Contents

**USCSP301: USCS303-Operating System (OS)**

**SJF ( with no pre-emption scheduling Algorithm…………………………………………………………………..….1**

**Aim……………………………………………………………………………………………………………………………………………2**

**Algorithm……………………………………………………………………………………………………………………………..…..3**

**Example of SJF……………………………………………………………………………………………………………………..……4**

**Gnatt Chart 5**

**Implementation………………………………………………………………………………………………………………………..6**

**Input…………………………………………………………………………………………………………………………………………7**

**Output………………………………………………………………………………………………………………………………………8**

**Sample Output………………………………………………………………………………………………………………………….9**

USCSP301\_USCS303\_Operating System(OS) Practical-02

Practical 02: Shortest Job First Scheduling Algorithm

Practical Date: 24th JULY ,2021.

### Practical aim: Implement SJF (with no preemption) scheduling algorithm in java

**Algorithm:**

CPU scheduling algorithm are used for scheduling different process present in the ready queue with available resource in an optimal way so that each and every process get execute by CPU.

Scheduling algorithm are broadly classified into two main type namely preemptive and non-preemptive.

FIRST COME FIRST OUT(FCFS) is also know as FIRST IN FIRST OUT (FIFO) SCHEDUAL algorithm is the and simplest

CPU .

A process scheduling different process to be assigned to the CPU based on particular scheduling algorithm . There

are six popular process scheduling algorithm which we are going to discuss in this chapter FIRST COME FIRST OUT(FCFS) scheduling.

**Example 1:** Consider the following example contain five processes .

|  |  |
| --- | --- |
| **Process Id** | **Burst Time** |
| P0 | 6 |
| P1 | 3 |
| P2 | 8 |
| P3 | 3 |
| P4 | 4 |

**Step 1:** Processes get execute according to their lowest burst time first .

|  |  |
| --- | --- |
| **Process Id** | **Burst Time** |
| P0 | 6 |
| P1 | 3 |
| P2 | 8 |
| P3 | 3 |
| P4 | 4 |

**Step 2: Following shows the scheduling and execution of processes Step 2.1:** At start P1 shortest execution time which is 0-3 second.

|  |  |
| --- | --- |
| System time | 0 |
| Processes scheduling finish time | P1 |
| Finish time | 0+3=3 |
| Waiting time | 3-3=0 |
| Turn Around time | 3-0=3 |

**Step 2.2:** next shortest execution time is for process P3 for duration 3-6 second.

|  |  |
| --- | --- |
| System time | 6 |
| Processes scheduling finish time | P1,p3 |
| Finish time | 3+3=6 |
| Waiting time | 6-3=3 |
| Turn Around time | 6-0=6 |

**Step 2.3:** Next job with shortest execution time is P4 for a duration 6-10 second.

|  |  |
| --- | --- |
| System time | 10 |
| Processes scheduling finish time | P1,p3’p4 |
| Finish time | 6+4=10 |
| Waiting time | 10-4=6 |
| Turn Around time | 10-0=10 |

**Step 2.4:** Next job with shortest execution time is p0 for duration of 10-16 second.

|  |  |
| --- | --- |
| System time | 10 |
| Processes scheduling finish time | P1,p3,p3,p4,p0 |
| Finish time | 10+6=16 |
| Waiting time | 16-6=10 |
| Turn Around time | 16-0=16 |

**Step 2.5 :** Similarly next job with shortest execution time is P2 for duration of 16-24 second.

|  |  |
| --- | --- |
| System Time | 16 |
| Process Scheduling Finish Time | P1,p3,P3,P4,P0,P2 |
| Finish Time | 16+8=24 |
| Waiting Time | 24-8=16 |
| Turn Around Time | 24-0=24 |

