COEN 346
PROGRAMMING ASSIGNEMENT 1

DESIGN REPORT

Ayman El Balaa
40200099

Objectives :

- Understanding scheduling algorithms and implementing them (First Come First Serve , Round Robin and Priority Round Robin)
- Evaluating the performance of each scheduling algorithm by calculating the turnaround and waiting time.
- Documenting the methodology of the project with well explained code and a concluding report.

Introduction and Theory :

In the world of computing , a CPU should efficiently allocate resources to the various processes. There are various ways in which that CPU can schedule the processes such as FCFS , RR and SJF, we will be focusing on the first two in this project.

Let's go over those different algorithms quickly :

First Come First Serve (FCFS)

By far the simplest CPU-scheduling algorithm is the first-come, first-served (FCFS) scheduling algorithm. With this scheme, the process that requests the CPU first is allocated the CPU first. When the CPU is free, it is allocated to the process at the head of the list. The running process is then removed from the queue.

Round Robin (RR)

The round-robin (RR) scheduling algorithm is similar to FCFS scheduling, but **preemption** is added to enable the system to switch between processes. A small unit of time, called a time quantum or time slice, is defined. A time quantum is generally from 10 to 100 milliseconds in length. The ready queue is treated as a circular queue. The CPU scheduler goes around the ready queue, allocating the CPU to each process for a time interval of up to 1 time quantum.

Priority Based Round Robin (PRI-RR)

Priority scheduling algorithm executes the processes depending upon their priority. Each process is allocated a priority and the process with the highest priority is executed first.It schedules tasks in order of priority and uses round robin scheduling for tasks with equal priority.

Programming :

The programming was done using **Java** and tested through Visual Studio Code on windows as well as manually compiling the java files and running them on Concordia's Linux PCs.

The following GitHub link was used to track changes and version control : https://github.com/Aymanbalaa/COEN346/tree/main/ProgAss

All Java files will also be attached to this report in a Zip folder.

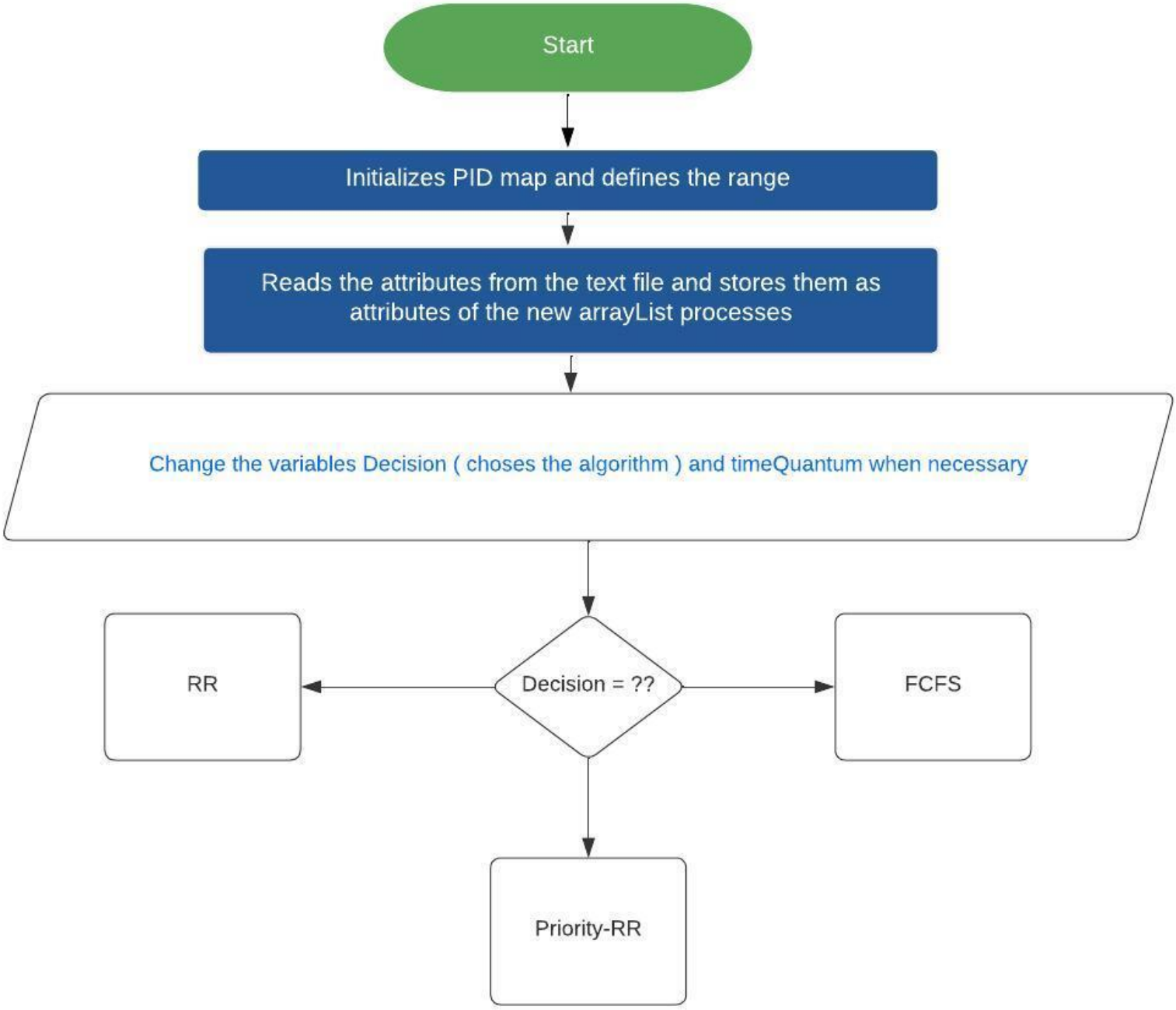
Methodology :

