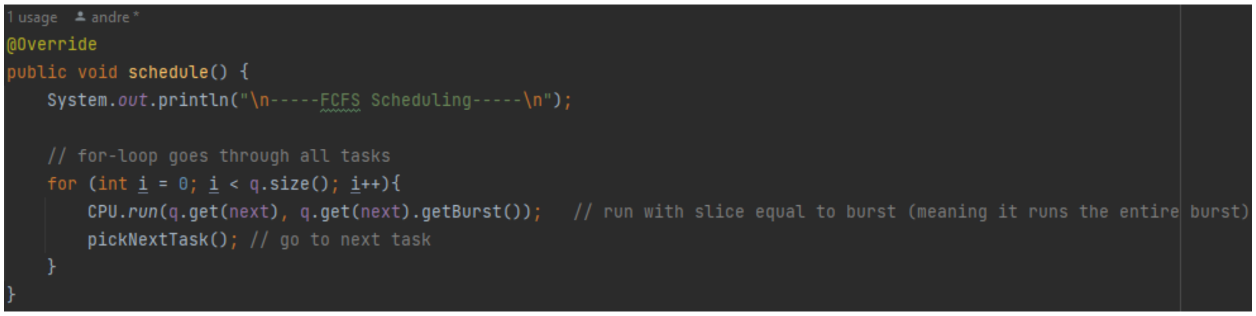
Andre Hei Wang Law - 4017 5600

1) Comparison of the Scheduling Algorithms:

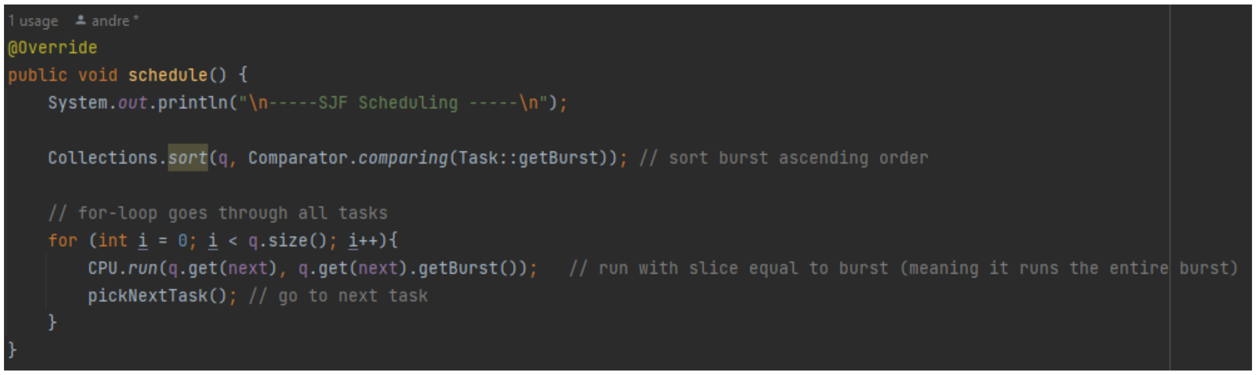
First-Come, First Served (FCFS):



```
1 usage  andre *
@Override
public void schedule() {
    System.out.println("\n-----FCFS Scheduling-----\n");

    // for-loop goes through all tasks
    for (int i = 0; i < q.size(); i++){
        CPU.run(q.get(next), q.get(next).getBurst());   // run with slice equal to burst (meaning it runs the entire burst)
        pickNextTask(); // go to next task
    }
}
```

Shorted-Job-First (SJF):