**Step 3:** Calculate average wating time and average turn around time. Average Waiting Time = (0+3+6+10+16)/5

= 35/5

=7

Average Turn Around Time =(3+6+10+16+24)/5

= 59/5

=11.8

**Gnatt Chart**

**Step 4:** After scheduling of all provided processes.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Process id** | **Burst time** | **Arrival time** | **Finish time** | **Turn Aroundtime** | **Wating time** |
| P1 | 3 | 0 | 0+3=3 | 3-0=3 | 3-3=0 |
| P3 | 3 | 0 | 3+3=6 | 6-0=6 | 6-3=3 |
| P4 | 4 | 0 | 6+4=10 | 10-0=10 | 10-4=6 |
| P0 | 6 | 0 | 10+6=16 | 16-0=16 | 16-6=10 |
| P2 | 8 | 0 | 16+8=24 | 24-0=24 | 24-8=16 |
| Average |  |  |  | 11.8000000 | 7.000000 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| P1 | P3 | P4 | P0 | P2 |

0 3 6 10 16 24

**Example 2:** Consider the following example containing five processes arrive at same time.

|  |  |
| --- | --- |
| **Processes ID** | **Burst Time** |
| P0 | 2 |
| P1 | 1 |
| P2 | 6 |

**Gnatt Chart**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Process id** | **Burst time** | **Arrival time** | **Finish time** | **Turn Around time** | **Wating time** |
| P1 | 1 | 0 | 1 | 1 | 1 |
| P0 | 2 | 0 | 3 | 3 | 3 |
| P2 | 6 | 0 | 9 | 9 | 9 |
| Average |  |  |  | 4.33333 | 1.33333 |
|  |  |  |  |  |  |

|  |  |  |
| --- | --- | --- |
| P1 | P0 | P2 |

0 1 3 9

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Example 3:** Consider the following example contain five processes arrive at same time .   |  |  | | --- | --- | | Process ID | Burst Time | | P0 | 25 | | P1 | 15 | | P2 | 10 | | P3 | 25 | | P4 | 10 | | P5 | 25 | | | | | | |
| **Process id** | **Burst time** | **Arrival time** | **Finish time** | **Turn Around time** | **Waiting time** |
| P2 | 10 | 0 | 10 | 10 | 0 |
| P4 | 10 | 0 | 20 | 20 | 10 |
| P1 | 15 | 0 | 35 | 35 | 20 |
| P0 | 25 | 0 | 60 | 60 | 35 |
| P3 | 25 | 0 | 85 | 85 | 60 |

**Gnatt chart:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Process id** | **Burst time** | **Arrival time** | **Finish time** | **Turn Around time** | **Waiting time** |
| P2 | 10 | 0 | 10 | 10 | 0 |
| P4 | 10 | 0 | 20 | 20 | 10 |
| P1 | 15 | 0 | 35 | 35 | 20 |
| P0 | 25 | 0 | 60 | 60 | 35 |
| P3 | 25 | 0 | 85 | 85 | 60 |
| P5 | 25 | 0 | 110 | 110 | 85 |
| Average |  |  |  | 53.3333 | 35.000000 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P2 | P4 | P1 | P0 | P3 | P5 |

**0 10 20 35 60 85 110**

**Example 4:** Consider the following example contain five processes arrive at same time .

|  |  |
| --- | --- |
| **Process Id** | **Burst Time** |
| P0 | 7 |
| P1 | 3 |
| P2 | 2 |
| P3 | 10 |
| P4 | 8 |

**Step 4:** After scheduling of all provided processes.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Process id** | **Burst time** | **Arrival time** | **Finish time** | **Turn Around time** | **Wating time** |
| P2 | 2 | 0 | 2 | 2 | 0 |
| P1 | 3 | 0 | 5 | 5 | 2 |
| P0 | 7 | 0 | 12 | 12 | 5 |
| P4 | 8 | 0 | 20 | 20 | 12 |
| P3 | 10 | 0 | 30 | 30 | 20 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Average |  |  |  | 13.80000 | 7.800000 |

Implementation:

// Name: Ritika Sahu

// Batch:B2

// PRN:2020016400783543

// Date:24th July,2021

// Prac-02:SJF (with no preemption) Algorithm

import java.util.Scanner;

public class P2\_SJF\_RS

{

// defining variables

int burstTime[];

int arrivalTime[]={0};

String[] processId;

int numberOfProcess;

void getProcessData(Scanner input) {

System.out.print("Enter the number of Process of Scheduling: ");

int inputNumberOfProcess=input.nextInt();

numberOfProcess=inputnumberOfProcess;

burstTime=new int[numberOfProcess];

arrivalTime=new int[numberOfProcess];

processId=new String[numberOfProcess];

String st="P";

for (int i=0;i<numberOfProcess;i++) {

processId[i]=st.concat(Integer.toString(i));

System.out.print("Enter the burst time for process-"+(i)+":");

burstTime[i]=input.nextInt(); }//for loop ends

}//getProcessData function ends

void sortAccordingBurstTime(int[]at,int[]bt,String[] pid) {

boolean swapped;

int temp;

String temp;

for(int i=0;i<numberOfProcess;i++){

swapped=false;

for(int j=0;j<numberOfProcess-i-1;j++){

if(bt[j]>bt[j+1]){

//swapping burst time

temp=bt[j];

bt[j]=bt[j+1];

bt[j+1]=temp;

//swapping arrival time

temp=at[j];

at[j]=at[j+1];

at[j+1]=temp;

//swapping process id

stemp=pid[j];

pid[j]=pid[j+1];

pid[j+1]=stemp;

//enchanced bubble sort swapped=true;

}//if ends

}//inner for(j) ends if (swapped==false){

break;

}

}//outer for(i) ends

}//sortAccordingBurstTime function ends

void shortestJobFirstNPAlgorithm(){

int finishTime[]=new int[numberOfProcess];

int bt[]=burstTime.clone();

int at[]=arrivalTime.clone();

String pid[]=processId.clone();

int waitingTime[]=new int[numberOfProcess];

int turnAroundTime[]=new int[numberOfProcess];

sortAccordingBurstTime(ab,bt,pid);

//calculating waiting & turn-around time for each process

finishTime[0]=at[0]+bt[0];

turnAroundTime[0]=finishTime[0]-at[0];

waitingTime[0]=turnAroundTime[0]-bt[0];

for (int i=1;i<numberOfProcess;i++){

finishTime[i]=bt[i]+finishTime[i-1];

turnAroundTime[i]=finishTime[i]-at[i];

waitingTime=turnAroundTime[i]-bt[i];

}

float sum=0;

for(int n:waitingTime){

sum+=n;

}

float averageWaitingTime=sum/numberOfProcess;

sum=0;

for (int n:turnAroundTime){

sum+=n;

}

float averageTurnAroundTime=sum/numberOfProcess;

//print on console the order of processes scheduled using

//Shortest Job First (with no preemption) Algorithm

Sytem.out.printIn("SJF (with no preemption) Scheduling Algorithm: ");

System.out.format("%20s%20s%20s%20s%20s%20s\n","ProcessId","BurstTime", "ArrivalTime", "FinishTime", "TurnAroundTime", "WaitingTime");

for (int i=0;i<numberOfProcess;i++){

System.out.format("%20s%20d%20d%20d%20d%20d\n", pid[i],bt[i],at[i],finidhTime[i],turnAroundTime[i],waitingTime[i]);

}

System.out.format("%80s%20f%20f\n", "Average",averageTurnAroundTime[i],AveragewaitingTime);

}//shortestJobFirstNPAlgorithm function ends

public static void main(String[] args){

Scanner input=new Scanner(System.in);

P2\_SJF\_RS obj=new P2\_SJF\_RS();