The program contains these different files , each defining a class with different member functions and fields. Using this Object-Oriented approach helped keep the code reusable and easily modifiable.

- Process.java
  Necessary to be able to create different processes with necessary attributes such as PID , name , timing …
- PidManager.java
  Implements different member functions to allocate and release PIDs for the currently active processes.
- Scheduler.java
  Abstract class that was used as a base class for the different scheduling algorithms.
- Main.java
  This is the actual runnable program it invokes functions from all classes , reads the information from the text file and starts scheduling.
- FCFS.java
- RR.java
- PriorityRR.java

Main Program Flowchart

The program can be summarized with this flowchart made using Lucid Chart



Algorithms Explained :

FCFS Algorithm Implementation :

Out of the three algorithms, FCFS was the easiest to implement. I first decided to sort the processes inside the array List based on their arrival time which is one of the attributes. The schedule() method of FCFS will then :

1. Fetch the first process in the processes list.
2. Create the process.
3. Wait for the process's burst to pass.
4. Terminate the process.
5. Iterate through the rest of the process in the list.

Round Robin (RR) Algorithm Implementation :

I implemented RR using a queue data structure. In the schedule() method, processes are continuously added to the queue as they arrive. The algorithm iteratively selects processes from the queue, executes them for a time quantum, updates turnaround and waiting times, and places the unfinished processes back in the queue. This continues until all processes are executed.

Priority Round Robin (PRI RR) Algorithm Implementation :

PRI RR combines the concepts of priority-based scheduling with round-robin scheduling. Each process is assigned a priority, and the queue is sorted based on these priorities. Processes with higher priorities are executed before lower-priority ones. The schedule() method follows a similar approach to RR, with the addition of sorting the queue based on process priority.

