CS419 Project 1: Implementing Scheduling Algorithms for a CPU Scheduler Simulator

You can team up with another student or work alone. Each team can have no more than two students.

When it is due:

Monday, November 9, at 11:59pm.

What to submit:

Please submit the completed **sjf.java** and **rr.java** files.
(There is no need to submit any other java files as you must not modify any of them.)

For a two-person team, you will just need to submit one copy (either member submits on Canvas). Please be sure to have both team members' names at the top of both files.

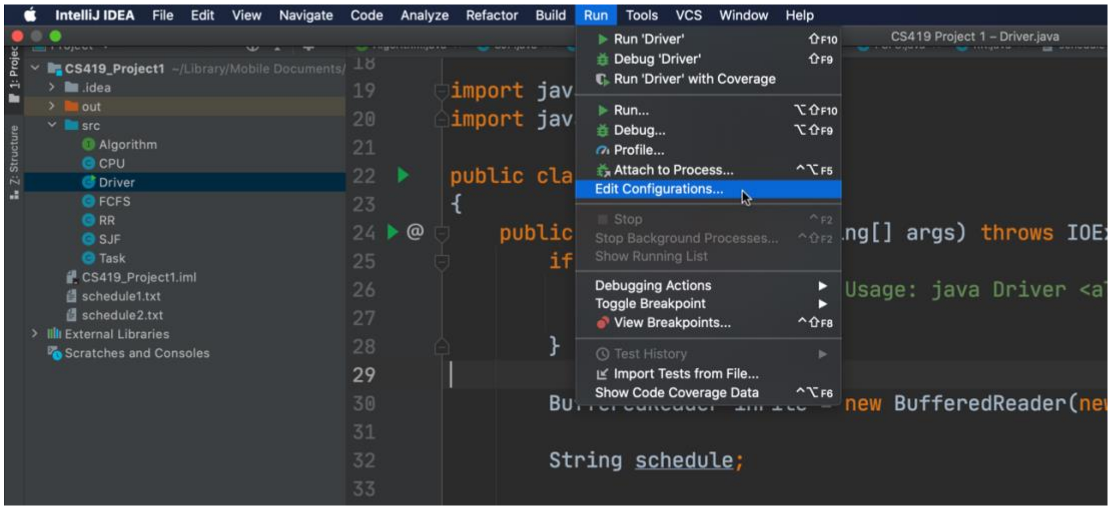
Objectives:

1. Reinforce the basic concepts of CPU scheduling.
2. Gain deeper understanding of two scheduling algorithms, Shortest job first and Round Robin, through implementation.
3. Compare the multiple scheduling algorithms via simulation

Instructions:

1. This project requires you to work on a Java program that simulates CPU scheduling using different scheduling algorithms. You will need to read and write multiple Java source files. We highly recommend that you use an IDE, such as IntelliJ IDEA (the free Community version is more than sufficient for this project) or Eclipse (also free to use), to work on this project.
2. You are provided with all the Java source files of the simulation program and you should not create any new java file. However, two of the Java files, **sjf.java** and **rr.java**, are incomplete and you will need to complete both of them, as follows:
    a. Implement the *non-preemptive version of shortest-job-first* scheduling algorithm in **sjf.java**
    b. Implement the *round-robin* scheduling algorithm in **rr.java**
        i. Use a time quantum of *5* for the first test case in schedule1.txt;
        ii. Use a time quantum of *10* for the second test case in schedule2.txt;
    c. Do not touch any other Java source file, and your implementation must work with the existing source files.