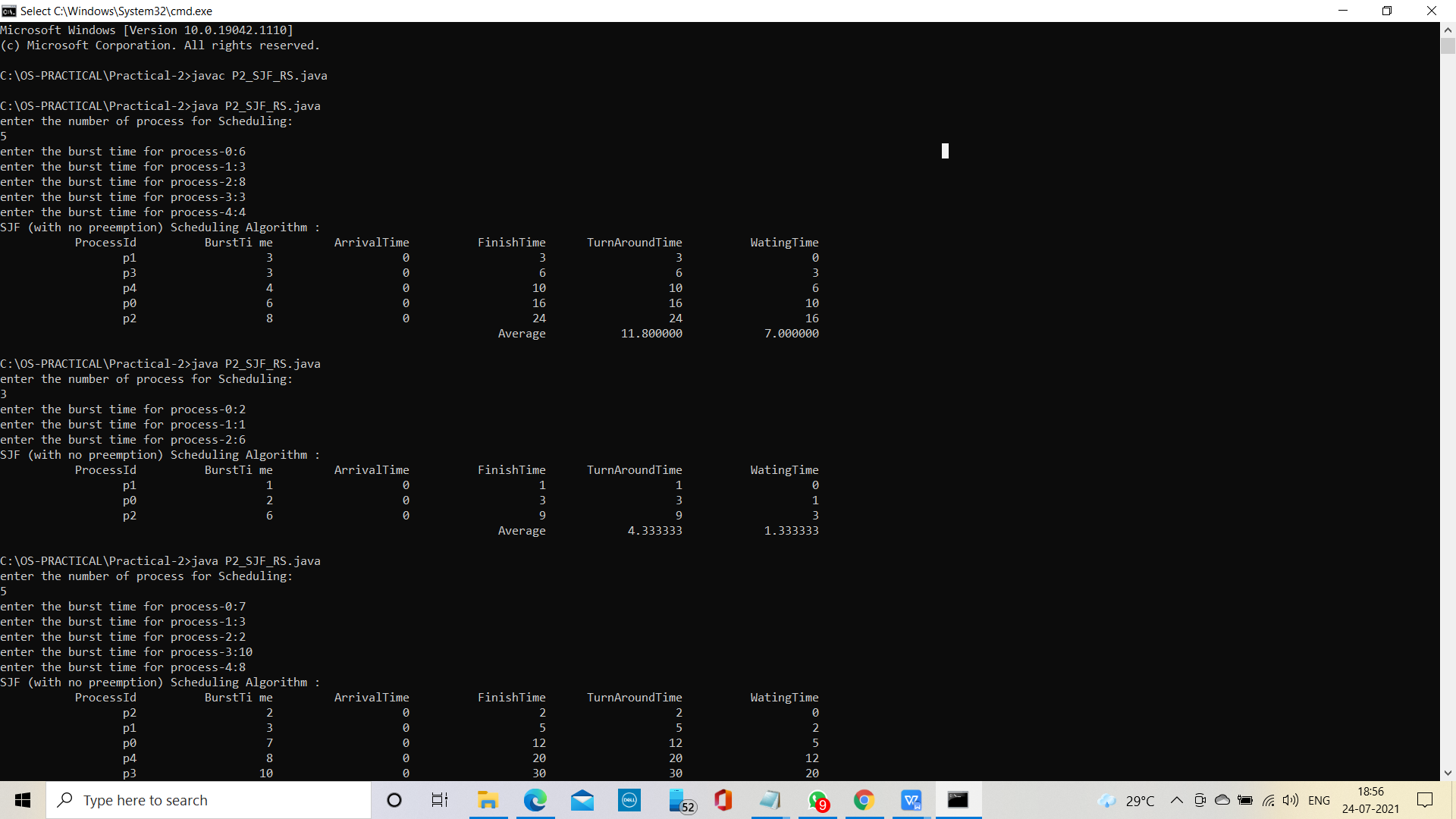
obj.getProcessData(input);

obj.shortestJobFirstNPAlgorithm();

