Shortest Job First (SJF) Scheduling

This method depends on the execution time of the process . In Ready Queue is chosen process that has less execution time (less burst time).

Two schemes:

A - non-preemptive

B – preemptive

A - non-preemptive.

In Ready Queue, the processes that have the least execution time are chosen, when more than one process have the same execution time, we depend on which comes first.

Example:

Processe id	Arrival time	Burst Time
P1	0	3
P2	0	1
P3	0	2

We have 3 processes in our ready queue. SJF will schedule the job which is having least execution time or burst time.

Gantt Chart

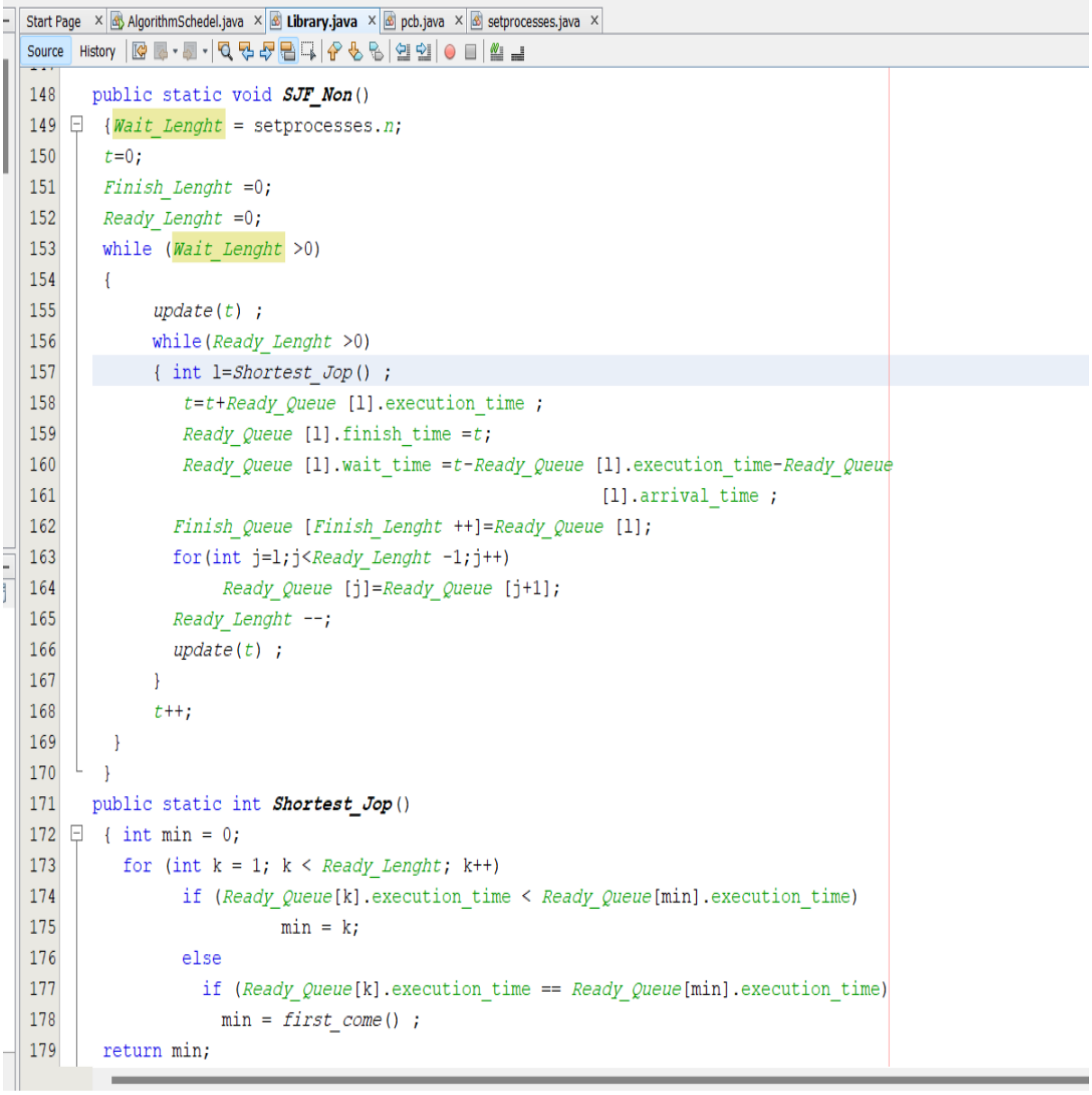
P2	P3	P1

0 3 6

1

we will calculate the Finish time and waiting time of each process.

Processe id	Finish time	Wait Time
P1	6	3
P2	1	0
P3	3	1



```
public static void SJF_Non()
 {Wait_Lenght = setprocesses.n;
 t=0;
 Finish_Lenght =0;
 Ready_Lenght =0;
 while (Wait_Lenght >0)
 {
      update(t) ;
      while(Ready_Lenght >0)
      { int l=Shortest_Jop() ;
         t=t+Ready_Queue [l].execution_time ;
         Ready_Queue [l].finish_time =t;
         Ready_Queue [l].wait_time =t-Ready_Queue [l].execution_time-Ready_Queue
                                           [l].arrival_time ;
        Finish_Queue [Finish_Lenght ++]=Ready_Queue [l];
        for(int j=l;j<Ready_Lenght -1;j++)
             Ready_Queue [j]=Ready_Queue [j+1];
        Ready_Lenght --;
        update(t) ;
      }
      t++;
  }
 }
public static int Shortest_Jop()
 { int min = 0;
   for (int k = 1; k < Ready_Lenght; k++)
         if (Ready_Queue[k].execution_time < Ready_Queue[min].execution_time)
                   min = k;
         else
           if (Ready_Queue[k].execution_time == Ready_Queue[min].execution_time)
             min = first_come() ;
 return min;
```

B-Preemptive

Among the processes in the ready queue , choose the process with the least execution time. If the CPU burst time (execution time) is equal, we choose the first come .

