Part 1: Scheduling

Compile

Go to the lab02_part1 folder

- compile: `javac lab/*.java`
- run: `java lab/Main`

Design

Our implementation of the scheduler that supports the first come first served (FCFS) and round robin (RR) scheduling algorithms starts with the Job class, which represents each job in the input file.

```
public Job(int total,int arrvialTime){
  this.total=total;
  left = total;
  this.arrvialTime = arrvialTime;
  CompletionTime = 0;
}
```

We initialize a list of job called `jobs` contains all the jobs from a single input file. The job class has a attribute left which represents its remaining running and we can calculate the waiting time of a job by calculating:

```
total_wating_time += (jobs[i].getCompletionTime() - jobs[i].getArrvialTime() -
jobs[i].getTotal());
```

and the average waiting time by:

```
int average_waiting_time = (int)total_wating_time/compeleted_jobs;
```

At the end of the simulation time, we calculate the waiting time for every jobs in the readyqueue and get the average waiting time by dividing the total waiting time of all the jobs by the number of jobs in the readyqueue and completed jobs. This is the way that we get our solution.

We have an important global variable, `current_time`, which simulates the running time of one second of CPU and represents one second. We use the `current_time` to determine when the job should be added into the readyqueue and set the completion time a job when it is completed. So the program will be running under a while loop:
`while(curr_time <=Simulation_Time)`
There are two scheduling algorithms:

- First_Come_First_Serve

In the while loop of `First_Come_First_Serve` class, we check if there is any job coming at each second:

```
for (Job job : jobs) {
  if (job.getArrivalTime() == curr_time) {
    readyQueue.add(job);
  }
}
```

We add the current_time by one while processing each job in the readyqueue for one second. If the job is completed, we remove the job and start executing next job in the readyqueue. If the readyqueue is empty and simulation time is not reached, we simply add current_time by one.

- Round_Robin

In the while loop of Round_Robin class, we check if there is any job coming at each second like First_Come_First_Serve. But after executing a job for one time slice, we stop executing the job a put it back to the readyqueue and pull the next job for the readyqueue for executing. If the job is completed, we remove the job and start executing next job in the readyqueue. If the readyqueue is empty and simulation time is not reached, we simply add current_time by one.