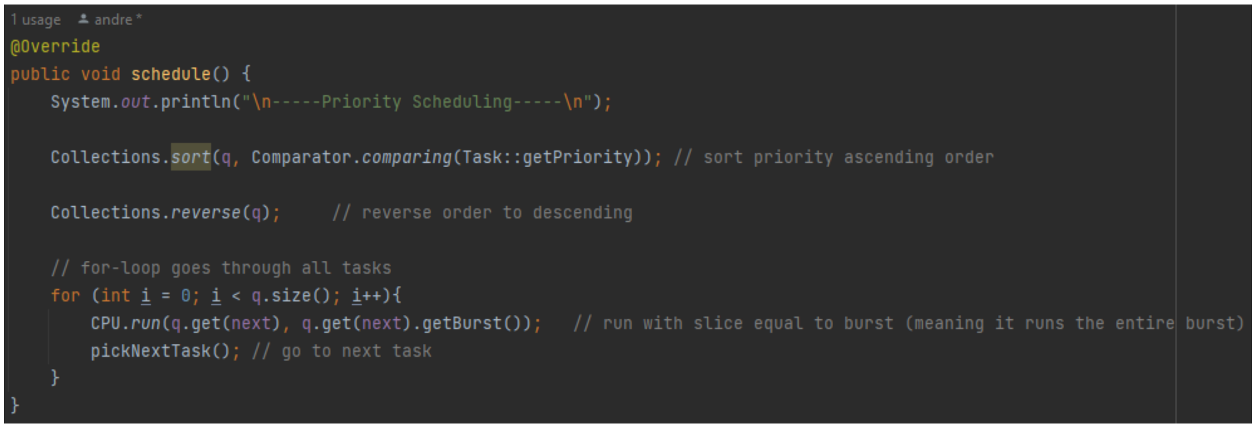


```
1 usage  andre *
@Override
public void schedule() {
    System.out.println("\n-----SJF Scheduling -----\n");

    Collections.sort(q, Comparator.comparing(Task::getBurst)); // sort burst ascending order

    // for-loop goes through all tasks
    for (int i = 0; i < q.size(); i++){
        CPU.run(q.get(next), q.get(next).getBurst());   // run with slice equal to burst (meaning it runs the entire burst)
        pickNextTask(); // go to next task
    }
}
```

Priority Scheduling (PS):



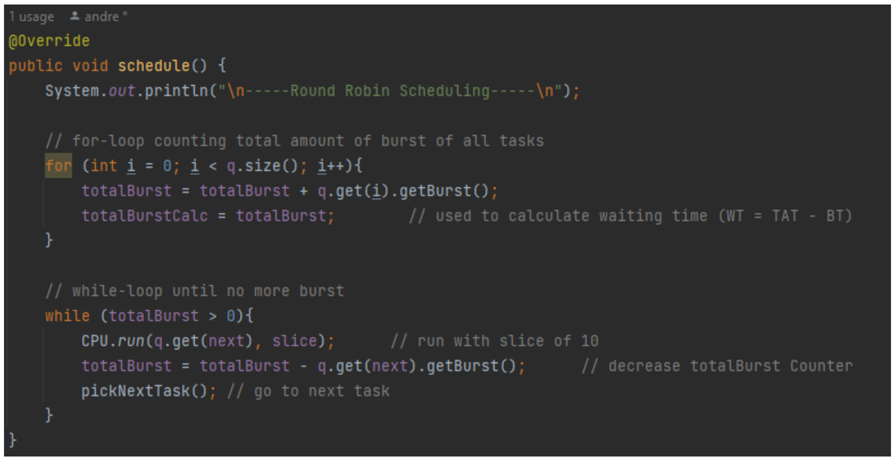
```
1 usage  andre *
@Override
public void schedule() {
    System.out.println("\n-----Priority Scheduling-----\n");

    Collections.sort(q, Comparator.comparing(Task::getPriority)); // sort priority ascending order

    Collections.reverse(q);     // reverse order to descending

    // for-loop goes through all tasks
    for (int i = 0; i < q.size(); i++){
        CPU.run(q.get(next), q.get(next).getBurst());   // run with slice equal to burst (meaning it runs the entire burst)
        pickNextTask(); // go to next task
    }
}
```