3. You are provided with two test cases, **schedule1.txt** and **schedule2.txt**, each containing a set of processes (first column), their arrival times (second column), and their CPU burst times (third column). Use both to test your implementation.
    a. Remember to change the time quanta for the round-robin scheduling algorithm when you change test cases.
4. Several simplifying assumptions have been made:
    a. Each process only has a single CPU burst and no I/O wait.
    b. There is no context switch delay.
    c. For the shortest-job-first algorithm, it knows the exact CPU burst time of every process (i.e., no need to do any approximation).
5. The provided **fcfs.java** contains the complete implementation of the first-come, first-served scheduling algorithm. We highly recommend that you read the source code of **fcfs.java** to get a better understanding of the elements of the implementation of a scheduling algorithm and how to work with the other Java classes.
6. We also embedded detailed comments in every source file to explain the code. Please read those comments and the source code to understand how the simulation works. You will need this understanding in order to implement the two scheduling algorithms, as your implementation and other classes will need information from each other in order to function. Again, a good idea will be to read the **fcfs.java** and use that implementation to guide your own.
7. The *main* method (in **Driver.java**) requires two arguments: the first one is the scheduling algorithm (select from: *fcfs*, *sjf*, *rr*), and the second one is the name of the text file containing the test case (Note: you may need to specify the full file name including the path, depending on where you place those test case files).
   If you use IntelliJ, you can specify those two arguments in the "Program arguments" box in the "Run/Debug Configurations" window, which is accessed by clicking the "Edit Configurations…" option under the "Run" tab, as shown below; Similar configuration is available in Eclipse.

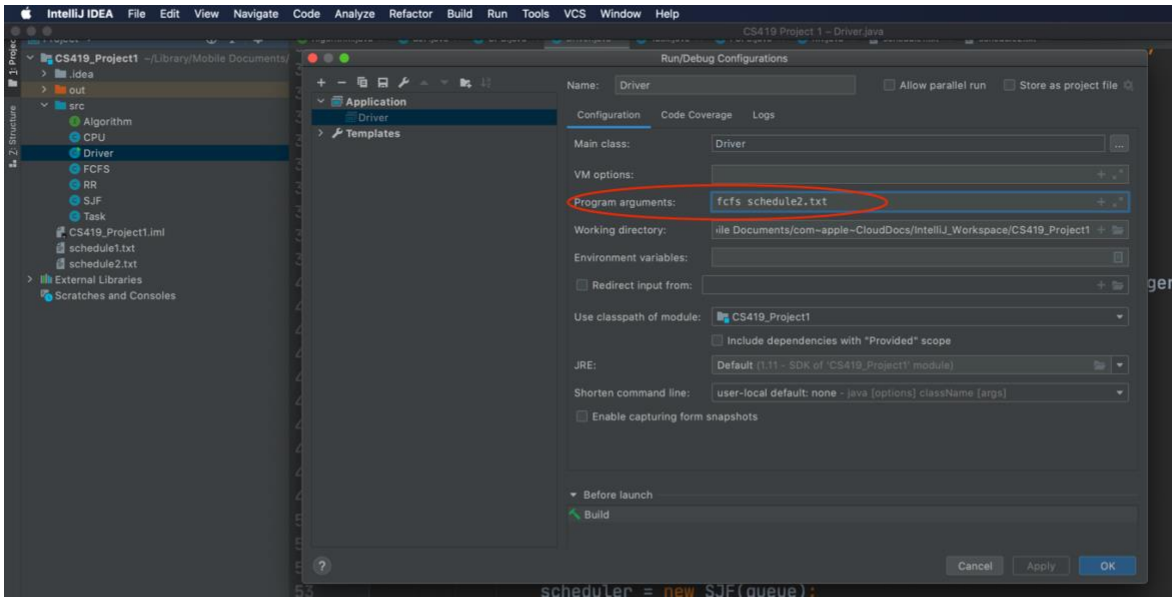


```
import jav
import jav

public cla
{
    public                       ng[] args) throws IOE
        if
                                 Usage: java Driver <a
        }

        Bu                       new BufferedReader(ne

        String schedule;
```



Name: Driver
Allow parallel run
Store as project file
Main class: Driver
VM options:
Program arguments: `fcfs schedule2.txt`
Working directory: ıile Documents/com~apple~CloudDocs/IntelliJ_Workspace/CS419_Project1
Environment variables:
Redirect input from:
Use classpath of module: CS419_Project1
Include dependencies with "Provided" scope
JRE: Default (1.11 - SDK of 'CS419_Project1' module)
Shorten command line: user-local default: none - java [options] className [args]
Enable capturing form snapshots

```
scheduler = new SJF(queue);
```

8. You will receive 70% of the points if you correctly implemented one of the two scheduling algorithms (either one); you will receive 100% of the points if you correctly implemented both scheduling algorithms.