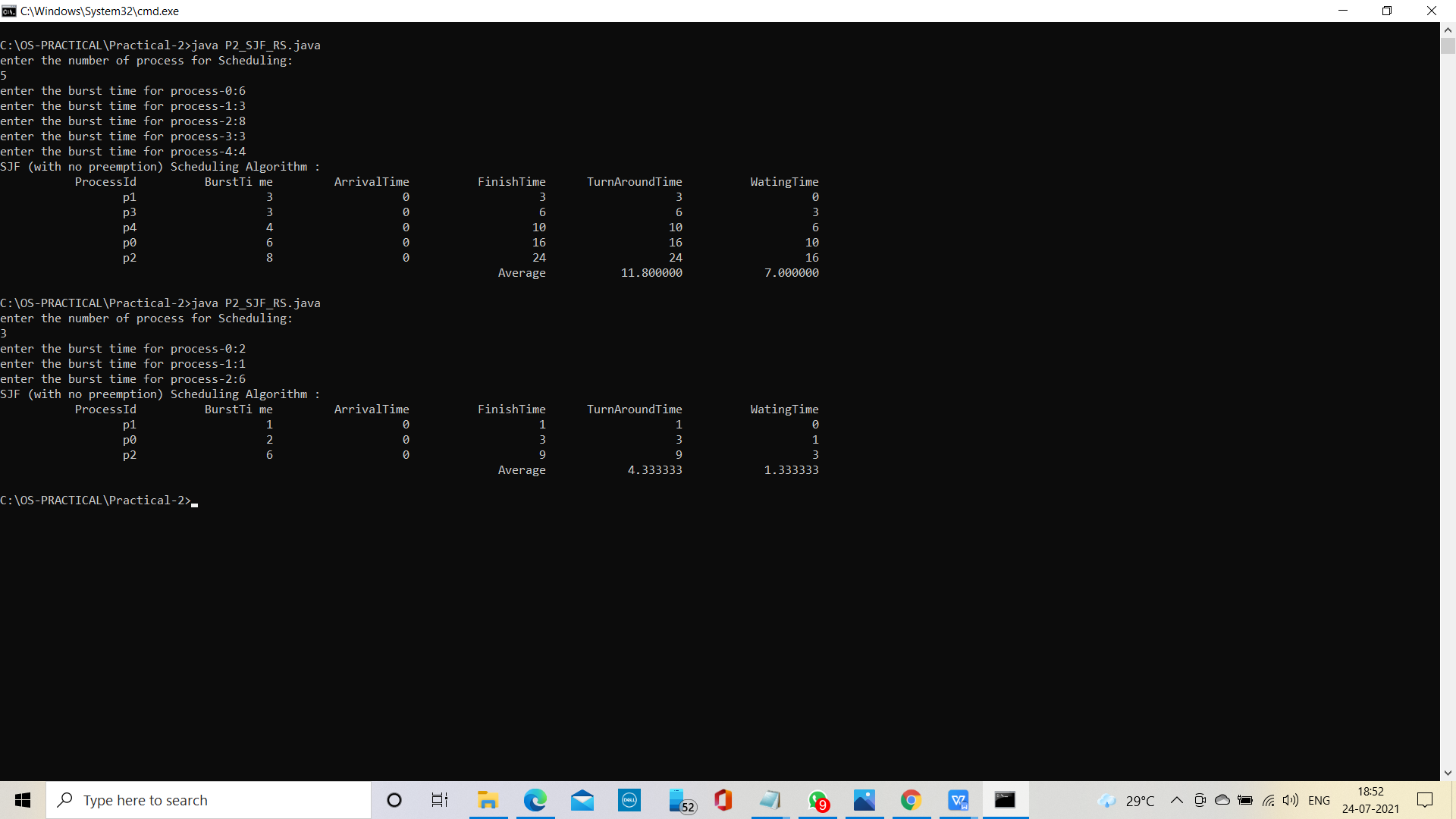
}//main ends

}//class ends P2\_SJF\_RS

