We made our RR and our FCFS implementation in the same file starting with our creation of our Job class. Each Job object contains a job's wait time, arrival time, length, progress, and the boolean value inQueue, whether its in the wait queue or not in RR.

We then have a parent class called Scheduler with children FCFS and RR. They share the run and getName functions.

FCFS's run implementation takes in a list of jobs, and adds them to a queue in order of arrival time. While the queue is not empty, the queue is dequeued into a variable called completed. The variable prev lets us keep track of the wait time from the previous job to calculate the wait time for the current job. We then add the wait time of the new job to the total wait time. Once the queue is empty, the total wait time divided by the number of jobs is returned. FCFS's getName function returns FCFS to distinguish it from RR when printing results.

RR's run implementation takes in a list of jobs, and creates a new empty queue. While the list of jobs is not empty, we go through each job in the list of jobs to find all jobs that have the arrival time of the current time of RR. If it does, the job is added to the queue and its inQueue value is updated to true. If the queue is not empty, the variable currJob holds the job dequeued from the queue and inQueue is updated to false. We then update the wait time by one for each job that is inQueue (inQueue = true). currJob's progress (time that the job has been on the CPU for) is updated by one. If currJob's length is equal to its progress, its wait time is added to the avgWaitTime and is removed from the list of jobs. If its not, the job is added back to the queue and inQueue is set to true. Time is updated by one after each iteration of the while loop. Once the jobs list is empty, avgWaitTime divided by the number of jobs is returned. RR's getName function returns RR to distinguish it from RR when printing results.