Somya Singh CSE 7343 Spring 2017
Email: somyas@smu.edu
Student Id: 47304053

Course Project
Phase 1: Process Management

Table of Contents

Introduction:

The project is to create a CPU scheduling algorithm simiulator which will be able to emulate the process management usually used by OS. The project here is reading the list of process from a text file and based on user requirement is doing one of the below scheduling algorithm;

1. Shortest Job First (SJF)
2. First Come First Serve (FCFS)
3. Non-Preemptive Priority Scheduling (Priority)
4. Round Robin scheduling with Time Quantum Q

The program is also simulating the Process moving in and out of various Queues (READY vs WAITING) and uodating the state of the process according to the process state cycle. Used CLI User Interface and modular design fashion using diferrent modules based on the operating system function perform during CPU scheduling

Tools:

The project is built using ATOM IDE (we can use any IDE) and using java v1.8.

Email: somyas@smu.edu
Student Id: 47304053

Assumption:

The section list down the assumptions that have been made for the project.

- SJF – Process having shortest burst time executed first.
- FCFS - Process arrived first executed first.
- ROUND-ROBIN - Process executed for the time quatum provided by the user as an input move back to the queue after completing the particular time quatam as it state halted after a while return as a current job to execute .
- PRIORITY NON-PREEMPTIVE - Process executed according to the priority where 1 is treated as highest priority

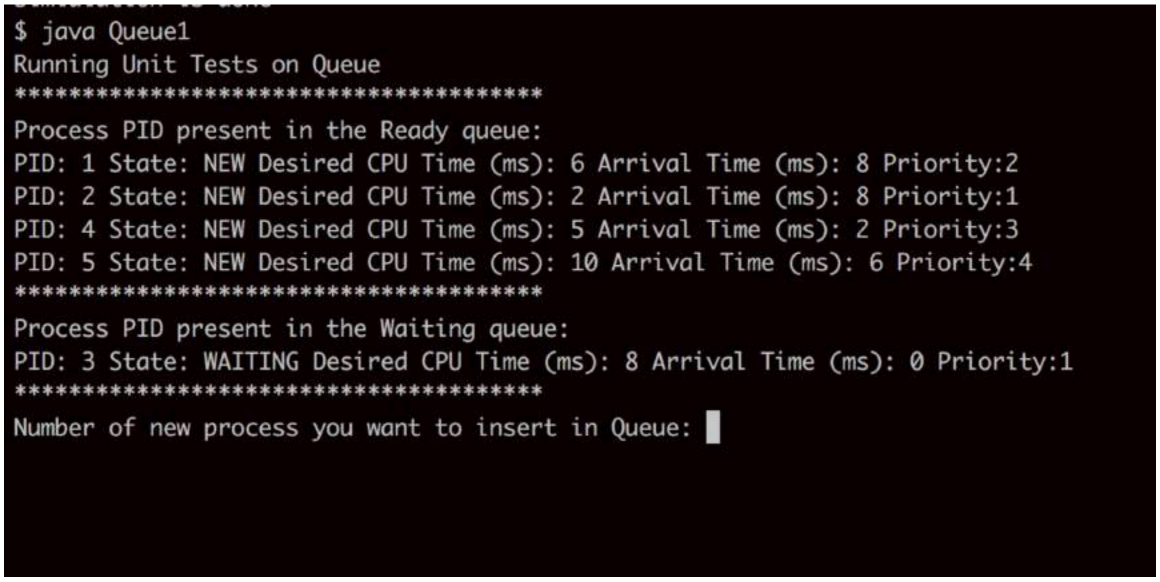
The Simulator is assuming all the processes are in the Queue but depending on its arrival time may or may not be Ready to be processed. In my project, I try to represent simulation of CPU scheduling using different aspect and concept of the operating system. I have consider the system current time in millisecond (unix tyme(epoch)) for calculating the time elapse between the process execution and the start, final and waiting time of the process. So I implied condition to match the system current system time with the process arrival time. I consider the state of the each process according to the process state cycle. So if the process is in the waiting state then with applying random probability process can change it state to ready. Maintain the state of each process as per the process state cycle. I am using various queue as Ready queue, Waiting queue as well as depending on the scheduling technique used priority queue and SJB Queue as a linked list and roundrobin queue as circular linked list.

Due to consideration of system current time in millisecond there is some overhead introduced in the program execution and writing various outputs and checks used in program.

How to Execute the Project

Part A:

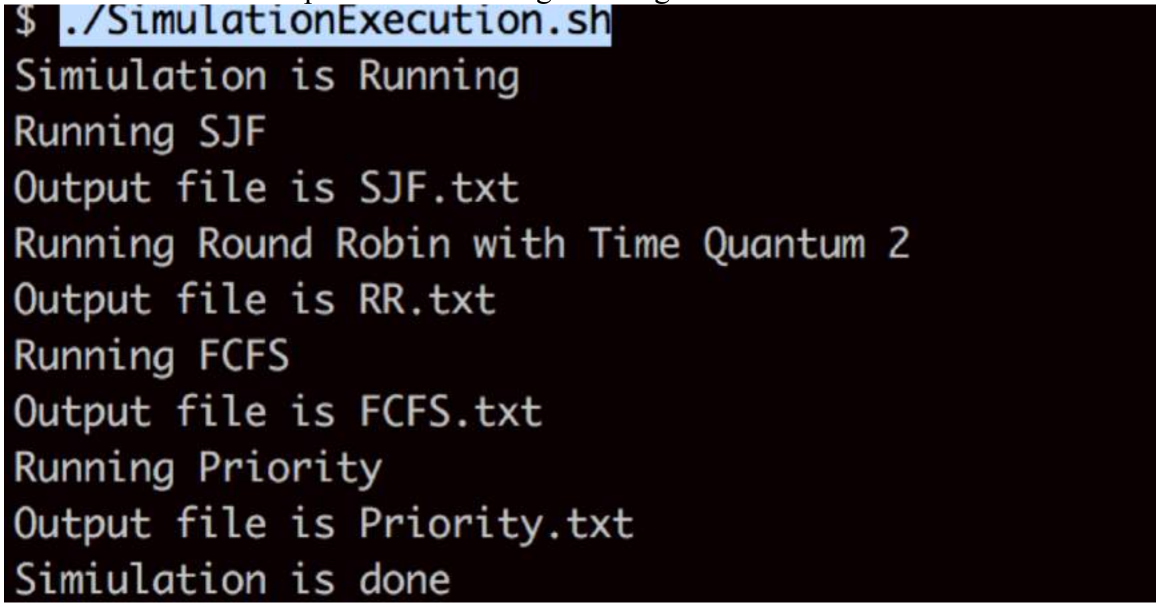
Step 1: Go to Terminal or CMD promt depnding on the Operating sytem you are working with.
Step 2: Navigate to the Project folder
Step 3: Type javac *.java to compile all the files
Step 4: In order to verify the PART A of the project execute "java Queue1"



```
$ java Queue1
Running Unit Tests on Queue
****************************************
Process PID present in the Ready queue:
PID: 1 State: NEW Desired CPU Time (ms): 6 Arrival Time (ms): 8 Priority:2
PID: 2 State: NEW Desired CPU Time (ms): 2 Arrival Time (ms): 8 Priority:1
PID: 4 State: NEW Desired CPU Time (ms): 5 Arrival Time (ms): 2 Priority:3
PID: 5 State: NEW Desired CPU Time (ms): 10 Arrival Time (ms): 6 Priority:4
****************************************
Process PID present in the Waiting queue:
PID: 3 State: WAITING Desired CPU Time (ms): 8 Arrival Time (ms): 0 Priority:1
****************************************
Number of new process you want to insert in Queue:
```

PART B:
For this part of project there are two ways to execute the scheduling algorithm.

Method 1: If you want to run all the algorithm at once you can execute by typing "./SimulationExecution.sh". This will create the ouptut file for each scheduling algorithm and one file which compares all the average waiting time.



```
$ ./SimulationExecution.sh
Simiulation is Running
Running SJF
Output file is SJF.txt
Running Round Robin with Time Quantum 2
Output file is RR.txt
Running FCFS
Output file is FCFS.txt
Running Priority
Output file is Priority.txt
Simiulation is done
```

Method 2:
If you want to run individual algorithm you can do it by typing as below;

1. java CPU "SJF" 0 to excute SJF
2. java CPU "ROUND ROBIN" 2 to execute Round-Robin with Time Quantum of n
3. java CPU "FCFS" 0 to run FCFS

4. java CPU "PRIORITY" 0 to run Non-Preemetive Priority scheduling.

The output files are:

1. SJF.txt for SJF
2. FCFS.txt for FCFS
3. RR.txt for RoundRobin
4. Priority.txt for Priority
5. ProjectOutput.txt to comapre average waiting time comparison.

Sample output files are also there for quick reference.

