CPU Scheduler:

The CPU scheduler selects from among the processes in the ready queue, and allocates a CPU core to one of them. The goal of the scheduler is to maximize CPU Utilization, CPU Throughput and reduce turnaround time, response time, and waiting time. The CPU can do this through various algorithms to select processes for running. In this project I have implemented a round robin scheduler.

Round Robin Scheduling:

In Round Robin Scheduling, each process gets a small unit of CPU time (time quantum q), usually 10-100 milliseconds. After this time has elapsed, the process is preempted and added to the end of the ready queue and the scheduler runs another process for the given time quantum.

My Program:

Included in my submission is the program jar file named “program.jar” and a CSV file with example processes modeled on the one given in the project document. To run the program you would need to open a command prompt in the folder with the jar and run with 3 or more arguments. The first argument should be the path to the CSV file or any other CSV file. The second argument is the time of context switches and the third argument is the time quantum. Any arguments past the 3rd one is another quantum argument. The program will then run the simulations with the given time quantums and open a visualization of the results of each to better compare the two.

Example: Running with 10usec of context switching and 1ms of time quantum

```
Average Process Turnaround Time: 107038.3

C:\New folder\CPU Scheduling Project>java -jar program.jar "C:\New folder\CPU Scheduling Project\processes.csv" 10 10000
```

“java -jar program.jar "C:\New folder\CPU Scheduling Project\processes.csv" 10 10000”

The output of this would be a time-by-time of what the CPU is doing and at the end, a summary is printed.

```
[CPU] Context Switching
[CPU] Running process with ID: 9
[CPU] Context Switching
[CPU] Running process with ID: 8
[CPU] Context Switching
[CPU] Running process with ID: 1
[CPU] Context Switching
[CPU] Running process with ID: 2
[CPU] Context Switching
[CPU] Running process with ID: 9
[CPU] Context Switching
[CPU] Running process with ID: 8
[CPU] Context Switching
[CPU] Running process with ID: 1
[CPU] Context Switching
[CPU] Running process with ID: 2
[CPU] Context Switching
[CPU] Running process with ID: 9
[CPU] Process completed execution at time: 625473 PID: 9
[CPU] Context Switching
[CPU] Running process with ID: 8
[CPU] Context Switching
[CPU] Running process with ID: 1
[CPU] Context Switching
[CPU] Running process with ID: 2
[CPU] Context Switching
```

```
[Completed] PID: 8
| Arrival Time: 222
| Burst Time: 123458
| Response Time: 63523
| Wait Time: 595351
| Completion Time: 719031
| Turnaround Time: 718809

[Completed] PID: 1
| Arrival Time: 0
| Burst Time: 302400
| Response Time: 0
| Wait Time: 759381
| Completion Time: 1061781
| Turnaround Time: 1061781

[Completed] PID: 2
| Arrival Time: 1
| Burst Time: 305005
| Response Time: 13473
| Wait Time: 761790
| Completion Time: 1066796
| Turnaround Time: 1066795

Total Time Taken: 1066796
Total Context Switches: 111
Total Idle Time: 1110
CPU Utilization: 99.89595011604843%

Average Process Response Time: 1347.3
Average Process Wait Time: 76179.0
Average Process Turnaround Time: 106679.5
```

The total time, context switched, idle time and utilization is calculated and printed. There are also averages for the process response time, wait time and turnaround time.

