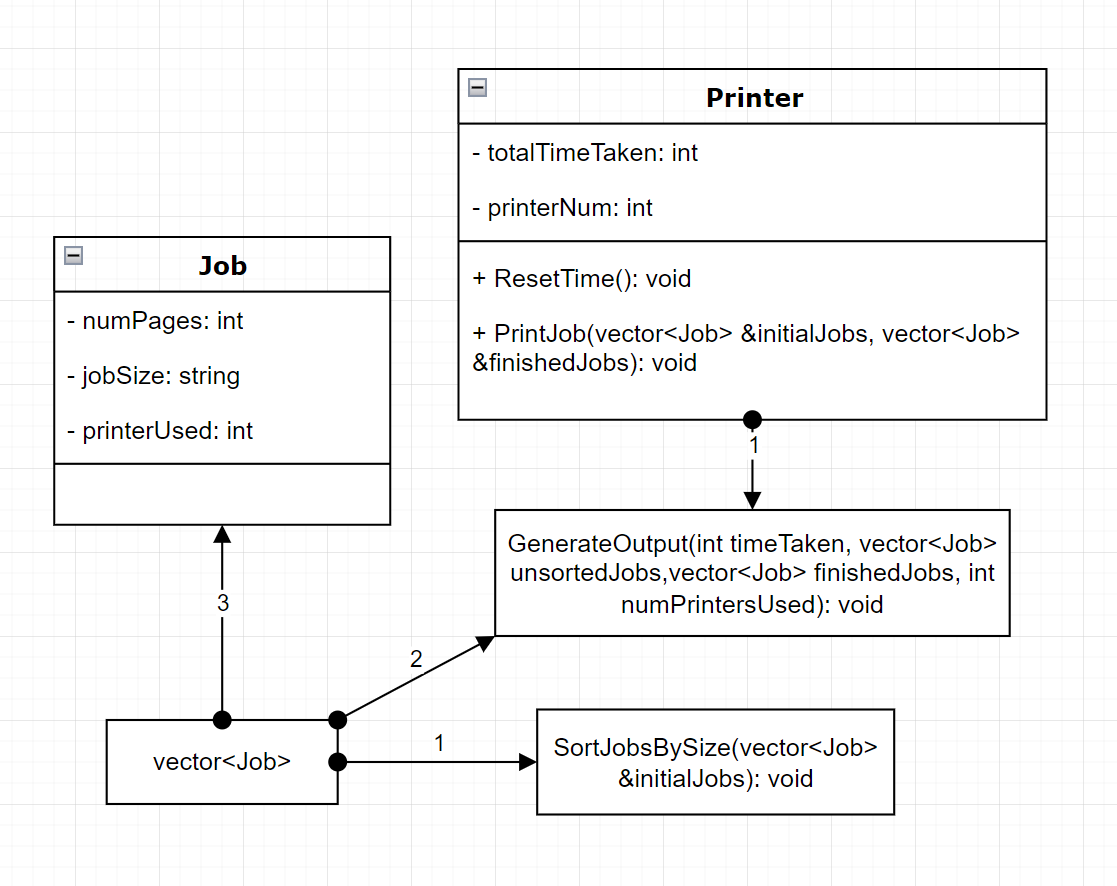
Epperson IV, Joe

December 13, 2021

**Printing Jobs Description**



I chose this project initially because I felt as though it was something I had seen before in person, and I thought it would be interesting to attempt to do this myself. When I am in the library printing things for classes, I have to send the job to the printer and then accept the job, I felt as though this would be a cool and unique challenge for me.

Initially I spent a while deciding how I was going to build this project. I was deciding what classes I would use, whether or not they needed to have child classes, what functions would need to be in these classes and what functions would be better left global, these were things I spent a while thinking about. I felt as though I wanted to have 3 or more classes, but eventually I felt like I was overcomplicating things so I settled with just two and made it very simple. One assumption I made was firstly that I would be getting the jobs more or less all at once. I felt as though that was the best way to process this information and complete the task, especially since I don’t have a constantly running and updating program. Another decision I made was that I was going to use a random number generator with modulus in order to decide how many pages would be used per job. Furthermore, I decided that I would simply output the result for 1, 2, and 3 printers at the same time using the same time in order to have a controlled comparison. Beyond that, these decisions and assumptions meant that I would not have to deal with user input and the program would depend solely on my ability to complete it.