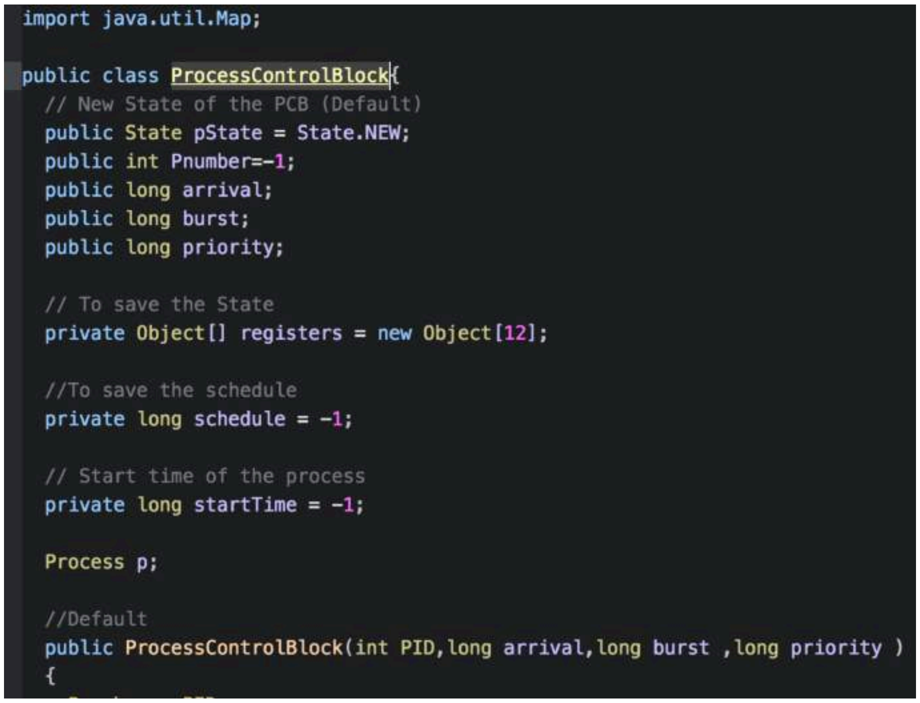
1. SJFOutput.txt
2. RoundRobinOutput.txt
3. FCFSOutput.txt
4. PRIORITYOutput.txt

Project Report:

PART A

1. Each process will be represented by its Process Control Block (PCB) and each process will be given a priority

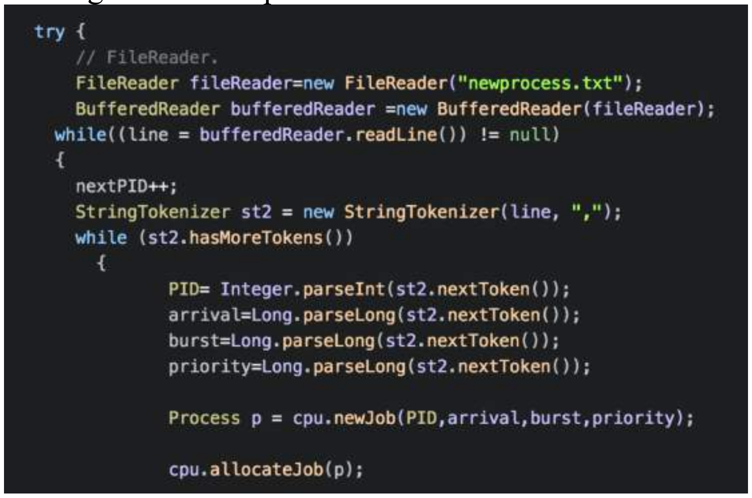
   In order to define PCB for each process I am calling ProcessControlBlock.java from my main program to create a PCB for each process that user has provided to simulation.



```java
import java.util.Map;

public class ProcessControlBlock{
  // New State of the PCB (Default)
  public State pState = State.NEW;
  public int Pnumber=-1;
  public long arrival;
  public long burst;
  public long priority;

  // To save the State
  private Object[] registers = new Object[12];

  //To save the schedule
  private long schedule = -1;

  // Start time of the process
  private long startTime = -1;

  Process p;

  //Default
  public ProcessControlBlock(int PID,long arrival,long burst ,long priority )
  {
```

2. Each process info can be entered manually by user input or loaded from text file. Proper Error Handling is expected.

   Using Java file reader and user input of desired algorithm I am reading the text file to get the list of process.

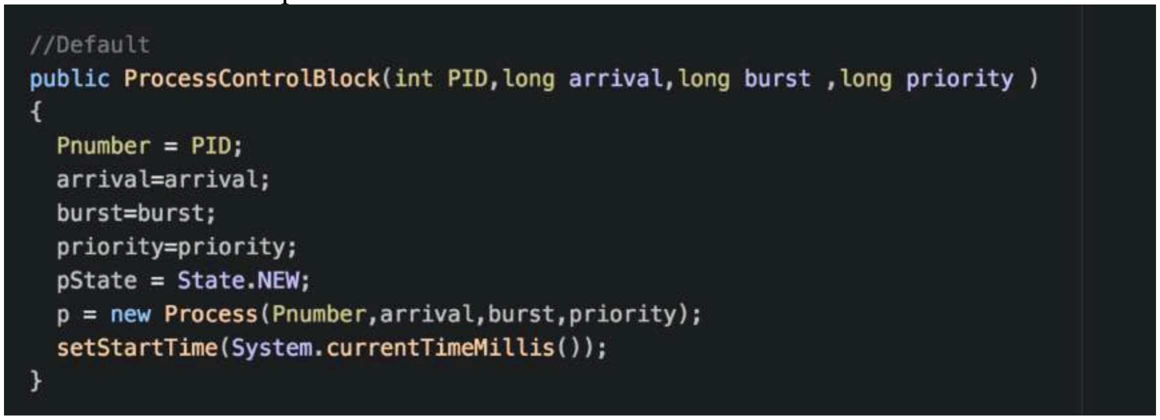


```java
try {
    // FileReader.
    FileReader fileReader=new FileReader("newprocess.txt");
    BufferedReader bufferedReader =new BufferedReader(fileReader);
  while((line = bufferedReader.readLine()) != null)
  {
    nextPID++;
    StringTokenizer st2 = new StringTokenizer(line, ",");
    while (st2.hasMoreTokens())
      {
            PID= Integer.parseInt(st2.nextToken());
            arrival=Long.parseLong(st2.nextToken());
            burst=Long.parseLong(st2.nextToken());
            priority=Long.parseLong(st2.nextToken());

            Process p = cpu.newJob(PID,arrival,burst,priority);

            cpu.allocateJob(p);
```

3. Each PCB should have a uniform format

   The process have following PCB format;
   1. Process ID PID
   2. Burst time: burst
   3. Priority: priority
   4. Arrival time: arrival

5. State: state of the process

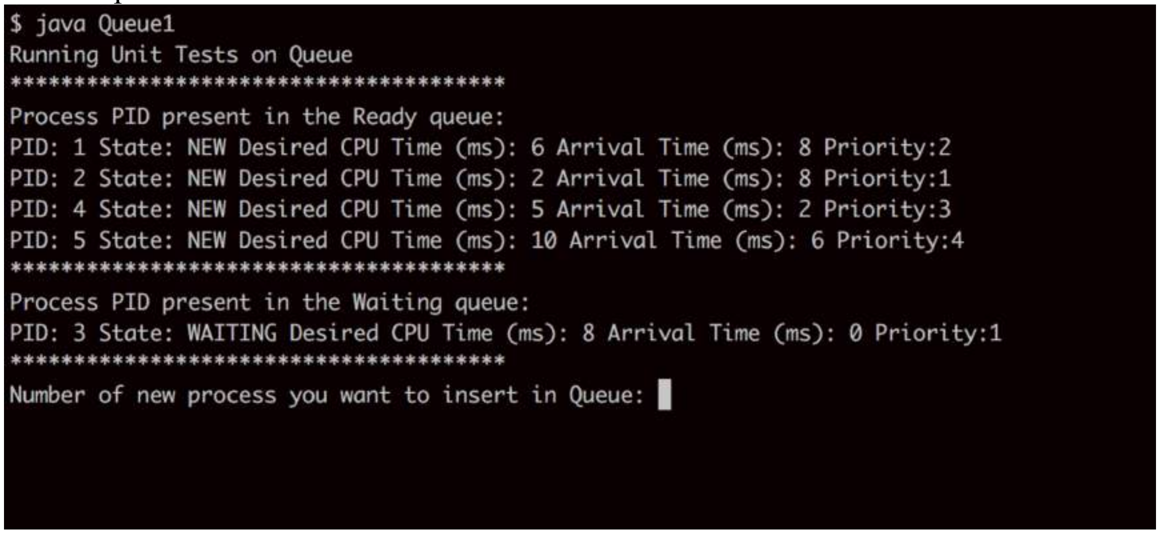


```
//Default
public ProcessControlBlock(int PID,long arrival,long burst ,long priority )
{
  Pnumber = PID;
  arrival=arrival;
  burst=burst;
  priority=priority;
  pState = State.NEW;
  p = new Process(Pnumber,arrival,burst,priority);
  setStartTime(System.currentTimeMillis());
}
```

4. The collections of PCBs form single or doubly linked lists for all queues implemented in the system

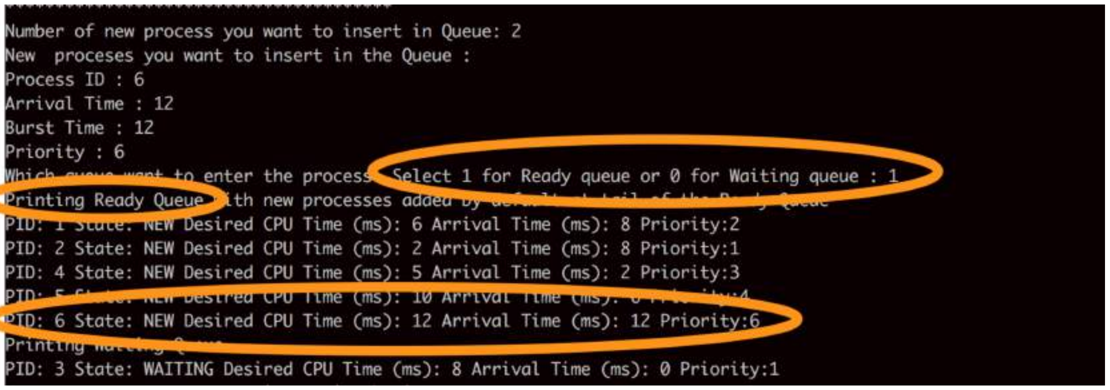
   I maintain various queue as Ready queue, Waiting queue using single link-list. Also depending on the scheduling technique used Priority, SJB and FCFS Queue are using single linked list and Round-Robin queue as circular linked list.