Found-Robin (RR):

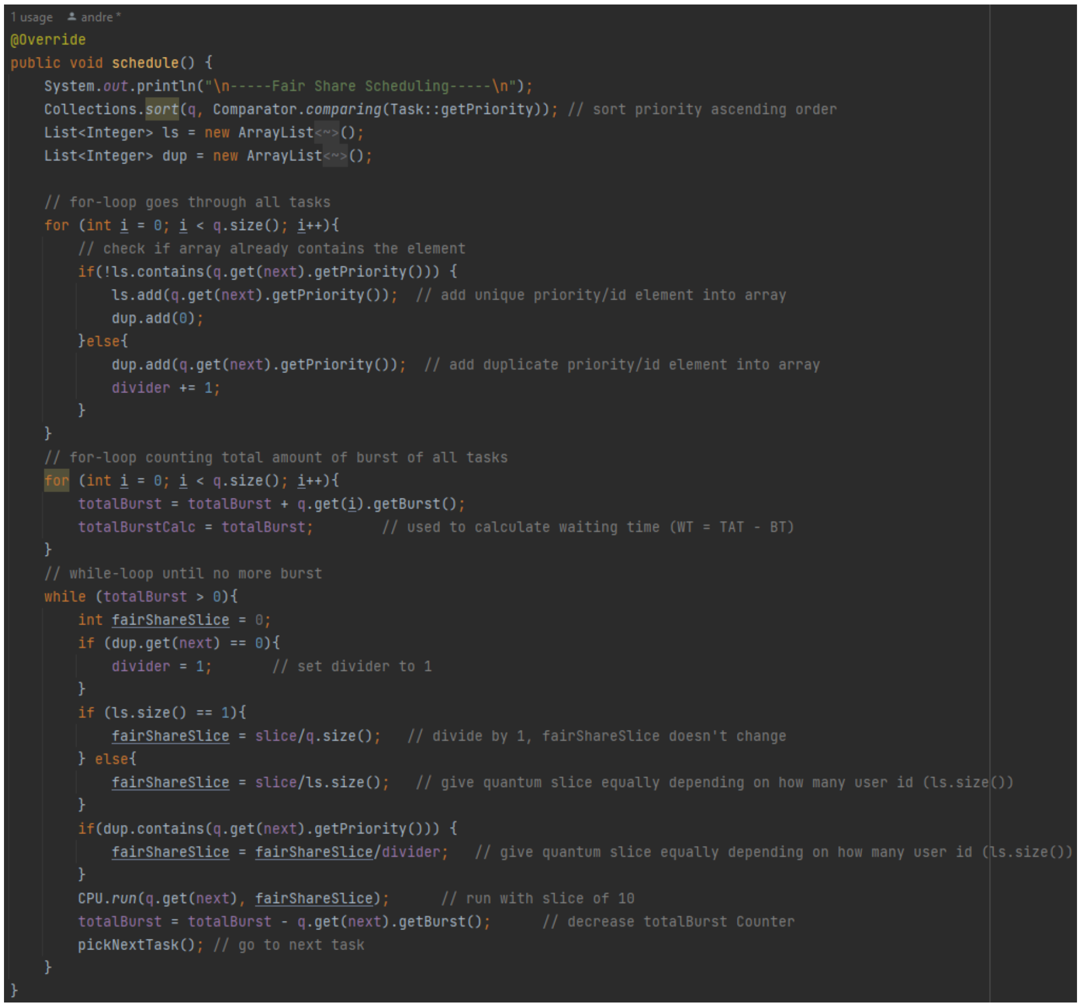


```
1 usage  andre *
@Override
public void schedule() {
    System.out.println("\n-----Round Robin Scheduling-----\n");

    // for-loop counting total amount of burst of all tasks
    for (int i = 0; i < q.size(); i++){
        totalBurst = totalBurst + q.get(i).getBurst();
        totalBurstCalc = totalBurst;        // used to calculate waiting time (WT = TAT - BT)
    }

    // while-loop until no more burst
    while (totalBurst > 0){
        CPU.run(q.get(next), slice);      // run with slice of 10
        totalBurst = totalBurst - q.get(next).getBurst();      // decrease totalBurst Counter
        pickNextTask(); // go to next task
    }
}
```