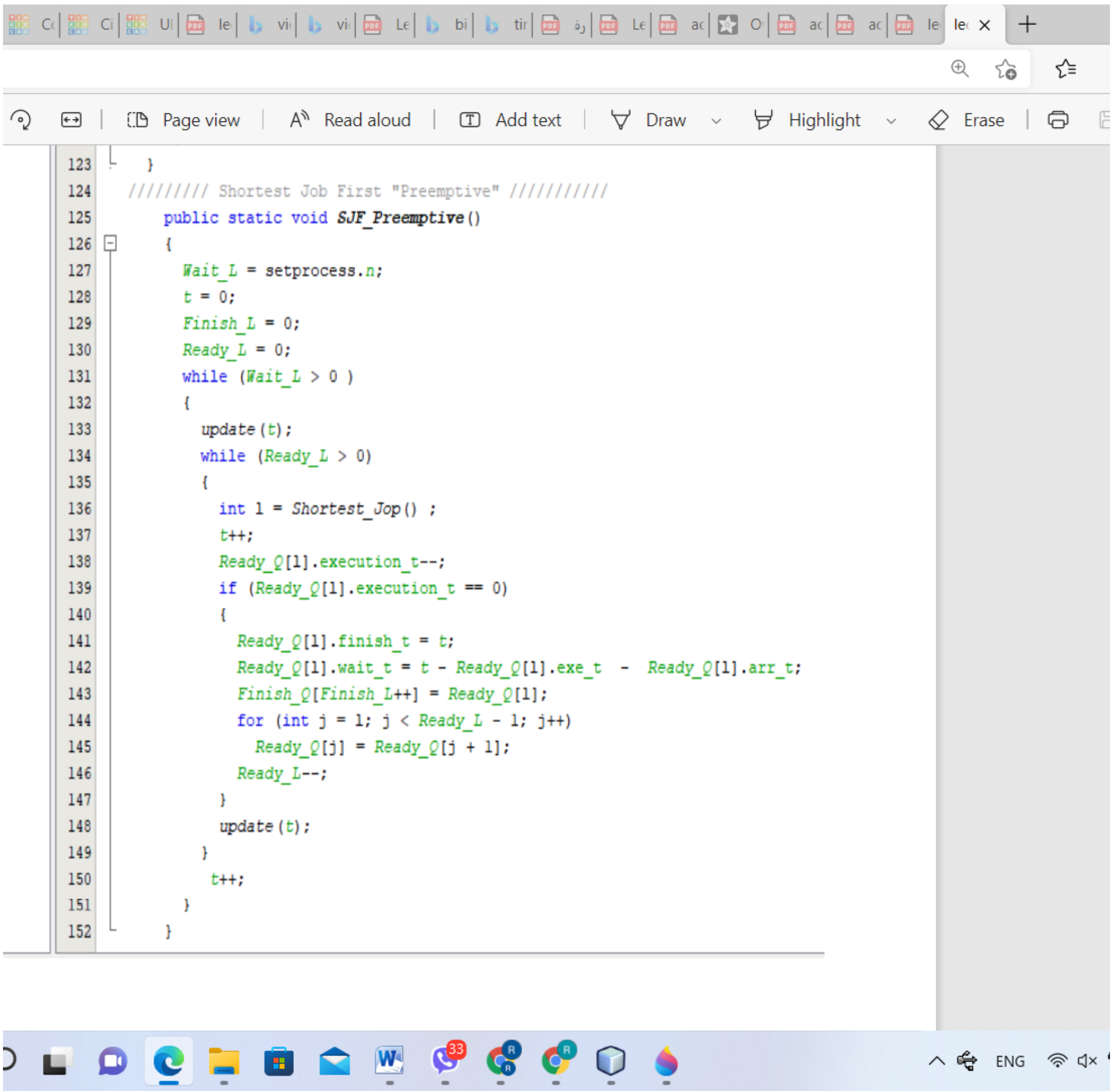
When a process reaches ready queue if the CPU burst time for the incoming process is less than the time remaining from the CPU burst time for the process running in the processor:

- Process executing in the processor will be stopped and returned this process to the ready queue with being updated the ready queue CPU burst time and ready queue arrive time the process.

- The newly arrived process will be entered into processer.

or else

- The newly arrived process will be entered into ready queue.



```
    }
//////// Shortest Job First "Preemptive" //////////
    public static void SJF_Preemptive()
    {
      Wait_L = setprocess.n;
      t = 0;
      Finish_L = 0;
      Ready_L = 0;
      while (Wait_L > 0 )
      {
        update(t);
        while (Ready_L > 0)
        {
          int l = Shortest_Jop() ;
          t++;
          Ready_Q[l].execution_t--;
          if (Ready_Q[l].execution_t == 0)
          {
            Ready_Q[l].finish_t = t;
            Ready_Q[l].wait_t = t - Ready_Q[l].exe_t  -  Ready_Q[l].arr_t;
            Finish_Q[Finish_L++] = Ready_Q[l];
            for (int j = l; j < Ready_L - 1; j++)
              Ready_Q[j] = Ready_Q[j + 1];
            Ready_L--;
          }
          update(t);
        }
         t++;
      }
    }
```