5. The Operation on Queue
   To implement this part of the projct I have created "**Queue1.java**" and "**ProcessControlBlock1.java**".
   The features are:
   1. The program reads the process list from a user input file (newprocess.txt) and load the process into PCB.



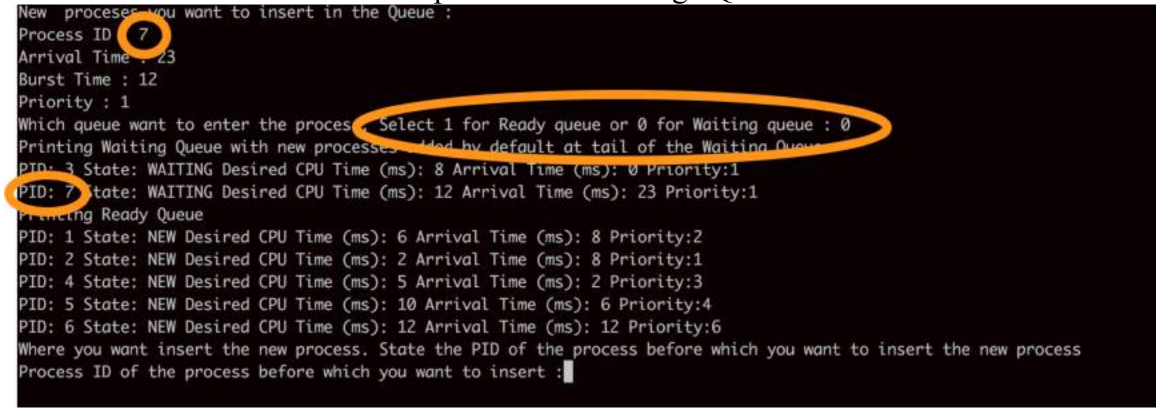
```
$ java Queue1
Running Unit Tests on Queue
****************************************
Process PID present in the Ready queue:
PID: 1 State: NEW Desired CPU Time (ms): 6 Arrival Time (ms): 8 Priority:2
PID: 2 State: NEW Desired CPU Time (ms): 2 Arrival Time (ms): 8 Priority:1
PID: 4 State: NEW Desired CPU Time (ms): 5 Arrival Time (ms): 2 Priority:3
PID: 5 State: NEW Desired CPU Time (ms): 10 Arrival Time (ms): 6 Priority:4
****************************************
Process PID present in the Waiting queue:
PID: 3 State: WAITING Desired CPU Time (ms): 8 Arrival Time (ms): 0 Priority:1
****************************************
Number of new process you want to insert in Queue:
```

   2. The simulation also gives the user an option to add a process in either "Ready" or "Waiting" Queue. Below you can see the user add a process in **Ready Queue with PID 6. The default position is tail**



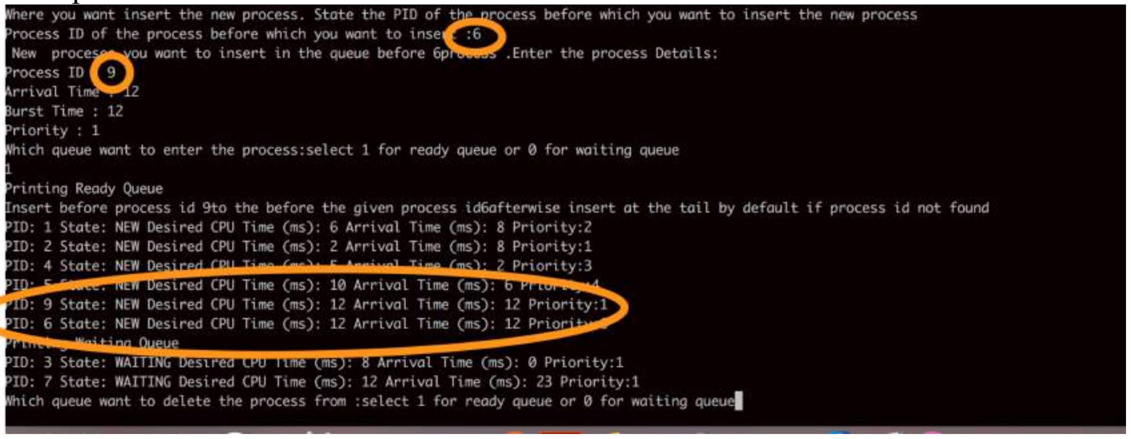
```
Number of new process you want to insert in Queue: 2
New  proceses you want to insert in the Queue :
Process ID : 6
Arrival Time : 12
Burst Time : 12
Priority : 6
Which queue want to enter the process Select 1 for Ready queue or 0 for Waiting queue : 1
Printing Ready Queue with new processes added by default at tail of the Ready Queue
PID: 1 State: NEW Desired CPU Time (ms): 6 Arrival Time (ms): 8 Priority:2
PID: 2 State: NEW Desired CPU Time (ms): 2 Arrival Time (ms): 8 Priority:1
PID: 4 State: NEW Desired CPU Time (ms): 5 Arrival Time (ms): 2 Priority:3
PID: 5 State: NEW Desired CPU Time (ms): 10 Arrival Time (ms): 6 Priority:4
PID: 6 State: NEW Desired CPU Time (ms): 12 Arrival Time (ms): 12 Priority:6
Printing Waiting Queue
PID: 3 State: WAITING Desired CPU Time (ms): 8 Arrival Time (ms): 0 Priority:1
```

The below user is able to insert a process in "Waiting" Queue.



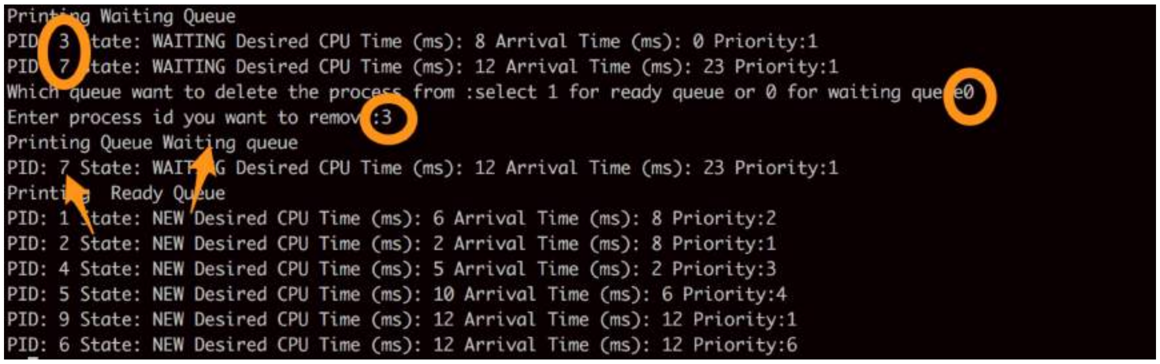
```
New  proceses you want to insert in the Queue :
Process ID : 7
Arrival Time : 23
Burst Time : 12
Priority : 1
Which queue want to enter the process Select 1 for Ready queue or 0 for Waiting queue : 0
Printing Waiting Queue with new processes added by default at tail of the Waiting Queue
PID: 3 State: WAITING Desired CPU Time (ms): 8 Arrival Time (ms): 0 Priority:1
PID: 7 State: WAITING Desired CPU Time (ms): 12 Arrival Time (ms): 23 Priority:1
Printing Ready Queue
PID: 1 State: NEW Desired CPU Time (ms): 6 Arrival Time (ms): 8 Priority:2
PID: 2 State: NEW Desired CPU Time (ms): 2 Arrival Time (ms): 8 Priority:1
PID: 4 State: NEW Desired CPU Time (ms): 5 Arrival Time (ms): 2 Priority:3
PID: 5 State: NEW Desired CPU Time (ms): 10 Arrival Time (ms): 6 Priority:4
PID: 6 State: NEW Desired CPU Time (ms): 12 Arrival Time (ms): 12 Priority:6
Where you want insert the new process. State the PID of the process before which you want to insert the new process
Process ID of the process before which you want to insert :
```

3. The simulation also gives user an option to insert a process between any current process



```
Where you want insert the new process. State the PID of the process before which you want to insert the new process
Process ID of the process before which you want to insert :6
 New  proceses you want to insert in the queue before 6process .Enter the process Details:
Process ID : 9
Arrival Time : 12
Burst Time : 12
Priority : 1
Which queue want to enter the process:select 1 for ready queue or 0 for waiting queue
1
Printing Ready Queue
Insert before process id 9to the before the given process id6afterwise insert at the tail by default if process id not found
PID: 1 State: NEW Desired CPU Time (ms): 6 Arrival Time (ms): 8 Priority:2
PID: 2 State: NEW Desired CPU Time (ms): 2 Arrival Time (ms): 8 Priority:1
PID: 4 State: NEW Desired CPU Time (ms): 5 Arrival Time (ms): 2 Priority:3
PID: 5 State: NEW Desired CPU Time (ms): 10 Arrival Time (ms): 6 Priority:4
PID: 9 State: NEW Desired CPU Time (ms): 12 Arrival Time (ms): 12 Priority:1
PID: 6 State: NEW Desired CPU Time (ms): 12 Arrival Time (ms): 12 Priority:6
Printing Waiting Queue
PID: 3 State: WAITING Desired CPU Time (ms): 8 Arrival Time (ms): 0 Priority:1
PID: 7 State: WAITING Desired CPU Time (ms): 12 Arrival Time (ms): 23 Priority:1
Which queue want to delete the process from :select 1 for ready queue or 0 for waiting queue
```

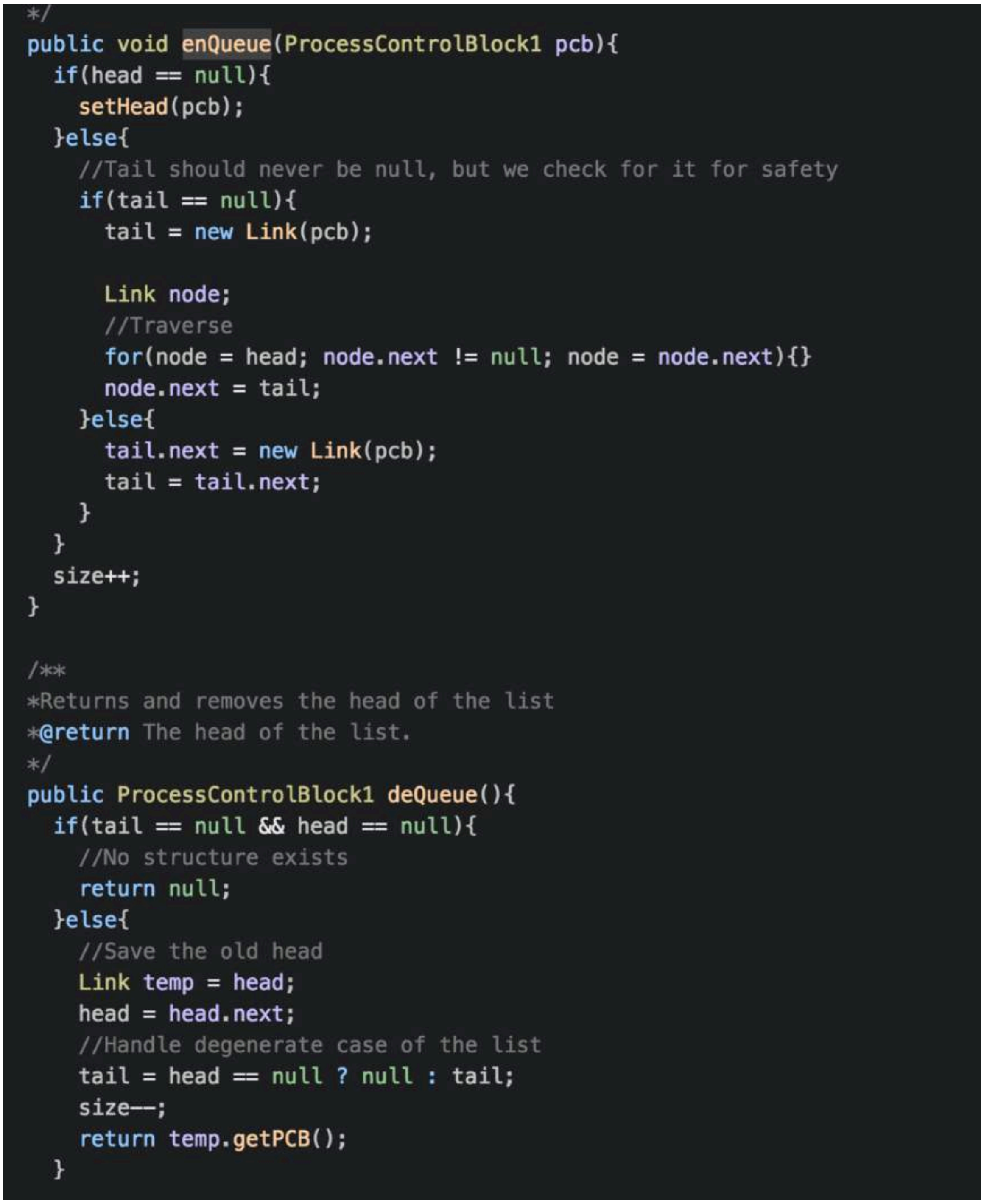
4. The simulation also provide user with an option to delete the Process.



```
Printing Waiting Queue
PID: 3 State: WAITING Desired CPU Time (ms): 8 Arrival Time (ms): 0 Priority:1
PID: 7 State: WAITING Desired CPU Time (ms): 12 Arrival Time (ms): 23 Priority:1
Which queue want to delete the process from :select 1 for ready queue or 0 for waiting queue0
Enter process id you want to remove:3
Printing Queue Waiting queue
PID: 7 State: WAITING Desired CPU Time (ms): 12 Arrival Time (ms): 23 Priority:1
Printing Ready Queue
PID: 1 State: NEW Desired CPU Time (ms): 6 Arrival Time (ms): 8 Priority:2
PID: 2 State: NEW Desired CPU Time (ms): 2 Arrival Time (ms): 8 Priority:1
PID: 4 State: NEW Desired CPU Time (ms): 5 Arrival Time (ms): 2 Priority:3
PID: 5 State: NEW Desired CPU Time (ms): 10 Arrival Time (ms): 6 Priority:4
PID: 9 State: NEW Desired CPU Time (ms): 12 Arrival Time (ms): 12 Priority:1
PID: 6 State: NEW Desired CPU Time (ms): 12 Arrival Time (ms): 12 Priority:6
```

6. The control program will call on different functions/methods to add/delete PCBs.

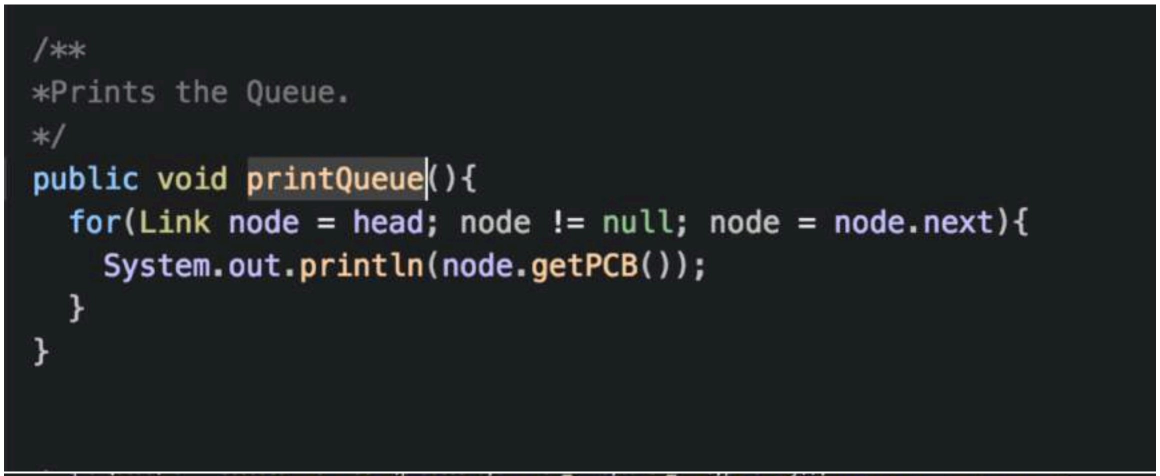
   We have created below thwo functions which are going to add and remove PCB's



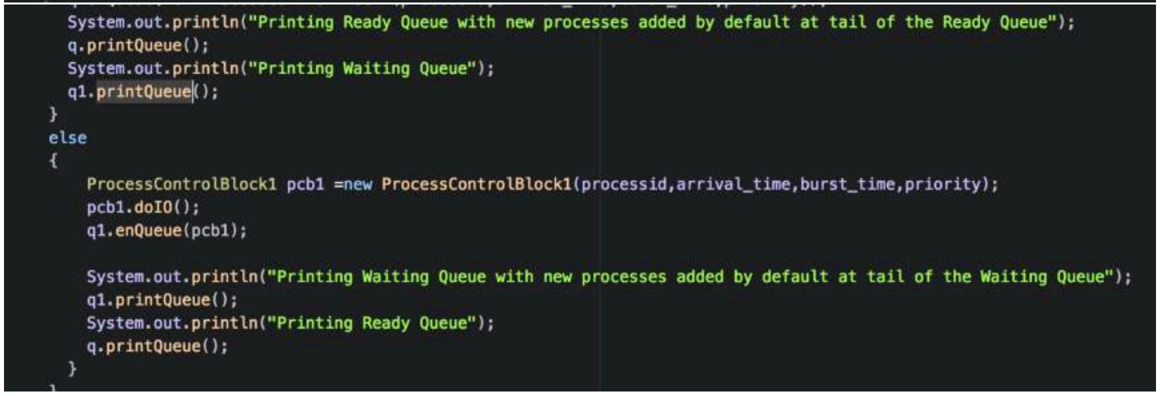
```
*/
public void enQueue(ProcessControlBlock1 pcb){
  if(head == null){
    setHead(pcb);
  }else{
    //Tail should never be null, but we check for it for safety
    if(tail == null){
      tail = new Link(pcb);

      Link node;
      //Traverse
      for(node = head; node.next != null; node = node.next){}
      node.next = tail;
    }else{
      tail.next = new Link(pcb);
      tail = tail.next;
    }
  }
  size++;
}

/**
*Returns and removes the head of the list
*@return The head of the list.
*/
public ProcessControlBlock1 deQueue(){
  if(tail == null && head == null){
    //No structure exists
    return null;
  }else{
    //Save the old head
    Link temp = head;
    head = head.next;
    //Handle degenerate case of the list
    tail = head == null ? null : tail;
    size--;
    return temp.getPCB();
  }
```