Fair Share (FS):



```
1 usage  andre *
@Override
public void schedule() {
    System.out.println("\n-----Fair Share Scheduling-----\n");
    Collections.sort(q, Comparator.comparing(Task::getPriority)); // sort priority ascending order
    List<Integer> ls = new ArrayList<~>();
    List<Integer> dup = new ArrayList<~>();

    // for-loop goes through all tasks
    for (int i = 0; i < q.size(); i++){
        // check if array already contains the element
        if(!ls.contains(q.get(next).getPriority())) {
            ls.add(q.get(next).getPriority());  // add unique priority/id element into array
            dup.add(0);
        }else{
            dup.add(q.get(next).getPriority());  // add duplicate priority/id element into array
            divider += 1;
        }
    }
    // for-loop counting total amount of burst of all tasks
    for (int i = 0; i < q.size(); i++){
        totalBurst = totalBurst + q.get(i).getBurst();
        totalBurstCalc = totalBurst;        // used to calculate waiting time (WT = TAT - BT)
    }
    // while-loop until no more burst
    while (totalBurst > 0){
        int fairShareSlice = 0;
        if (dup.get(next) == 0){
            divider = 1;       // set divider to 1
        }
        if (ls.size() == 1){
            fairShareSlice = slice/q.size();   // divide by 1, fairShareSlice doesn't change
        } else{
            fairShareSlice = slice/ls.size();   // give quantum slice equally depending on how many user id (ls.size())
        }
        if(dup.contains(q.get(next).getPriority())) {
            fairShareSlice = fairShareSlice/divider;   // give quantum slice equally depending on how many user id (ls.size())
        }
        CPU.run(q.get(next), fairShareSlice);      // run with slice of 10
        totalBurst = totalBurst - q.get(next).getBurst();      // decrease totalBurst Counter
        pickNextTask(); // go to next task
    }
}
```

2) Advantages and Drawbacks:

First-Come, First Served (FCFS):

Advantages: Simple to implement; no starvation; fairness.

Drawbacks: Poor response time; convoy effect; inefficient use of resources.

Shorted-Job-First (SJF):

Advantages: Reduces average waiting time and turnaround time; efficient use of resources.

Drawbacks: Prediction of CPU burst time is difficult; may lead to starvation of long processes.

Priority Scheduling (PS):

Advantages: Allows priority to be given to important processes; reduces response time of important processes.

Drawbacks: Low priority processes may face starvation; may result in low priority processes not completing.

Found-Robin (RR):

Advantages: Time slice given to each process, ensuring each process gets a fair share of CPU time; efficient use of resources.

Drawbacks: Large time quantum may lead to poor response time; small time quantum may lead to high overhead.

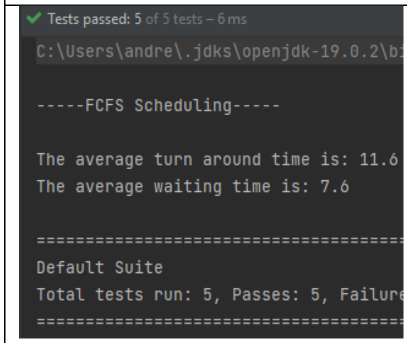
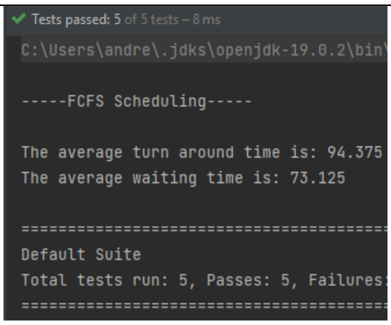
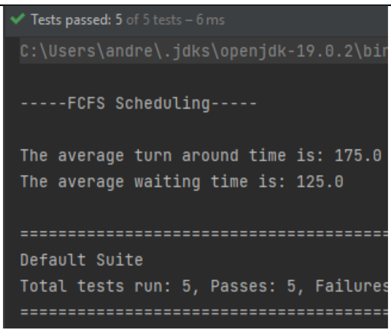
Fair Share (FS):

Advantages: Ensures fair distribution of CPU time among all processes; prevents any single process from dominating the system.

Drawbacks: Overhead may be high; not suitable for systems with time-sensitive tasks.

3) Average Waiting and Turnaround Time:

First-Come, First Served (FCFS):

book.txt	schedule.txt	rr-schedule.txt
		

```
Tests passed: 5 of 5 tests – 6 ms
C:\Users\andre\.jdks\openjdk-19.0.2\bi

-----FCFS Scheduling-----

The average turn around time is: 11.6
The average waiting time is: 7.6

======================================
Default Suite
Total tests run: 5, Passes: 5, Failure
======================================
```

```
Tests passed: 5 of 5 tests – 8 ms
C:\Users\andre\.jdks\openjdk-19.0.2\bin\

-----FCFS Scheduling-----

The average turn around time is: 94.375
The average waiting time is: 73.125

========================================
Default Suite
Total tests run: 5, Passes: 5, Failures:
========================================
```

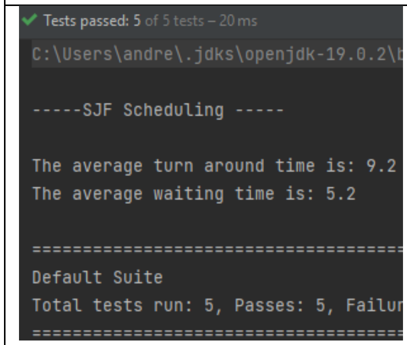
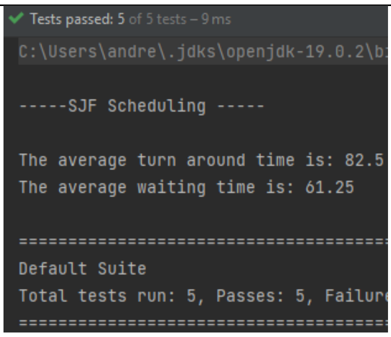
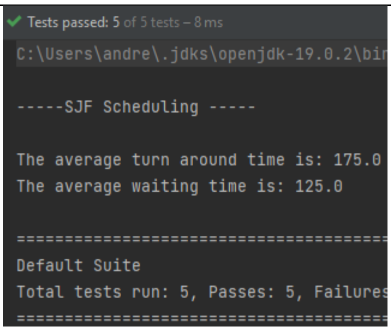
```
Tests passed: 5 of 5 tests – 6 ms
C:\Users\andre\.jdks\openjdk-19.0.2\bin

-----FCFS Scheduling-----

The average turn around time is: 175.0
The average waiting time is: 125.0

=======================================
Default Suite
Total tests run: 5, Passes: 5, Failures
=======================================
```

Shorted-Job-First (SJF):

book.txt	schedule.txt	rr-schedule.txt
		

```
Tests passed: 5 of 5 tests – 20 ms
C:\Users\andre\.jdks\openjdk-19.0.2\b

-----SJF Scheduling -----

The average turn around time is: 9.2
The average waiting time is: 5.2

=====================================
Default Suite
Total tests run: 5, Passes: 5, Failur
=====================================
```

```
Tests passed: 5 of 5 tests – 9 ms
C:\Users\andre\.jdks\openjdk-19.0.2\bi

-----SJF Scheduling -----

The average turn around time is: 82.5
The average waiting time is: 61.25

======================================
Default Suite
Total tests run: 5, Passes: 5, Failure
======================================
```