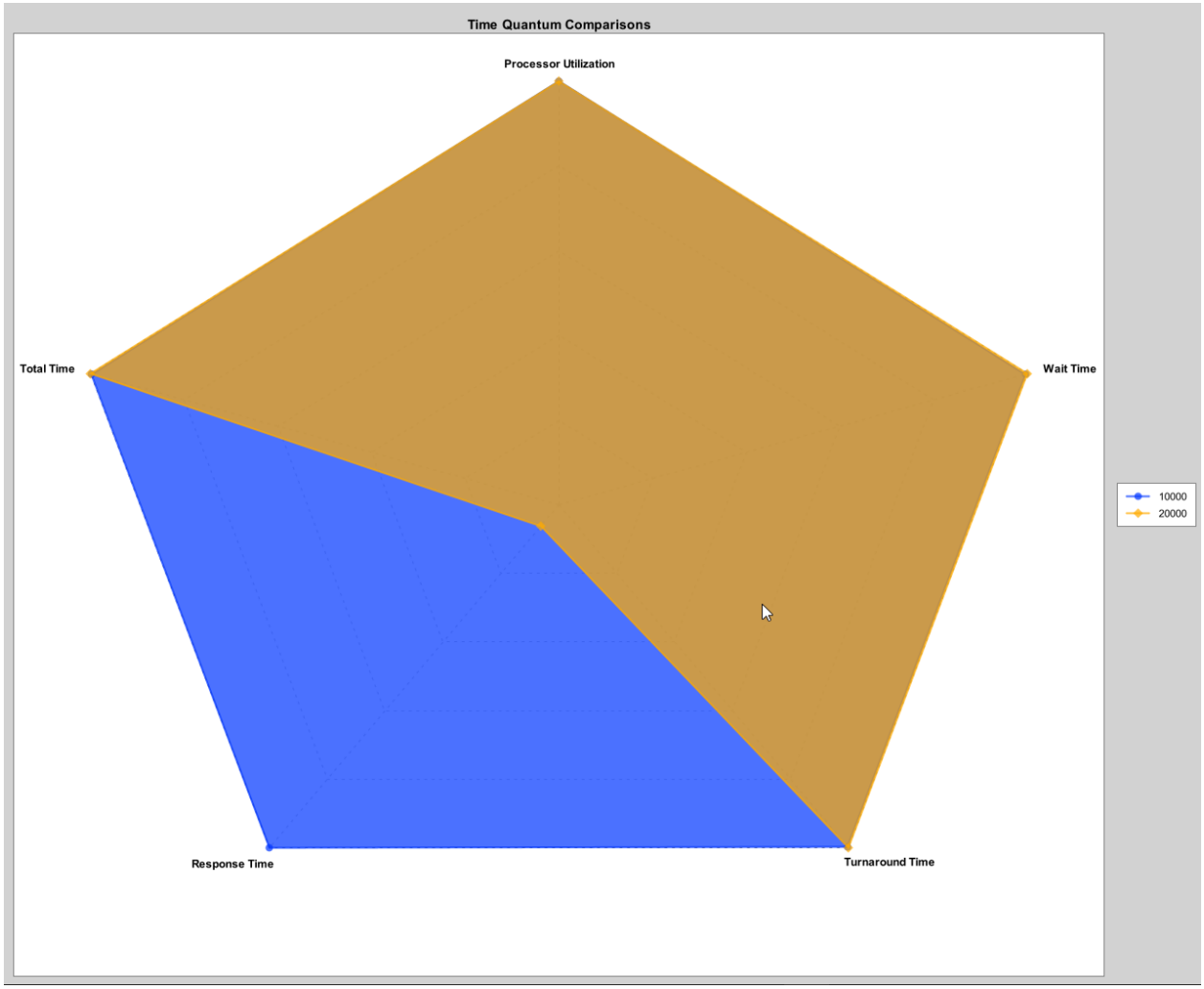
Example running multiple time quantums:

```
Average Process Turnaround Time: 106679.5

C:\New folder\CPU Scheduling Project>java -jar program.jar "C:\New folder\CPU Scheduling Project\processes.csv" 10 10000 20000
```

“java -jar program.jar "C:\New folder\CPU Scheduling Project\processes.csv" 10 10000 20000”

This input would be the same as running single time quantums for both 10000 and 20000, except when done this way a visualization is opened.