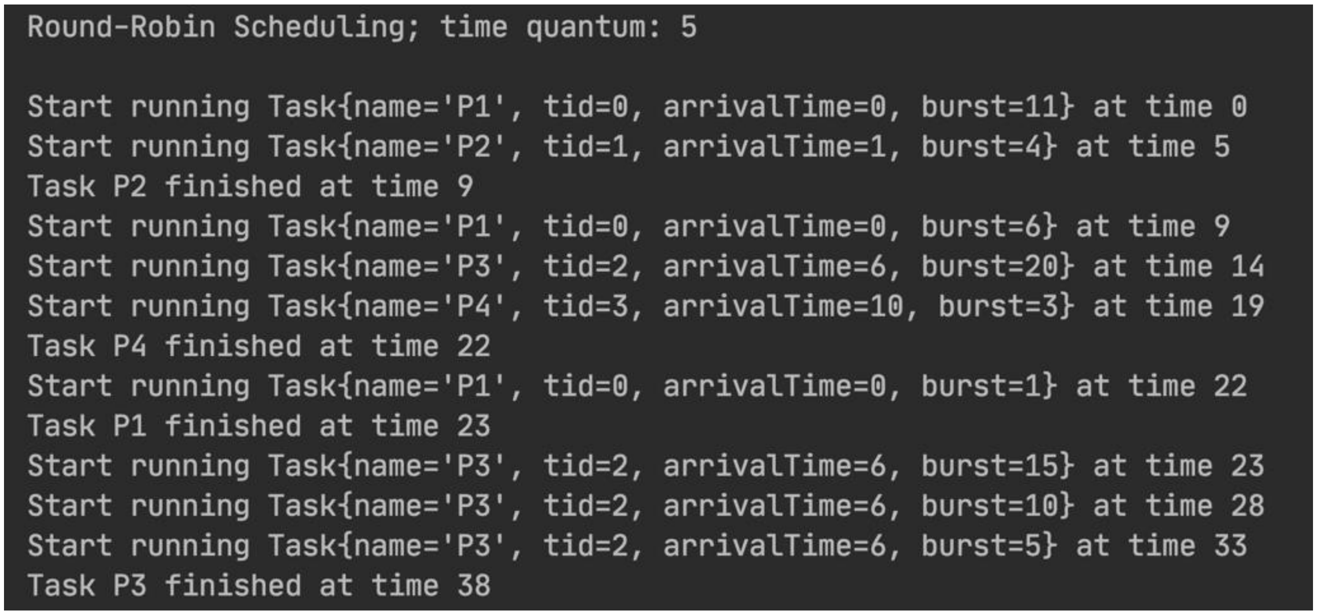
Sample Output from running test case #1 (schedule1.txt):

<u>Shortest job first:</u>

```
Shortest Job First (non-preemptive) Scheduling

Start running Task{name='P1', tid=0, arrivalTime=0, burst=11} at time 0
Task P1 finished at time 11
Start running Task{name='P4', tid=3, arrivalTime=10, burst=3} at time 11
Task P4 finished at time 14
Start running Task{name='P2', tid=1, arrivalTime=1, burst=4} at time 14
Task P2 finished at time 18
Start running Task{name='P3', tid=2, arrivalTime=6, burst=20} at time 18
Task P3 finished at time 38
```

Round robin: (make sure the time quantum is set to *5* for this test case)