The following formulas are then used to calculate the average waiting and turnaround time :

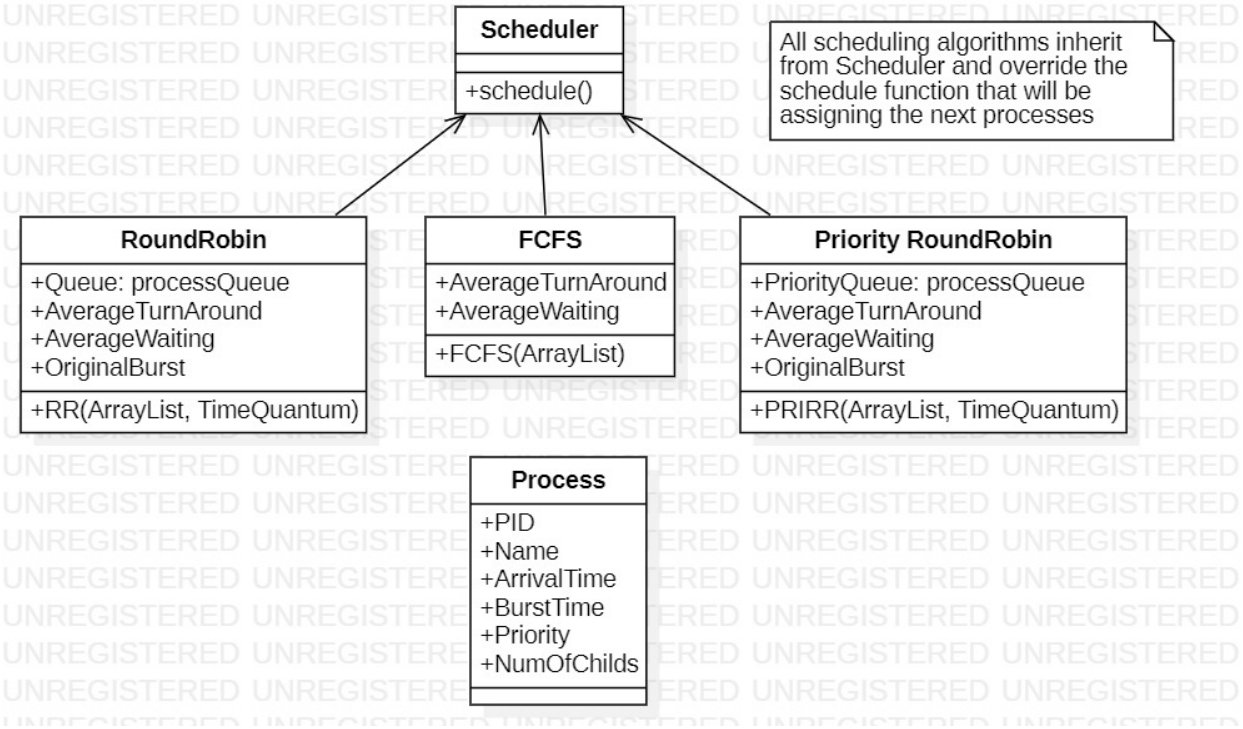
TurnAround = Completion Time – Arrival Time
Average TurnAround = SUM of TurnAround / Number of Processes

Waiting = Turnaround – Burst Time
Average Waiting = SUM of waiting / Number of Processes

Those values are then printed out after the continuous updates(process is created, waiting , resumed or deleted) are done and all processes are executed.

Data Structures Explained :

The following UML diagram was designed using STAR UML and outlines the data structures used in the implementation of this process scheduling program.Few notes will follow.



In Round Robin the **Queue** just needed to store the processes depending on when they come in meanwhile in Priority Round Robin, we had to keep their priorities in mind which is why we used a **Priority Queue.**

The diagram outlined mainly the differences and characteristics of each class , there are a lot of other member functions and attributes that are common or unique to each class that were not mentioned here for simplicity and clarity reasons.

Final Notes :

Our project is now complete ! All classes/functions are implemented with all necessary components and our report details and explains all methods used to implement this process scheduling simulation.

As we already have mentioned, the code is somewhat following an object-oriented strategy which makes it easy to change/modify in the future if we needed to add functionality (perhaps another scheduling algorithm ?)

In order to test the program it is crucial that the schedule.txt text file follow the same format as the one already attached with the project files or available on GitHub here since otherwise the program will have trouble assigning the attributes to the processes.

The scheduling algorithm can be easily changed (FCFS – RR -PRIRR) at the beginning of the Main.java alongside the timeQuantum.