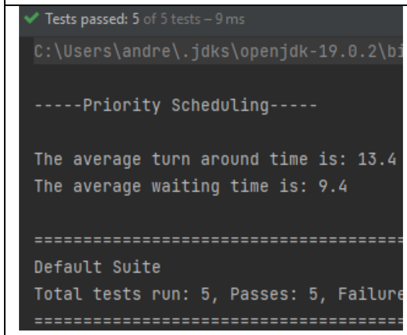
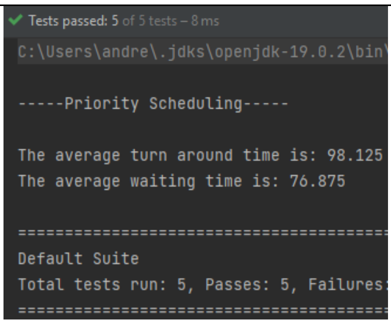
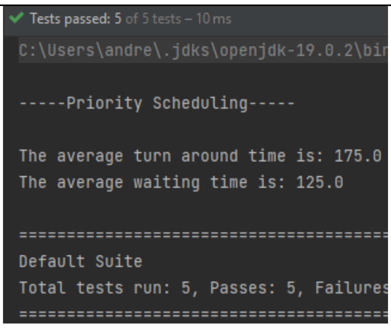
```
Tests passed: 5 of 5 tests – 8 ms
C:\Users\andre\.jdks\openjdk-19.0.2\bin

-----SJF Scheduling -----

The average turn around time is: 175.0
The average waiting time is: 125.0

=======================================
Default Suite
Total tests run: 5, Passes: 5, Failures
=======================================
```

Priority Scheduling (PS):

book.txt	schedule.txt	rr-schedule.txt
		

```
Tests passed: 5 of 5 tests – 9 ms
C:\Users\andre\.jdks\openjdk-19.0.2\bi

-----Priority Scheduling-----

The average turn around time is: 13.4
The average waiting time is: 9.4

======================================
Default Suite
Total tests run: 5, Passes: 5, Failure
======================================
```

```
Tests passed: 5 of 5 tests – 8 ms
C:\Users\andre\.jdks\openjdk-19.0.2\bin\

-----Priority Scheduling-----

The average turn around time is: 98.125
The average waiting time is: 76.875

========================================
Default Suite
Total tests run: 5, Passes: 5, Failures:
========================================
```

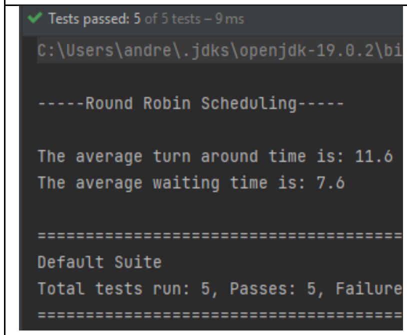
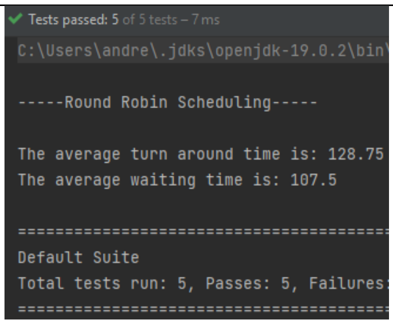
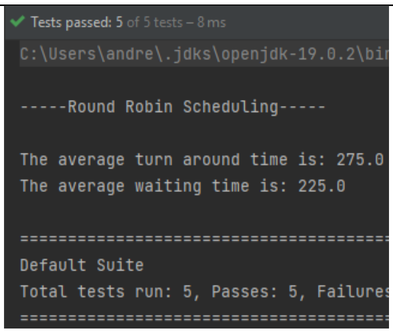
```
Tests passed: 5 of 5 tests – 10 ms
C:\Users\andre\.jdks\openjdk-19.0.2\bin

-----Priority Scheduling-----

The average turn around time is: 175.0
The average waiting time is: 125.0

=======================================
Default Suite
Total tests run: 5, Passes: 5, Failures
=======================================
```

