King Saud University
College of Computer and Information Sciences
Computer Science Department

227 Project
Spring 2021

Round Robin Simulation Project

ASSUME QUANTUM = 2

439101298	مهند الرشيد	12321
438104296	حمد محمد الربيعه	35606
439102162	عبدالملك العرجاني	12321
439101644	عبدالعزيز الجمهور	12321

Leader Email: 439101298@student.ksu.edu.sa

Problem

CPU scheduling is a difficult task, many algorithms have been presented to accommodate different needs when dealing with scheduling. One such algorithm is Round Robin (abbreviated RR), which assigns what's called a quantum time **t** for each process and cycles through processes in a queue. In this project we intend to simulate this algorithm and measure its performance

Phase 1

In phase 1 we give general statistics about the given CSV file. We will investigate these metrics

- Total number of processes.
- Maximum burst time
- Minimum burst time
- How many processes have more than average burst time?
- How many processes have less than average burst time?

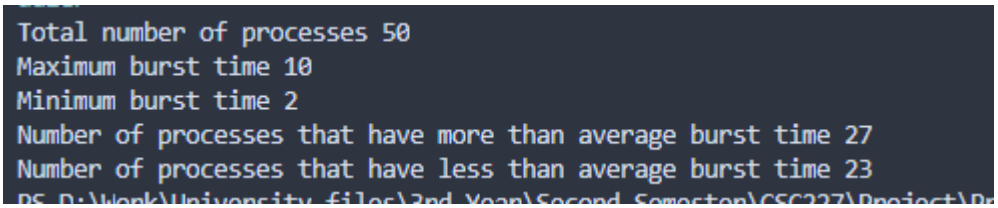
Data Structures

In order for us to measure the CSV file we need to transform it to a format we can easily manipulate. So we parse the CSV and save it into a **Multidimensional Array** (int[][]).

```
static int[][] parseCSV(String filename){
    try{
        int numLines = getLineCount(filename);
        int maxNumLines = numLines; // Keep copy of numLines
        int[][] csv = new int[numLines][3];
        FileReader file = new FileReader(filename);
        BufferedReader buffer = new BufferedReader(file);
        String line = "";
        String[] lineByLine = new String[3];
        for(line=buffer.readLine(); numLines!=0 && line!=null; numLines--, line=buffer.readLine()){
            lineByLine = line.split(",");
            for(int i=0; i<lineByLine.length; i++)
                csv[maxNumLines-numLines][lineByLine.length-i-1] = Integer.parseInt(lineByLine[i]);

        }
        buffer.close();
        return csv;
    } catch(IOException e){
        e.printStackTrace();
        return null;
    }
}
```

Results:

For the file given to us in LMS our program concludes the following statistics:



Phase 2

In Phase 2 we intend to simulate the Round Robin algorithm and print the following statistics about it

- Time the first process arrives.
- PID of the second process executed.
- Time the last process executed.
- PID of the last process executed.
- Finish time of every process.
- Total time for execution of all processes.
- Average turnaround time.
- Average waiting time.