7. You also need some utility functions/methods to show the contents of different queues

   For this I have implemented a function called “printQueue”.



```
/**
*Prints the Queue.
*/
public void printQueue(){
  for(Link node = head; node != null; node = node.next){
    System.out.println(node.getPCB());
  }
}
```



```
    System.out.println("Printing Ready Queue with new processes added by default at tail of the Ready Queue");
    q.printQueue();
    System.out.println("Printing Waiting Queue");
    q1.printQueue();
  }
  else
  {
      ProcessControlBlock1 pcb1 =new ProcessControlBlock1(processid,arrival_time,burst_time,priority);
      pcb1.doIO();
      q1.enQueue(pcb1);

      System.out.println("Printing Waiting Queue with new processes added by default at tail of the Waiting Queue");
      q1.printQueue();
      System.out.println("Printing Ready Queue");
      q.printQueue();
    }
```

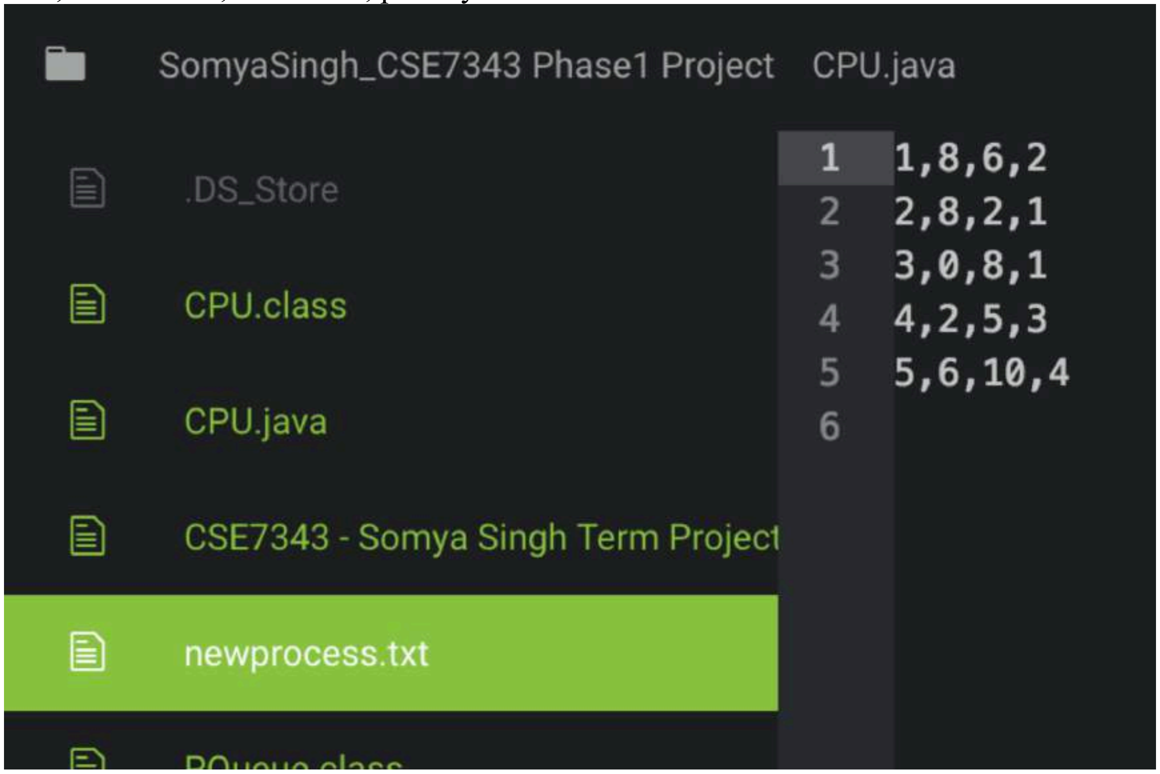
PART B: Implemeneting Scheduler

For this part of my project below are the important files with there mechanism;

1. CPU.java: This is my main program which is doing below functionality;
    a. Reading the input file to get the list of all the processes.
    b. Calls Queue java file to create a PCB for each process that user has defined in the input file
    c. Schedule the PCB into the Queues using different scheduler.
    d. Execute the jobs by fetching the processe from their Queues
    e. If the process completes successfully it TERMINATES the process and remove it from Queue by changing its state to TERMINATED. Then it calculates the Start time, end time and total Wait time for each process
    f. Finally the program prints the "GANTT CHART" and average waiting time.
2. Scheduler.java: This is the main scheduling program which calls the respective Queues to execute the process in the CPU.java.
3. Queue.java / SJBQueue / RoundRobinQueue / PriorityQueue : This helps to arrange the processes into there respective order based on scheduling algortihm used

4. ProcessControlBlock.java: This forms the PCB for process
5. State.java: which defines the various state of process

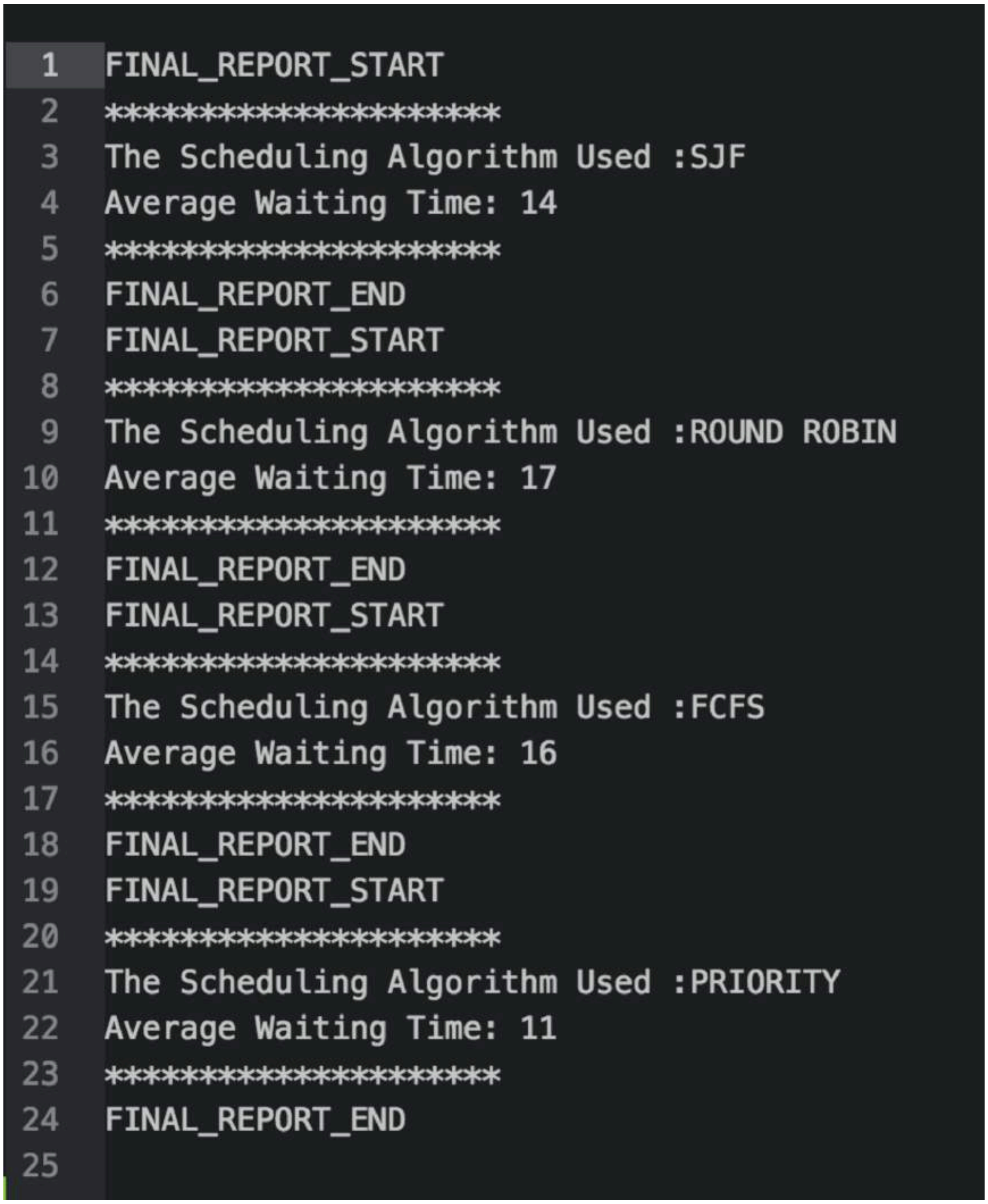
Input file is "newprocess.txt". The columns are
PID, arrival time, burst time, priority


SomyaSingh_CSE7343 Phase1 Project
CPU.java
.DS_Store
CPU.class
CPU.java
CSE7343 - Somya Singh Term Project
newprocess.txt
PQueue.class

```
1,8,6,2
2,8,2,1
3,0,8,1
4,2,5,3
5,6,10,4
```

The "SimulationExecution.sh" is the shell script which runs all the Scheduling algorithm at once and create an output file for each Scheduling algorithm. **It also create one single file which will have all the Average Waiting time "ProjectOutput.txt"**

Email: somyas@smu.edu
Student Id: 47304053

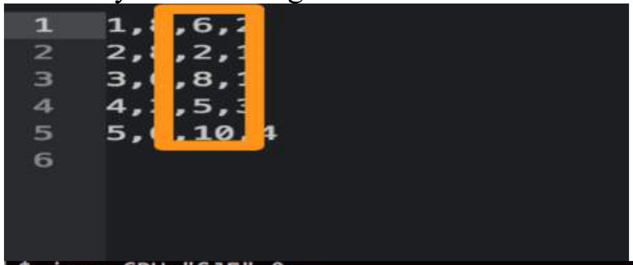


```
FINAL_REPORT_START
*********************
The Scheduling Algorithm Used :SJF
Average Waiting Time: 14
*********************
FINAL_REPORT_END
FINAL_REPORT_START
*********************
The Scheduling Algorithm Used :ROUND ROBIN
Average Waiting Time: 17
*********************
FINAL_REPORT_END
FINAL_REPORT_START
*********************
The Scheduling Algorithm Used :FCFS
Average Waiting Time: 16
*********************
FINAL_REPORT_END
FINAL_REPORT_START
*********************
The Scheduling Algorithm Used :PRIORITY
Average Waiting Time: 11
*********************
FINAL_REPORT_END
```

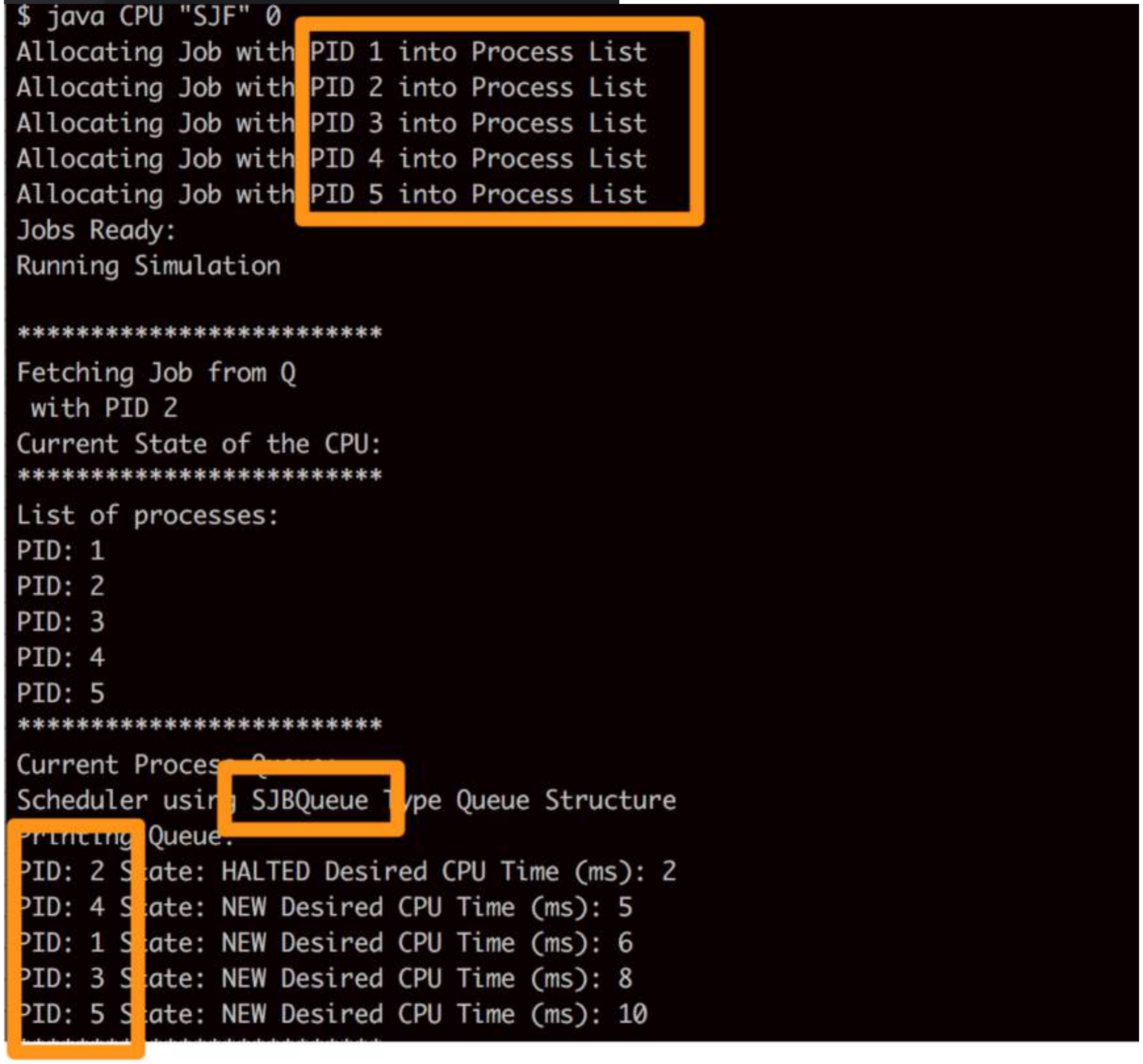
Some of the sample screen shots are attached below:

For the below screen shot you can see the first step is allocating the job for each PID in the file and then depending on the algorithm it will pick and process the processes.

For below we are using SJF which based on my input file should be starting with PID 2 followed by PID 4, 1,3 and last 5.
This is by lowest to highest burst time.

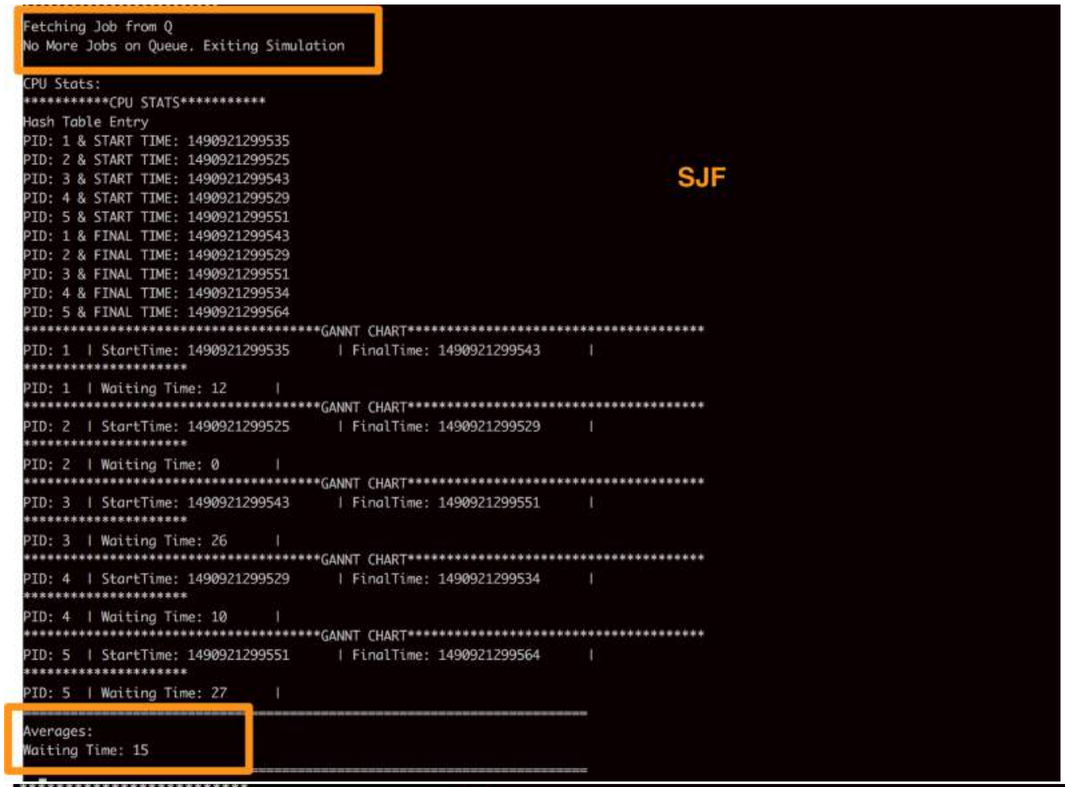


```
1,[illegible],6,[illegible]
2,[illegible],2,[illegible]
3,[illegible],8,[illegible]
4,[illegible],5,[illegible]
5,[illegible],10,4
```

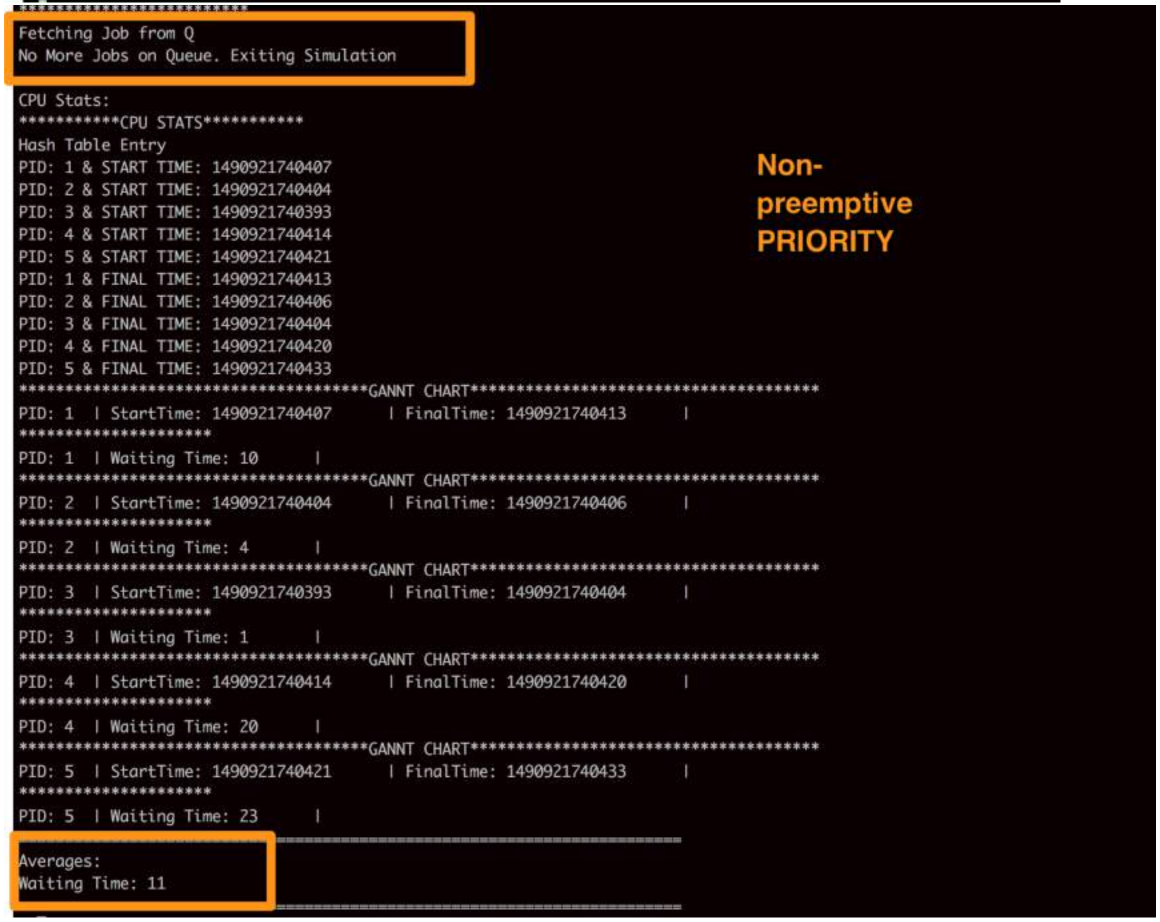


```
$ java CPU "SJF" 0
Allocating Job with PID 1 into Process List
Allocating Job with PID 2 into Process List
Allocating Job with PID 3 into Process List
Allocating Job with PID 4 into Process List
Allocating Job with PID 5 into Process List
Jobs Ready:
Running Simulation

************************
Fetching Job from Q
 with PID 2
Current State of the CPU:
************************
List of processes:
PID: 1
PID: 2
PID: 3
PID: 4
PID: 5
************************
Current Process Queue:
Scheduler using SJBQueue Type Queue Structure
Printing Queue:
PID: 2 State: HALTED Desired CPU Time (ms): 2
PID: 4 State: NEW Desired CPU Time (ms): 5
PID: 1 State: NEW Desired CPU Time (ms): 6
PID: 3 State: NEW Desired CPU Time (ms): 8
PID: 5 State: NEW Desired CPU Time (ms): 10
```