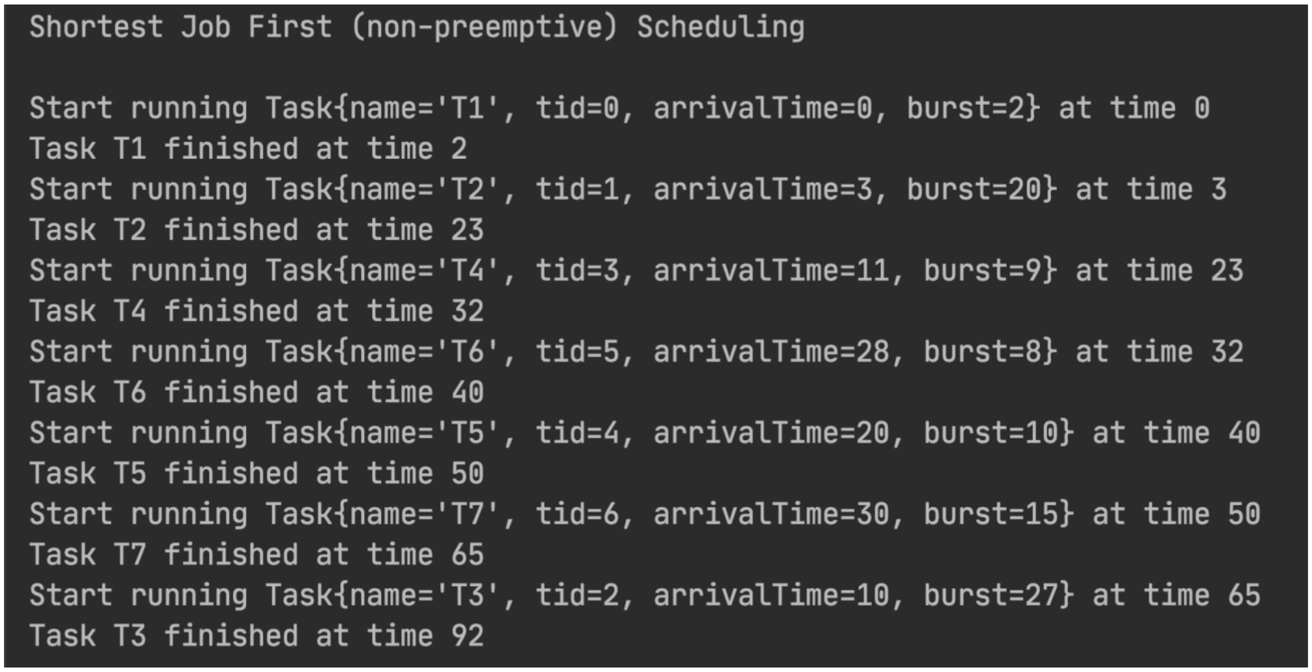


```
Round-Robin Scheduling; time quantum: 5

Start running Task{name='P1', tid=0, arrivalTime=0, burst=11} at time 0
Start running Task{name='P2', tid=1, arrivalTime=1, burst=4} at time 5
Task P2 finished at time 9
Start running Task{name='P1', tid=0, arrivalTime=0, burst=6} at time 9
Start running Task{name='P3', tid=2, arrivalTime=6, burst=20} at time 14
Start running Task{name='P4', tid=3, arrivalTime=10, burst=3} at time 19
Task P4 finished at time 22
Start running Task{name='P1', tid=0, arrivalTime=0, burst=1} at time 22
Task P1 finished at time 23
Start running Task{name='P3', tid=2, arrivalTime=6, burst=15} at time 23
Start running Task{name='P3', tid=2, arrivalTime=6, burst=10} at time 28
Start running Task{name='P3', tid=2, arrivalTime=6, burst=5} at time 33
Task P3 finished at time 38
```

Sample Output from running test case #2 (schedule2.txt):