Data Structures

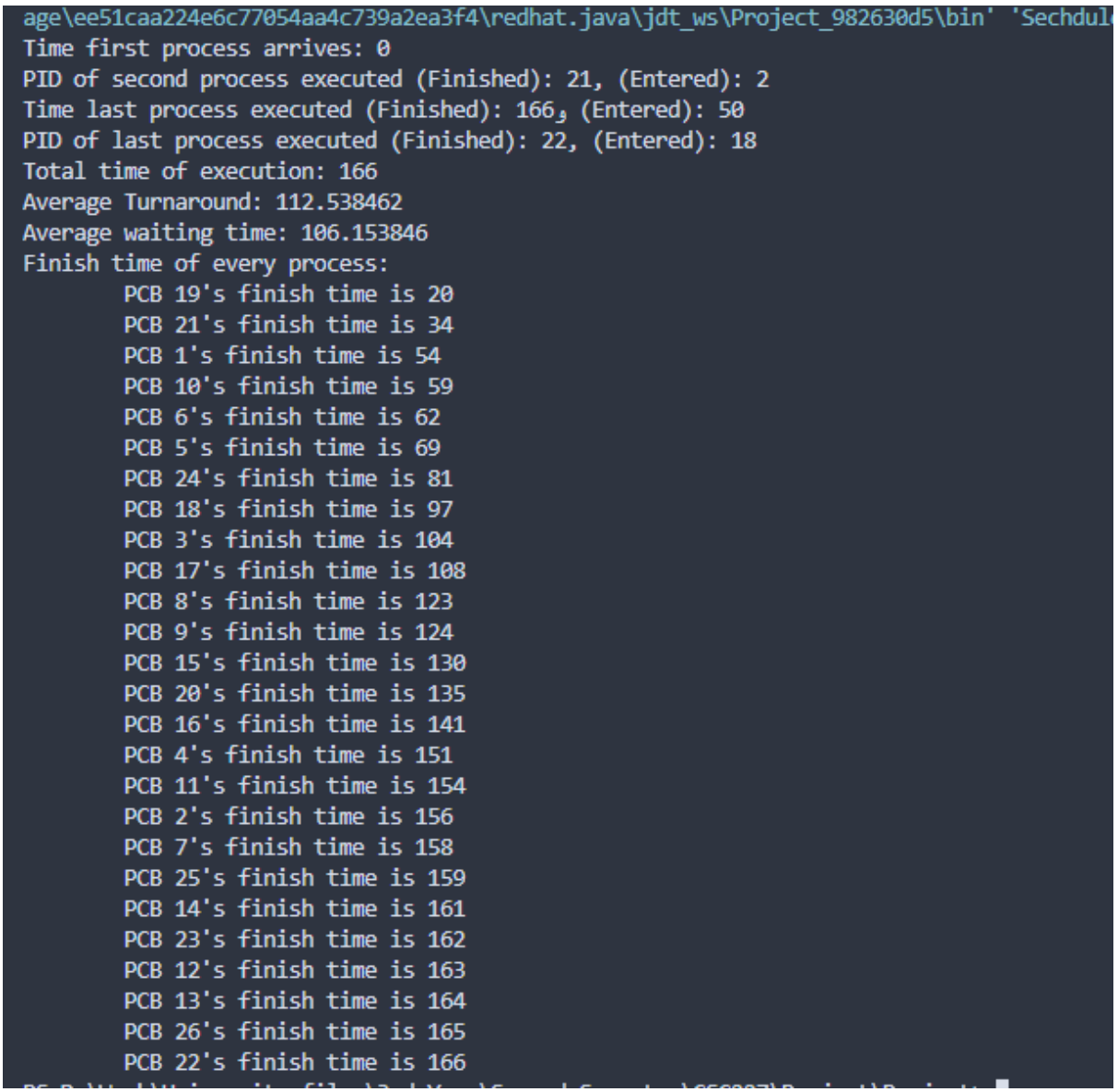
We needed a PCB block data structure to house the process's data. We created a PCB block which implements the comparable interface, which will allow us to sort the process based on time of arrival and PID.

We also used Java's Collections.sort which will sort our multidimensional array representing CSV based on the previously implemented comparable interface.

Finally we used a Queue implemented as Java's LinkedList<PCB> to simulate the dispatcher's job of feeding processes to the cpu.

```
class PCB implements Comparator<PCB> {
    int pid, submissionTime, finishTime, burstLeft, burstMax;

    public PCB(int pid, int submissionTime, int finishTime, int burst){···

    int getTurnaround(){
        return this.finishTime - this.submissionTime;
    }

    int getWaiting(){
        return this.getTurnaround() - this.burstMax;
    }

    public String toString(){
        return String.format("PID:%d SubmissionTime:%d FinishTime:%d BurstLeft:%d", pid, submissionTime, finishTime, burstLeft);
    }

    public int compare(PCB pcb1, PCB pcb2) {
        // Used to sort based on time and burst
        if(pcb1.submissionTime == pcb2.submissionTime)
            return pcb1.pid - pcb2.pid;
        return pcb1.submissionTime - pcb2.submissionTime;
    }
}
```

Results



Conclusion

Round Robin is an algorithm that prioritizes response over average turnaround, and that is apparent as we see multiple processes get cpu time in terms of a quantum. This is useful for applications where response is useful like user interfaces and other programs that deal with human input