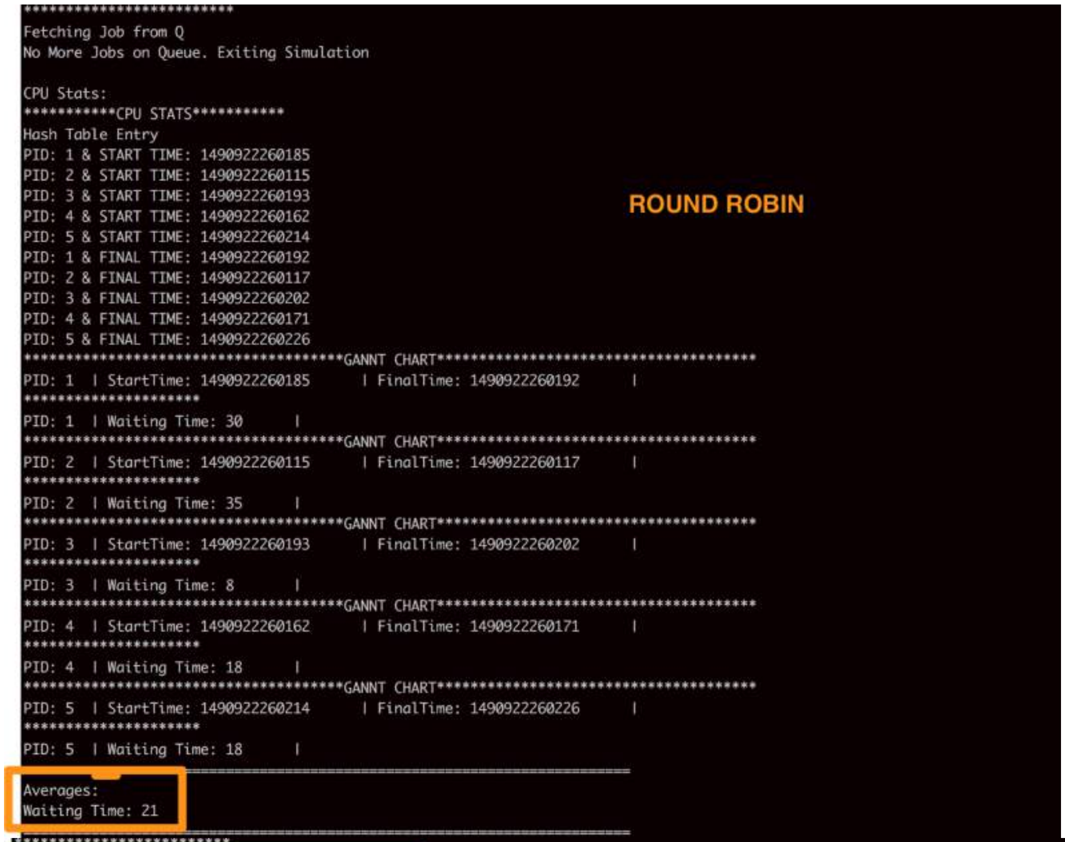
Finally once the scheduler is completed the user will be able to see the "GANTT Chart" with each process Start time, End Time , Waiting time and Average Waiting time



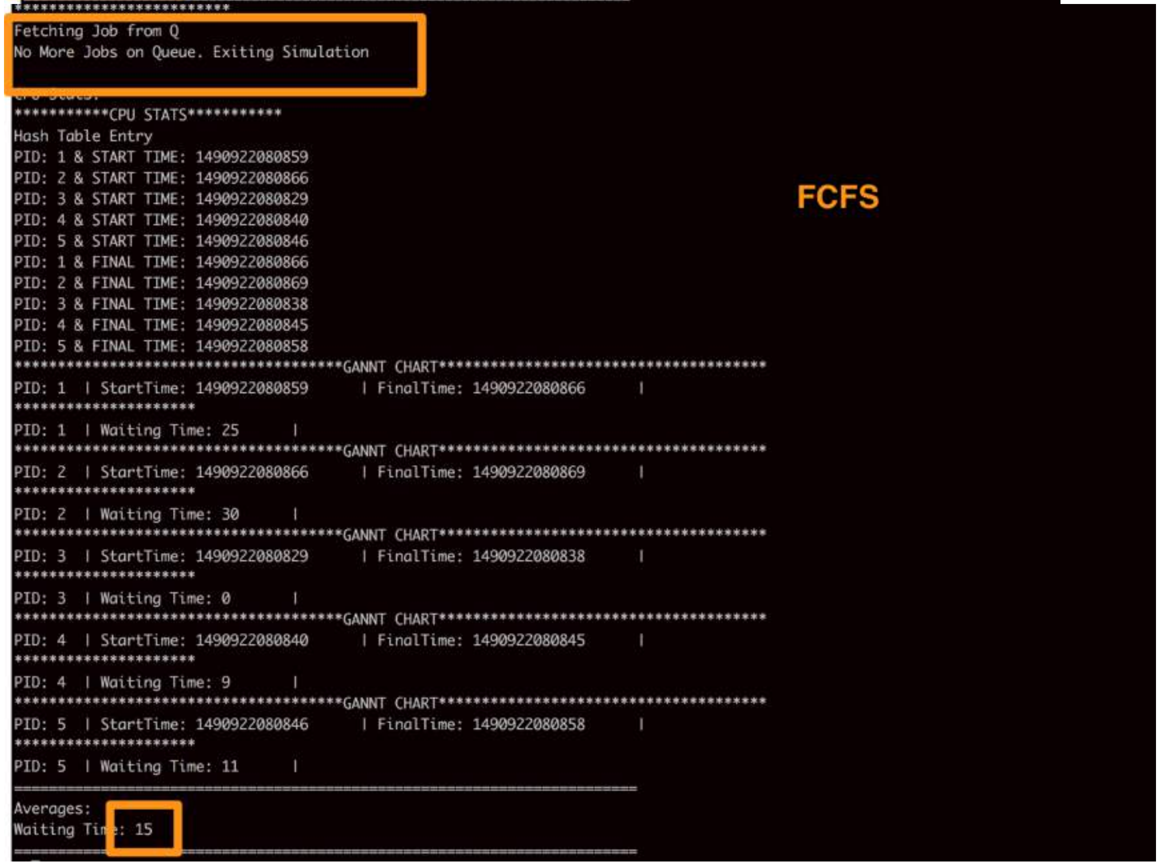
```
Fetching Job from Q
No More Jobs on Queue. Exiting Simulation

CPU Stats:
***********CPU STATS***********
Hash Table Entry
PID: 1 & START TIME: 1490921299535
PID: 2 & START TIME: 1490921299525
PID: 3 & START TIME: 1490921299543
PID: 4 & START TIME: 1490921299529
PID: 5 & START TIME: 1490921299551
PID: 1 & FINAL TIME: 1490921299543
PID: 2 & FINAL TIME: 1490921299529
PID: 3 & FINAL TIME: 1490921299551
PID: 4 & FINAL TIME: 1490921299534
PID: 5 & FINAL TIME: 1490921299564
*************************************GANNT CHART***************************************
PID: 1  | StartTime: 1490921299535      | FinalTime: 1490921299543      |
********************
PID: 1  | Waiting Time: 12       |
*************************************GANNT CHART***************************************
PID: 2  | StartTime: 1490921299525      | FinalTime: 1490921299529      |
********************
PID: 2  | Waiting Time: 0        |
*************************************GANNT CHART***************************************
PID: 3  | StartTime: 1490921299543      | FinalTime: 1490921299551      |
********************
PID: 3  | Waiting Time: 26       |
*************************************GANNT CHART***************************************
PID: 4  | StartTime: 1490921299529      | FinalTime: 1490921299534      |
********************
PID: 4  | Waiting Time: 10       |
*************************************GANNT CHART***************************************
PID: 5  | StartTime: 1490921299551      | FinalTime: 1490921299564      |
********************
PID: 5  | Waiting Time: 27       |
----------------------------------------------------------------------
Averages:
Waiting Time: 15
----------------------------------------------------------------------
```



```
Fetching Job from Q
No More Jobs on Queue. Exiting Simulation

CPU Stats:
***********CPU STATS***********
Hash Table Entry
PID: 1 & START TIME: 1490921740407
PID: 2 & START TIME: 1490921740404
PID: 3 & START TIME: 1490921740393
PID: 4 & START TIME: 1490921740414
PID: 5 & START TIME: 1490921740421
PID: 1 & FINAL TIME: 1490921740413
PID: 2 & FINAL TIME: 1490921740406
PID: 3 & FINAL TIME: 1490921740404
PID: 4 & FINAL TIME: 1490921740420
PID: 5 & FINAL TIME: 1490921740433
*************************************GANNT CHART***************************************
PID: 1  | StartTime: 1490921740407      | FinalTime: 1490921740413      |
********************
PID: 1  | Waiting Time: 10       |
*************************************GANNT CHART***************************************
PID: 2  | StartTime: 1490921740404      | FinalTime: 1490921740406      |
********************
PID: 2  | Waiting Time: 4        |
*************************************GANNT CHART***************************************
PID: 3  | StartTime: 1490921740393      | FinalTime: 1490921740404      |
********************
PID: 3  | Waiting Time: 1        |
*************************************GANNT CHART***************************************
PID: 4  | StartTime: 1490921740414      | FinalTime: 1490921740420      |
********************
PID: 4  | Waiting Time: 20       |
*************************************GANNT CHART***************************************
PID: 5  | StartTime: 1490921740421      | FinalTime: 1490921740433      |
********************
PID: 5  | Waiting Time: 23       |
----------------------------------------------------------------------
Averages:
Waiting Time: 11
----------------------------------------------------------------------
```