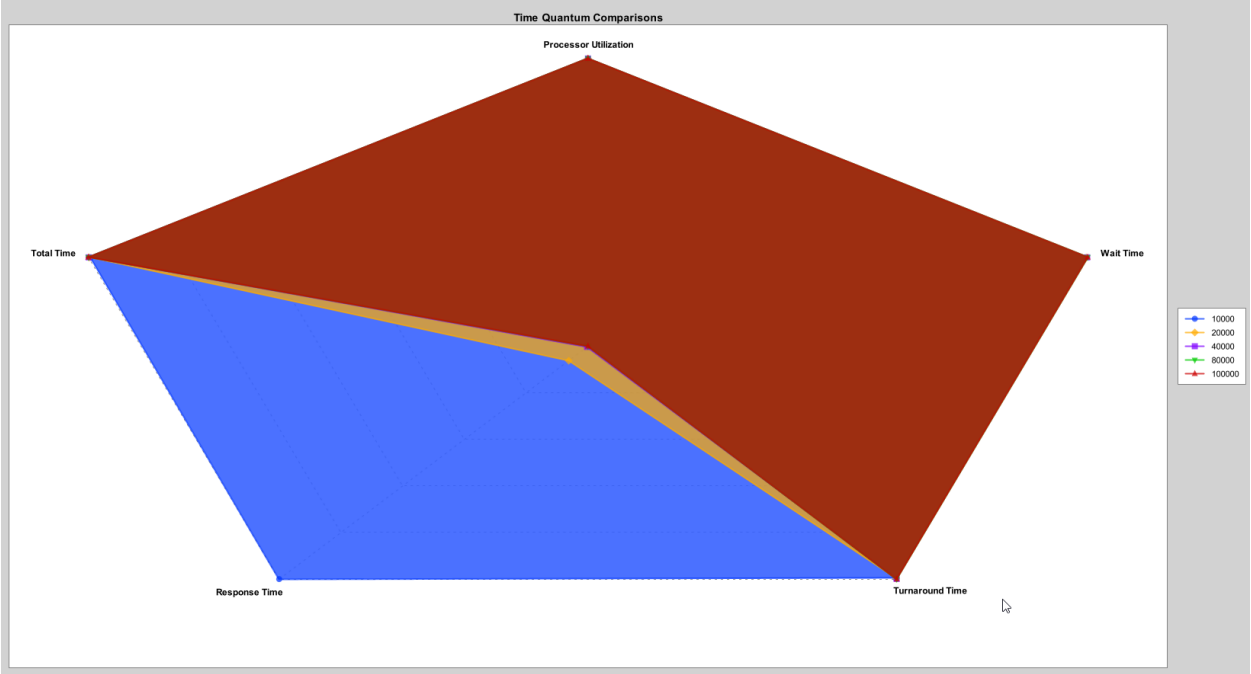


The format for this chart is a radar chart and due to the visualization method the higher values (bigger shape) is the more optimal time quantum. The output of each is also in the command line/console.

Comparison between 5 time quantums:

Command Used: “java -jar program.jar "C:\New folder\CPU Scheduling Project\processes.csv"

10 10000 20000 40000 80000 100000”



Outputs:

100,000:

```
Total Time Taken: 1065846
Total Context Switches: 16
Total Idle Time: 160
CPU Utilization: 99.9849884504891%

Average Process Response Time: 10347.3
Average Process Wait Time: 76084.0
Average Process Turnaround Time: 106584.5
```

80,000:

```
Total Time Taken: 1065856
Total Context Switches: 17
Total Idle Time: 170
CPU Utilization: 99.9840503782875%

Average Process Response Time: 8347.3
Average Process Wait Time: 76085.0
Average Process Turnaround Time: 106585.5
```

40,000

```
Total Time Taken: 1065996
Total Context Switches: 31
Total Idle Time: 310
CPU Utilization: 99.97091921545672%

Average Process Response Time: 4347.3
Average Process Wait Time: 76099.0
Average Process Turnaround Time: 106599.5
```

20,000

```
Total Time Taken: 1066266
Total Context Switches: 58
Total Idle Time: 580
CPU Utilization: 99.94560456771575%

Average Process Response Time: 2347.3
Average Process Wait Time: 76126.0
Average Process Turnaround Time: 106626.5
```

10,000

```
Total Time Taken: 1066796
Total Context Switches: 111
Total Idle Time: 1110
CPU Utilization: 99.89595011604843%

Average Process Response Time: 1347.3
Average Process Wait Time: 76179.0
Average Process Turnaround Time: 106679.5
```

The differences between these values for this dataset is very similar and there are trade offs as you change values. 10,000 has the greatest response time but the overall speed and time taken is fastest on the 100,000 time quantum.

These values and chart changes for each changed datapoint, whether in the csv file or in the command.

This is a chart of the same dataset and time quantums but the context switch time is significantly changed to 1000