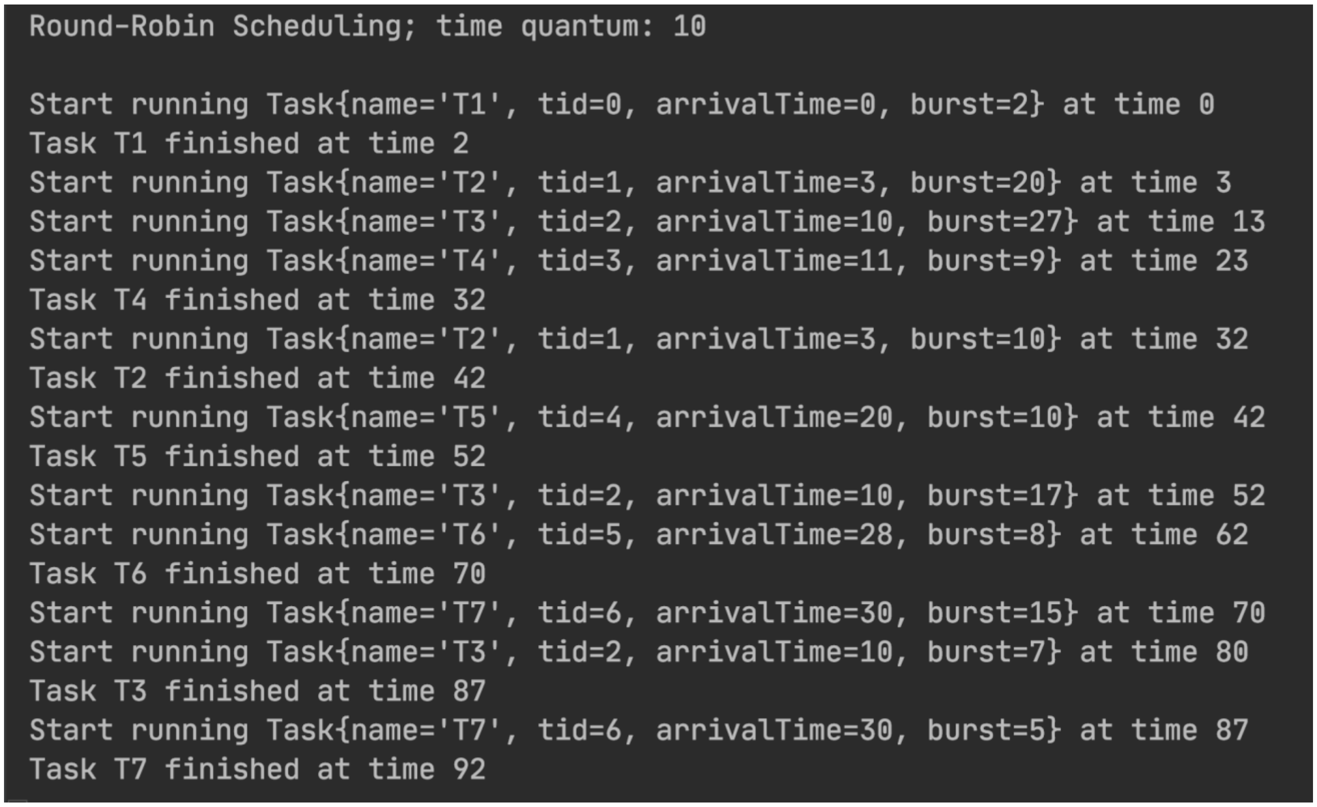
Shortest job first:



```
Shortest Job First (non-preemptive) Scheduling

Start running Task{name='T1', tid=0, arrivalTime=0, burst=2} at time 0
Task T1 finished at time 2
Start running Task{name='T2', tid=1, arrivalTime=3, burst=20} at time 3
Task T2 finished at time 23
Start running Task{name='T4', tid=3, arrivalTime=11, burst=9} at time 23
Task T4 finished at time 32
Start running Task{name='T6', tid=5, arrivalTime=28, burst=8} at time 32
Task T6 finished at time 40
Start running Task{name='T5', tid=4, arrivalTime=20, burst=10} at time 40
Task T5 finished at time 50
Start running Task{name='T7', tid=6, arrivalTime=30, burst=15} at time 50
Task T7 finished at time 65
Start running Task{name='T3', tid=2, arrivalTime=10, burst=27} at time 65
Task T3 finished at time 92
```

Round robin: (make sure the time quantum is set to *10* for this test case)



```
Round-Robin Scheduling; time quantum: 10

Start running Task{name='T1', tid=0, arrivalTime=0, burst=2} at time 0
Task T1 finished at time 2
Start running Task{name='T2', tid=1, arrivalTime=3, burst=20} at time 3
Start running Task{name='T3', tid=2, arrivalTime=10, burst=27} at time 13
Start running Task{name='T4', tid=3, arrivalTime=11, burst=9} at time 23
Task T4 finished at time 32
Start running Task{name='T2', tid=1, arrivalTime=3, burst=10} at time 32
Task T2 finished at time 42
Start running Task{name='T5', tid=4, arrivalTime=20, burst=10} at time 42
Task T5 finished at time 52
Start running Task{name='T3', tid=2, arrivalTime=10, burst=17} at time 52
Start running Task{name='T6', tid=5, arrivalTime=28, burst=8} at time 62
Task T6 finished at time 70
Start running Task{name='T7', tid=6, arrivalTime=30, burst=15} at time 70
Start running Task{name='T3', tid=2, arrivalTime=10, burst=7} at time 80
Task T3 finished at time 87
Start running Task{name='T7', tid=6, arrivalTime=30, burst=5} at time 87
Task T7 finished at time 92
```