Round-Robin (RR):

book.txt	schedule.txt	rr-schedule.txt
		

```
Tests passed: 5 of 5 tests – 9 ms
C:\Users\andre\.jdks\openjdk-19.0.2\bi

-----Round Robin Scheduling-----

The average turn around time is: 11.6
The average waiting time is: 7.6

======================================
Default Suite
Total tests run: 5, Passes: 5, Failure
======================================
```

```
Tests passed: 5 of 5 tests – 7 ms
C:\Users\andre\.jdks\openjdk-19.0.2\bin\

-----Round Robin Scheduling-----

The average turn around time is: 128.75
The average waiting time is: 107.5

========================================
Default Suite
Total tests run: 5, Passes: 5, Failures:
========================================
```

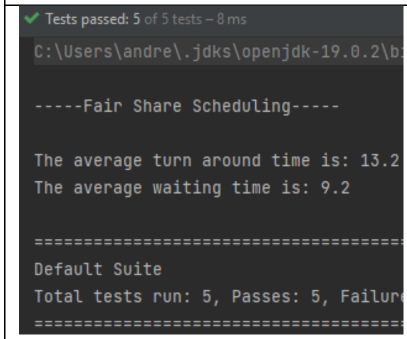
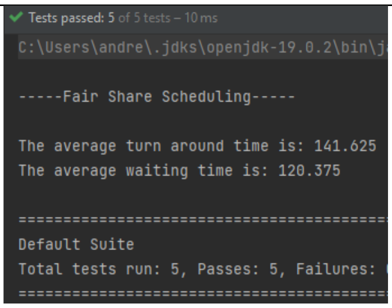
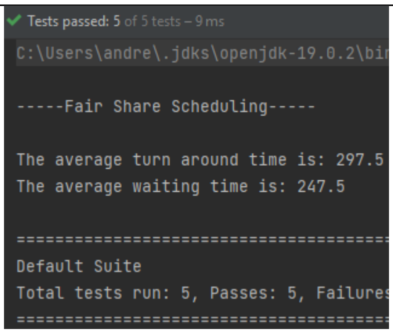
```
Tests passed: 5 of 5 tests – 8 ms
C:\Users\andre\.jdks\openjdk-19.0.2\bin

-----Round Robin Scheduling-----

The average turn around time is: 275.0
The average waiting time is: 225.0

=======================================
Default Suite
Total tests run: 5, Passes: 5, Failures
=======================================
```

Fair Share (FS):

book.txt	schedule.txt	rr-schedule.txt
		

```
Tests passed: 5 of 5 tests – 8 ms
C:\Users\andre\.jdks\openjdk-19.0.2\bi

-----Fair Share Scheduling-----

The average turn around time is: 13.2
The average waiting time is: 9.2

======================================
Default Suite
Total tests run: 5, Passes: 5, Failure
======================================
```

```
Tests passed: 5 of 5 tests – 10 ms
C:\Users\andre\.jdks\openjdk-19.0.2\bin\j

-----Fair Share Scheduling-----

The average turn around time is: 141.625
The average waiting time is: 120.375

=========================================
Default Suite
Total tests run: 5, Passes: 5, Failures:
=========================================
```

```
Tests passed: 5 of 5 tests – 9 ms
C:\Users\andre\.jdks\openjdk-19.0.2\bin

-----Fair Share Scheduling-----

The average turn around time is: 297.5
The average waiting time is: 247.5

=======================================
Default Suite
Total tests run: 5, Passes: 5, Failures
=======================================
```