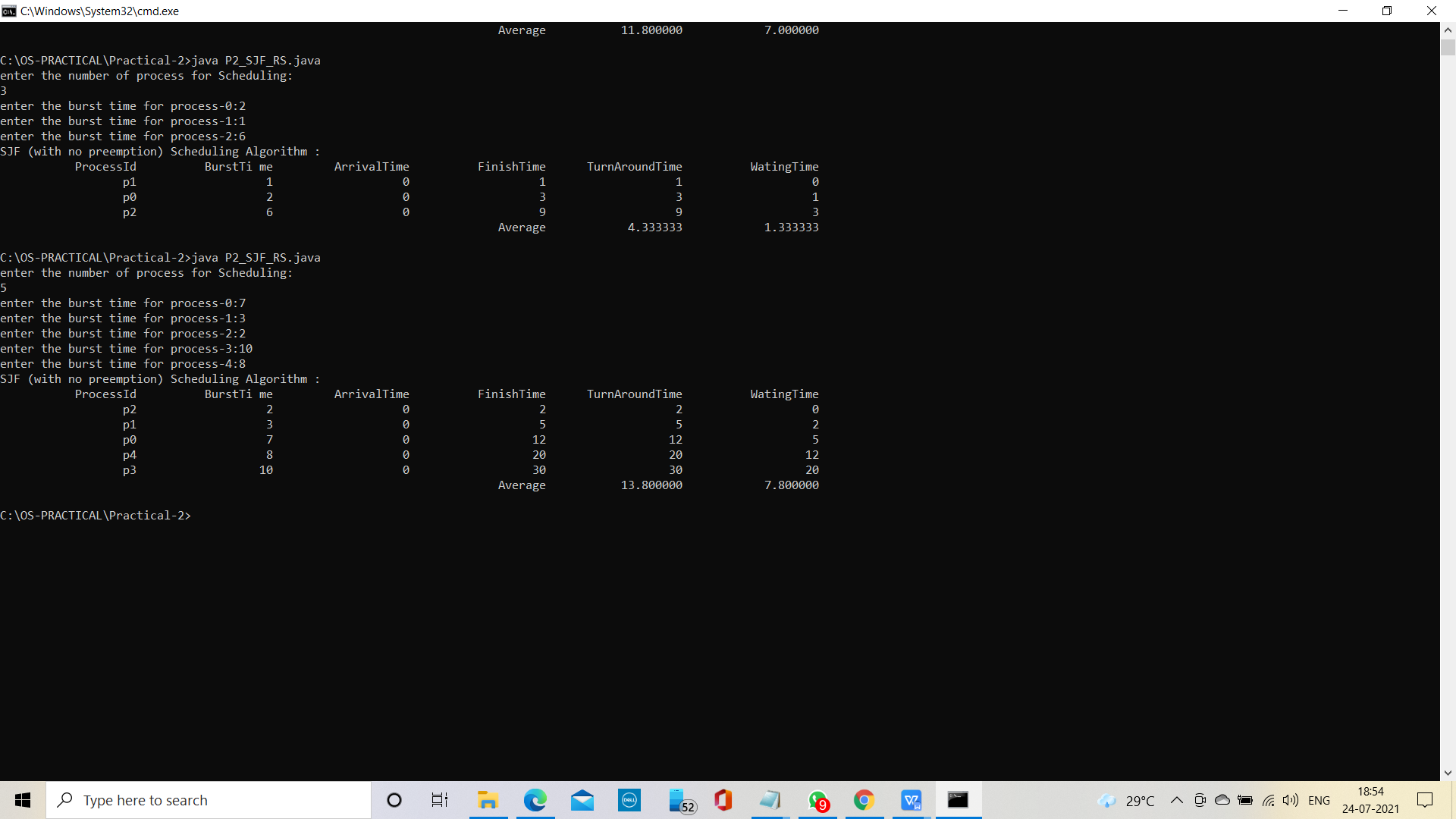
SAMPLE OUTPUT 1



SAMPLE OUTPUT 2:



SAMPLE OUTPUT 3:



SAMPLE OUTPUT 4:

