Assignment 3

Name: Disha Khater

Roll no: 64

GR no: 12010067

CS-A Batch 3

---

1. First come First serve (zeroth arrival time)

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.LinkedList;
import java.util.Scanner;

public class FCFS {

   public static void main(String[] args) {
      Scanner sc=new Scanner(System.in);
      System.out.println("enter number of processors");
      int n=sc.nextInt();
      System.out.println("Enter Burst time");
      int burst[]=new int[n];
      int waiting[]=new int[n];
      for(int i=0;i<n;i++){
         burst[i]=sc.nextInt();

      }

      //Zeroth Arrival Time

      waiting[0]=0;
      for(int i=1;i<n;i++){
         waiting[i]=waiting[i-1]+burst[i-1];

      }
      for(int i=0;i<n;i++){
         System.out.println("Waiting time P"+i+": "+waiting[i]);
      }
      int turnaround[]=new int[n];
      for(int i=0;i<n;i++){
         turnaround[i]=burst[i]+waiting[i];

      }
      for(int i=0;i<n;i++){
         System.out.println("Turn around time P"+i+": "+turnaround[i]);
      }
```

```
    }


}
```

```
enter number of processors
3
Enter Burst time
24
3
4
Waiting time P0: 0
Waiting time P1: 24
Waiting time P2: 27
Turn around time P0: 24
Turn around time P1: 27
Turn around time P2: 31

Process finished with exit code 0
```

2. First come first serve (different arrival time)

```
import java.util.Scanner;
//FIRST COME FIRST SERVE
public class FCFS_DIFF_ARRIVAL {

    public static void main(String[] args) {

        System.out.println("Enter the number of process");
        Scanner in = new Scanner(System.in);
        int numberOfProcesses = in.nextInt();

        int pid[] = new int[numberOfProcesses];
        int bt[] = new int[numberOfProcesses];
        int ar[] = new int[numberOfProcesses];
        int ct[] = new int[numberOfProcesses];
        int ta[] = new int[numberOfProcesses];
        int wt[] = new int[numberOfProcesses];

        for(int i = 0; i < numberOfProcesses; i++) {
            System.out.println("Enter process " + (i+1) + " arrival time: ");
            ar[i] = in.nextInt();
```

```
        System.out.println("Enter process " + (i+1) + " burst time: ");
        bt[i] = in.nextInt();
        pid[i] = i+1;
    }
    int temp;
    for (int i = 0; i < numberOfProcesses; i++) {
        for (int j = i+1; j < numberOfProcesses; j++) {

            if(ar[i] > ar[j]) {
                temp = ar[i];
                ar[i] = ar[j];
                ar[j] = temp;

                temp = pid[i];
                pid[i] = pid[j];
                pid[j] = temp;
                temp = bt[i];
                bt[i] = bt[j];
                bt[j] = temp;
            }
        }
    }

    System.out.println();
    ct[0] = bt[0] + ar[0];
    for(int i = 1; i < numberOfProcesses; i++) {
        ct[i] = ct[i - 1] + bt[i];
    }
    for(int i = 0; i < numberOfProcesses; i++) {
        ta[i] = ct[i] - ar[i];
        wt[i] = ta[i] - bt[i];
    }
    System.out.println("Process\t\tAT\t\tBT\t\tCT\t\tTAT\t\tWT");
    for(int i = 0; i < numberOfProcesses; i++) {
        System.out.println(pid[i]+"\t\t\t" + ar[i] + "\t\t" + bt[i]+ "\t\t" + ct[i]+ "\t\t" + ta[i]+ "\t\t" + wt[i]);
    }

    System.out.println("gantt chart: ");
    for(int i = 0; i < numberOfProcesses; i++) {
        System.out.print("P" + pid[i] +" ");
    }
  }

}
```

```
Enter the number of process
5
Enter process 1 arrival time:
0
Enter process 1 burst time:
10
Enter process 2 arrival time:
3
Enter process 2 burst time:
5
Enter process 3 arrival time:
5
Enter process 3 burst time:
2
Enter process 4 arrival time:
6
Enter process 4 burst time:
6
Enter process 5 arrival time:
8
Enter process 5 burst time:
4
```

```
Process     AT      BT      CT      TAT     WT
1           0       10      10      10      0
2           3       5       15      12      7
3           5       2       17      12      10
4           6       6       23      17      11
5           8       4       27      19      15
gantt chart:
P1 P2 P3 P4 P5
Process finished with exit code 0
```

3. Shortest Job First Scheduling (zeroth arrival time)

- Shortest Job First (SJF) is a Scheduling Algorithm where the process are executed in ascending order of their burst time, that is, the process having the shortest burst time is executed first and so on.

```
import java.util.Scanner;

public class SJF {
```

```
public static void main(String args[])
{
    Scanner sc = new Scanner(System.in);
    System.out.println ("enter no of process:");
    int n = sc.nextInt();
    int pid[] = new int[n];
    int at[] = new int[n];
    int bt[] = new int[n];
    int ct[] = new int[n];
    int ta[] = new int[n];
    int wt[] = new int[n];
    int f[] = new int[n];

    int st=0, tot=0;
    float avgwt=0, avgta=0;

    for(int i=0;i<n;i++)
    {
        System.out.println ("enter process " + (i+1) + " arrival time:");
        at[i] = sc.nextInt();
        System.out.println ("enter process " + (i+1) + " brust time:");
        bt[i] = sc.nextInt();
        pid[i] = i+1;
        f[i] = 0;
    }


    while(true)
    {
        int c=n, min = 999999;

        if (tot == n)
            break;

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

            if ((at[i] <= st) && (f[i] == 0) && (bt[i]<min))
            {
                min=bt[i];
                c=i;
            }
        }
        if (c==n)
            st++;
        else
        {
            ct[c]=st+bt[c];
            st+=bt[c];
            ta[c]=ct[c]-at[c];
            wt[c]=ta[c]-bt[c];
            f[c]=1;
            pid[tot] = c + 1;
            tot++;
        }
    }
```

```
    System.out.println("\npid  arrival brust  complete turn waiting");
    for(int i=0;i<n;i++)
    {
       avgwt+= wt[i];
       avgta+= ta[i];
       System.out.println(pid[i]+"\t\t"+at[i]+"\t\t"+bt[i]+"\t\t"+ct[i]+"\t\t"+ta[i]+"\t\t"+wt[i]);
    }
    System.out.println ("\naverage tat is "+ (float)(avgta/n));
    System.out.println ("average wt is "+ (float)(avgwt/n));
    sc.close();
    for(int i=0;i<n;i++)
    {
       System.out.print(pid[i] + " ");
    }
  }
}
```

```
enter no of process:
4
enter process 1 arrival time:
0
enter process 1 brust time:
2
enter process 2 arrival time:
0
enter process 2 brust time:
8
enter process 3 arrival time:
0
enter process 3 brust time:
1
enter process 4 arrival time:
0
enter process 4 brust time:
4
```

```
pid  arrival brust  complete turn waiting
3       0       2       3       3       1
1       0       8       15      15      7
4       0       1       1       1       0
2       0       4       7       7       3

average tat is 6.5
average wt is 2.75
3 1 4 2
Process finished with exit code 0
```

4. Shortest Job First Scheduling (Different arrival time)

```
import java.util.Scanner;

public class SJF {
    public static void main(String args[])
    {
        Scanner sc = new Scanner(System.in);
        System.out.println ("enter no of process:");
        int n = sc.nextInt();
        int pid[] = new int[n];
        int at[] = new int[n];
        int bt[] = new int[n];
        int ct[] = new int[n];
        int ta[] = new int[n];
        int wt[] = new int[n];
        int f[] = new int[n];

        int st=0, tot=0;
        float avgwt=0, avgta=0;

        for(int i=0;i<n;i++)
        {
            System.out.println ("enter process " + (i+1) + " arrival time:");
            at[i] = sc.nextInt();
            System.out.println ("enter process " + (i+1) + " brust time:");
            bt[i] = sc.nextInt();
            pid[i] = i+1;
            f[i] = 0;
        }


        while(true)
        {
            int c=n, min = 999999;

            if (tot == n)
                break;

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

                if ((at[i] <= st) && (f[i] == 0) && (bt[i]<min))
                {
                    min=bt[i];
                    c=i;
                }
            }
            if (c==n)
                st++;
            else
            {
                ct[c]=st+bt[c];
                st+=bt[c];
                ta[c]=ct[c]-at[c];
                wt[c]=ta[c]-bt[c];
                f[c]=1;
                pid[tot] = c + 1;
                tot++;
```

```
            }
        }

        System.out.println("\npid  arrival brust  complete turn waiting");
        for(int i=0;i<n;i++)
        {
            avgwt+= wt[i];
            avgta+= ta[i];

System.out.println(pid[i]+"\t\t"+at[i]+"\t\t"+bt[i]+"\t\t"+ct[i]+"\t\t"+ta[i]
+"\t\t"+wt[i]);
        }
        System.out.println ("\naverage tat is "+ (float)(avgta/n));
        System.out.println ("average wt is "+ (float)(avgwt/n));
        sc.close();
        for(int i=0;i<n;i++)
        {
            System.out.print(pid[i] + " ");
        }
    }
}
```

```
enter no of process:
4
enter process 1 arrival time:
1
enter process 1 brust time:
3
enter process 2 arrival time:
2
enter process 2 brust time:
4
enter process 3 arrival time:
1
enter process 3 brust time:
2
enter process 4 arrival time:
4
enter process 4 brust time:
4
```

```
pid  arrival brust  complete turn waiting
3       1       3       6       5       2
1       2       4       10      8       4
2       1       2       3       2       0
4       4       4       14      10      6

average tat is 6.25
average wt is 3.0
3 1 2 4
Process finished with exit code 0
```