DUE: Thursday, April 25

The purpose of this assignment is to investigate SRTF (Shortest Remaining Time First) scheduling through deterministic modeling in Java. For simplicity we are ignoring dispatch latency and assuming that each process consists of a single CPU burst.

Create a NetBeans or IntelliJ project called *SmithScheduling* (with your last name in place of *Smith*). To submit the assignment, upload the project folder so that I can open and run it in the IDE.

Write a program that calculates the execution schedule and average waiting time for a given workload using SRTF (that is, preemptive SJF) and, for the sake of comparison, SJF (non-preemptive) and FCFS. The workload is specified a sequence of integers separated by whitespace, like this—

$$a_1 \quad b_1 \quad a_2 \quad b_2 \quad \cdots \quad a_n \quad b_n$$

where (a_k, b_k) are the arrival time and CPU burst length of the *k*-th process. You may assume that the processes are given in ascending order of arrival time (i.e., $a_k \leq a_{k+1}$).

Your program will output the execution schedule and average waiting time for each scheduling algorithm. Use A, B, C, ... to denote the processes in arrival order. Note that the first process can arrive at any time – it need not be time 0.

The following execution snapshot illustrates the required I/O format (user input shown in bold).

```
Enter process arrival times and burst lengths.
0 10 3 5 4 2 7 8 8 4

SRTF: A3 B1 C2 B4 E4 A7 D8
Average waiting time: 5.80

SJF: A10 C2 E4 B5 D8
Average waiting time: 7.40

FCFS: A10 B5 C2 D8 E4
Average waiting time: 9.00
```

In this example, process *A* enters the system at time 0 and has a burst length of 10 units, process *B* enters the system at time 3 and has a burst length of 5 units, and so on.

The execution schedule for each algorithm shows the order in which the processes are selected to run. The schedule for SRTF shown above indicates that process *A* runs for 3 units of time, then process *B* runs for 1 unit of time, then process *C* runs for 2 units of time, and so on.

Average waiting times are displayed with 2 digits of precision.

Implement a class *Process* to encapsulate the arrival time, waiting time, CPU time remaining, and any other relevant attributes of a process. Use PriorityQueue<Process> for your ready queue. You will need to pass a comparator[1] to the *PriorityQueue* constructor to specify how processes are to be ordered (by arrival time for FCFS, and by CPU time remaining for SRTF and SJF).

1 Specifically, implement the *Comparator<Process>* interface. This is simple and will be reviewed in class.

Implement separate helper methods for the three different simulations, so that the *main* method only needs to organize user input, invoke the helper methods, and report results. (Alternatively, you could implement an abstract *Scheduler* class and extend it with concrete subclasses *SRTF*, *SJF*, and *FCFS* responsible for managing the respective simulations. This would be a bit more involved, but the resulting code would be easier to extend to other scheduling algorithms.)

Each simulation consists of a loop that repeats once for each unit of time. If a process is arriving at the current time, it is put into the ready queue. When a process finishes executing, it is removed from the system and another is taken from the queue (if not empty). You will need some code in the body of the simulation loop to build the execution schedule and keep track of the total waiting time for all processes (in order to compute the average when the simulation ends). SRTF will require a little more work that the other two algorithms since a newly arriving process may preempt the currently running one.

As always, your code must at least compile to be eligible for partial credit. Obvious bugs or missing functionality must be documented. Be as specific as possible and for bugs give a simple example of a workload that causes the erroneous behavior. The documentation should give me a clear and accurate picture of what to expect when I test your code.

You can create your own workloads and perform the simulation by hand for testing purposes, but here are two more execution snapshots you can use for testing.

```
Enter process arrival times and burst lengths:
0 9 3 5 4 2

STRF: A3 B1 C2 B4 A6
Average waiting time: 3.00

SJF: A9 C2 B5
Average waiting time: 4.33

FCFS: A9 B5 C2
Average waiting time: 5.33
```

```
Enter process arrival times and burst lengths:
1 8 2 5 3 5 4 2

SRTF: A1 B2 D2 B3 C5 A7
Average waiting time: 5.00

SJF:  A8 D2 B5 C5
Average waiting time: 6.75

FCFS: A8 B5 C5 D2
Average waiting time: 8.25
```