I decided that I would create a class for Jobs and I would create a class for Printers. The Jobs class contains private variables: firstly an int which contains the number of pages for the current job, a string which contains the word either “Small”, “Medium”, or “Large” based on the number of pages, and lastly an int which contains the printer which is used when the job is done, this is initialized to zero until the job is completed. The public functions within the class contains setters and getters to be used in other functions, as well as a function void GetJob with an int parameter which takes the int and assigns it to the variable within the class and changes the size name based on the number of pages, this is useful for easy sorting later. My intention is to use the Job class along with the GetJob function in order to fill a vector with jobs, send them through a sorting function which sorts them by the order in which the jobs were received and then based on the size of the jobs. This vector will be sent to the printer and used to execute the jobs and determine the time taken for each occasion depending on the number of printers used. The Printer class has two private variables, firstly an int which contains the amount of time(in seconds) the printer has spent in total on the jobs, which increases per job and based on the number of pages. The second variable is an int which determines which printer the current one is, this will be used in order to tell the job objects which printer they used in order to later output which job was completed by which printer.The public functions contains setters and getters as well as a function which accepts the vector of unfinished jobs and finished jobs both by reference, the function completes the jobs, adds the time to the total time variable, removes the job from the unfinished jobs vector and adds it to the finished jobs vector. The only other public function is used to reset the time on the printer, this is used in between trials so that the time does not run over from using one printer to using two printers.

There are two other global functions which generally work together. The first function accepts the times each printer took as a parameter and returns the variable greatest amount of time, this is useful when using multiple printers, because whichever printer took the most time will be the entire time which was required to complete all of the jobs. The second function generates the output for the program, this has four parameters: the greatest time of the three printers, a vector of the jobs in the order in which they were assigned, a vector of the jobs in the order in which they were completed, as well as an integer which tells the function how many printers were used so that the function knows what to output. This function first takes the time given(in seconds) and converts it into hours, minutes, and seconds. It then outputs what is required in the project description; each job in the order they were assigned, followed by each job in the order they were completed as well as the printer which was used, and lastly followed by the total time required to complete the jobs. This function changes output based upon the number of printers which were used.

Lastly into the main function where I put all of these pieces together. I began by giving a random seed to the random number generator by inputting the seed with the current time, I also created a constant variable which is the size of the vectors being used, this is the 100 jobs. I then created a tempJob Job object in order to store each job before they are stored into the vector. I also created three vectors of type Job, one for the unfinished jobs, one for the finished jobs, and one or the jobs in the order in which they were assigned so that when I sort the jobs, I do not lose the original order. I then use a for loop going from 0 to 99 to fill the vector with unfinished jobs, with the random generator determining the page number. I then copied the full vector into the variable for the jobs in order of assignment, and then sorted the unfinished jobs vector using the function mentioned above. I then created three Printer objects and assigned them with their proper printer numbers, and initialized three time variables which will eventually receive the times from the three printers and be put into the function which determined the greatest of these three. The rest of the main is outputting the values based on the number of printers, I feel as though the code for this portion is quite trivial and not worth talking about.

Moving onto the BigO notation. There are 13 loops which run from 0 to 99 or 1 to 100. These are included within three functions, the function which sorts through the jobs and puts them in the proper order runs from 0 to n on three occasions, once to order the small sized jobs, then medium and large; the next function is the output function which runs from 0 to n on six occasions, within the function are two for loops which run on three occasions, these loops output the jobs based on assignment time and the jobs based on completion time, this goes for each experiment: 1 printer, 2 printers, and 3 printers; lastly the main function has 1 loop which runs from 0 to n, and three loops which run until the vector of size n is empty; the first loop is to fill the initial vector with jobs and the last three are used to print the jobs. Realistically the loop which uses two printers will run 50 times and the loop which uses three printers will run roughly 33 times, but for the worst case scenario we will call this n. With ‘n’ being 100, this program runs for 13n + some trivial number of other operations, and since the constants are irrelevant in BigO notation, my project has a BigO(n).