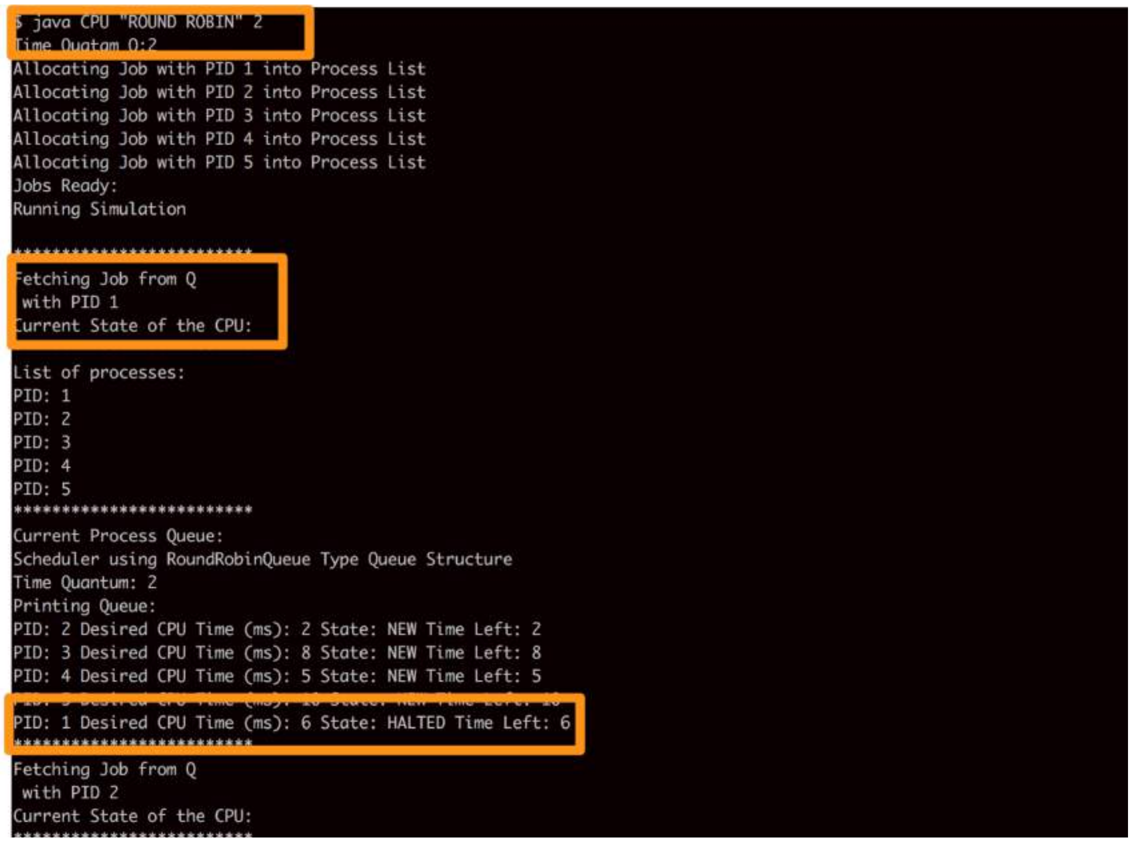


```
*************************
Fetching Job from Q
No More Jobs on Queue. Exiting Simulation

CPU Stats:
***********CPU STATS***********
Hash Table Entry
PID: 1 & START TIME: 1490922260185
PID: 2 & START TIME: 1490922260115
PID: 3 & START TIME: 1490922260193
PID: 4 & START TIME: 1490922260162
PID: 5 & START TIME: 1490922260214
PID: 1 & FINAL TIME: 1490922260192
PID: 2 & FINAL TIME: 1490922260117
PID: 3 & FINAL TIME: 1490922260202
PID: 4 & FINAL TIME: 1490922260171
PID: 5 & FINAL TIME: 1490922260226
*************************************GANNT CHART***************************************
PID: 1  | StartTime: 1490922260185      | FinalTime: 1490922260192      |
*********************
PID: 1  | Waiting Time: 30      |
*************************************GANNT CHART***************************************
PID: 2  | StartTime: 1490922260115      | FinalTime: 1490922260117      |
*********************
PID: 2  | Waiting Time: 35      |
*************************************GANNT CHART***************************************
PID: 3  | StartTime: 1490922260193      | FinalTime: 1490922260202      |
*********************
PID: 3  | Waiting Time: 8       |
*************************************GANNT CHART***************************************
PID: 4  | StartTime: 1490922260162      | FinalTime: 1490922260171      |
*********************
PID: 4  | Waiting Time: 18      |
*************************************GANNT CHART***************************************
PID: 5  | StartTime: 1490922260214      | FinalTime: 1490922260226      |
*********************
PID: 5  | Waiting Time: 18      |

Averages:
Waiting Time: 21
```



```
*************************
Fetching Job from Q
No More Jobs on Queue. Exiting Simulation

***********CPU STATS***********
Hash Table Entry
PID: 1 & START TIME: 1490922080859
PID: 2 & START TIME: 1490922080866
PID: 3 & START TIME: 1490922080829
PID: 4 & START TIME: 1490922080840
PID: 5 & START TIME: 1490922080846
PID: 1 & FINAL TIME: 1490922080866
PID: 2 & FINAL TIME: 1490922080869
PID: 3 & FINAL TIME: 1490922080838
PID: 4 & FINAL TIME: 1490922080845
PID: 5 & FINAL TIME: 1490922080858
*************************************GANNT CHART***************************************
PID: 1  | StartTime: 1490922080859      | FinalTime: 1490922080866      |
*********************
PID: 1  | Waiting Time: 25      |
*************************************GANNT CHART***************************************
PID: 2  | StartTime: 1490922080866      | FinalTime: 1490922080869      |
*********************
PID: 2  | Waiting Time: 30      |
*************************************GANNT CHART***************************************
PID: 3  | StartTime: 1490922080829      | FinalTime: 1490922080838      |
*********************
PID: 3  | Waiting Time: 0       |
*************************************GANNT CHART***************************************
PID: 4  | StartTime: 1490922080840      | FinalTime: 1490922080845      |
*********************
PID: 4  | Waiting Time: 9       |
*************************************GANNT CHART***************************************
PID: 5  | StartTime: 1490922080846      | FinalTime: 1490922080858      |
*********************
PID: 5  | Waiting Time: 11      |

Averages:
Waiting Ti : 15
```

Also for Round Robin based on user input of Time quantum the schedular is moving process in and out of active working. For below you can see the Time Quantum defines is 2 and the Process burst time was 8 so it executed it for 2 sec and then moved the process to HALTED state with time left as 6.



```
$ java CPU "ROUND ROBIN" 2
Time Quatam Q:2
Allocating Job with PID 1 into Process List
Allocating Job with PID 2 into Process List
Allocating Job with PID 3 into Process List
Allocating Job with PID 4 into Process List
Allocating Job with PID 5 into Process List
Jobs Ready:
Running Simulation

*************************
Fetching Job from Q
 with PID 1
Current State of the CPU:

List of processes:
PID: 1
PID: 2
PID: 3
PID: 4
PID: 5
*************************
Current Process Queue:
Scheduler using RoundRobinQueue Type Queue Structure
Time Quantum: 2
Printing Queue:
PID: 2 Desired CPU Time (ms): 2 State: NEW Time Left: 2
PID: 3 Desired CPU Time (ms): 8 State: NEW Time Left: 8
PID: 4 Desired CPU Time (ms): 5 State: NEW Time Left: 5
PID: 5 Desired CPU Time (ms): 10 State: NEW Time Left: 10
PID: 1 Desired CPU Time (ms): 6 State: HALTED Time Left: 6
*************************
Fetching Job from Q
 with PID 2
Current State of the CPU:
*************************
```

JAVA Packages used:

import java.io.*
import java.util.*
import java.net.*