1. FCFS

```
import java.util.*;

public class FCFS {
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];   // process ids
int ar[] = new int[n];    // arrival times
int bt[] = new int[n];    // burst or execution times
int ct[] = new int[n];    // completion times
int ta[] = new int[n];    // turn around times
int wt[] = new int[n];    // waiting times
int temp;
float avgwt=0,avgta=0;

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

//sorting according to arrival times
for(int i = 0 ; i <n; i++)
{
for(int  j=0;  j < n-(i+1) ; j++)
{
if( ar[j] > ar[j+1] )
{
temp = ar[j];
ar[j] = ar[j+1];
ar[j+1] = temp;
temp = bt[j];
bt[j] = bt[j+1];
bt[j+1] = temp;
temp = pid[j];
pid[j] = pid[j+1];
pid[j+1] = temp;
```

```
}
}
}
// finding completion times
for(int  i = 0 ; i < n; i++)
{
if( i == 0)
{
ct[i] = ar[i] + bt[i];
}
else
{
if( ar[i] > ct[i-1])
{
ct[i] = ar[i] + bt[i];
}
else
ct[i] = ct[i-1] + bt[i];
}
ta[i] = ct[i] - ar[i] ;         // turnaround time= completion time- arrival time
wt[i] = ta[i] - bt[i] ;          // waiting time= turnaround time- burst time
avgwt += wt[i] ;                 // total waiting time
avgta += ta[i] ;                // total turnaround time
}
System.out.println("\npid  arrival  burst  complete turn waiting");
for(int  i = 0 ; i< n;  i++)
{
System.out.println(pid[i] + "  \t " + ar[i] + "\t" + bt[i] + "\t" + ct[i] + "\t" + ta[i] + "\t"  + wt[i] ) ;
}
sc.close();
System.out.println("\naverage waiting time: "+ (avgwt/n));     // printing average waiting time.
System.out.println("average turnaround time:"+(avgta/n));    // printing average turnaround
time.
}
}
```

2. SJF Non preemptive

```
import java.util.*;

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]; // at means arrival time
int bt[] = new int[n]; // bt means burst time
int ct[] = new int[n]; // ct means complete time
int ta[] = new int[n]; // ta means turn around time
int wt[] = new int[n];  //wt means waiting time
int f[] = new int[n];  // f means it is flag it checks process is completed or not
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) + " burst time:");
bt[i] = sc.nextInt();
pid[i] = i+1;
f[i] = 0;
}
boolean a = true;
while(true)
{
int c=n, min=999;
if (tot == n) // total no of process = completed process loop will be terminated
break;
for (int i=0; i<n; i++)
{
/*
* If i'th process arrival time <= system time and its flag=0 and burst<min
* That process will be executed first
*/
if ((at[i] <= st) && (f[i] == 0) && (bt[i]<min))
{
min=bt[i];
c=i;
}
}
/* If c==n means c value can not updated because no process arrival time< system time so we
increase the system time */
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;
tot++;
}
}
System.out.println("\npid  arrival burst  complete turn waiting");
for(int i=0;i<n;i++)
{
avgwt+= wt[i];
avgta+= ta[i];
System.out.println(pid[i]+"\t"+at[i]+"\t"+bt[i]+"\t"+ct[i]+"\t"+ta[i]+"\t"+wt[i]);
}
System.out.println ("\naverage tat is "+ (float)(avgta/n));
System.out.println ("average wt is "+ (float)(avgwt/n));
sc.close();
}
}
```

3. Java Program for Shortest Job First (SRTF) Scheduling (Preemptive)

```
import java.util.*;

public class SRTF {
        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]; // it takes pid of process
                int at[] = new int[n]; // at means arrival time
                int bt[] = new int[n]; // bt means burst time
                int ct[] = new int[n]; // ct means complete time
                int ta[] = new int[n];// ta means turn around time
                int wt[] = new int[n];  // wt means waiting time
                int f[] = new int[n];  // f means it is flag it checks process is completed or not
                int k[]= new int[n];   // it is also stores burst time
           int i, st=0, tot=0;
           float avgwt=0, avgta=0;

           for (i=0;i<n;i++)
```

```
{
	pid[i]= i+1;
	System.out.println ("enter process " +(i+1)+ " arrival time:");
	at[i]= sc.nextInt();
	System.out.println("enter process " +(i+1)+ " burst time:");
	bt[i]= sc.nextInt();
	k[i]= bt[i];
	f[i]= 0;
}

while(true){
	int min=99,c=n;
	if (tot==n)
		break;

	for ( 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
	{
		bt[c]--;
		st++;
		if (bt[c]==0)
		{
			ct[c]= st;
			f[c]=1;
			tot++;
		}
	}
}

for(i=0;i<n;i++)
{
	ta[i] = ct[i] - at[i];
	wt[i] = ta[i] - k[i];
	avgwt+= wt[i];
```

```
                avgta+= ta[i];
            }

            System.out.println("pid  arrival  burst  complete turn waiting");
            for(i=0;i<n;i++)
            {
                System.out.println(pid[i] +"\t"+ at[i]+"\t"+ k[i] +"\t"+ ct[i] +"\t"+ ta[i] +"\t"+
wt[i]);
            }

            System.out.println("\naverage tat is "+ (float)(avgta/n));
            System.out.println("average wt is "+ (float)(avgwt/n));
            sc.close();
        }
}
```

4. RR

```
import java.util.Scanner;
public class RoundRobin
{
public static void main(String args[])
{
int n,i,qt,count=0,temp,sq=0,bt[],wt[],tat[],rem_bt[];
float awt=0,atat=0;
bt = new int[10];
wt = new int[10];
tat = new int[10];
rem_bt = new int[10];
Scanner s=new Scanner(System.in);
System.out.print("Enter the number of process (maximum 10) = ");
n = s.nextInt();
System.out.print("Enter the burst time of the process\n");
for (i=0;i<n;i++)
{
System.out.print("P"+i+" = ");
bt[i] = s.nextInt();
rem_bt[i] = bt[i];
}
System.out.print("Enter the quantum time: ");
qt = s.nextInt();
while(true)
{
for (i=0,count=0;i<n;i++)
```

```
{
temp = qt;
if(rem_bt[i] == 0)
{
count++;
continue;
}
if(rem_bt[i]>qt)
rem_bt[i]= rem_bt[i] - qt;
else
if(rem_bt[i]>=0)
{
temp = rem_bt[i];
rem_bt[i] = 0;
}
sq = sq + temp;
tat[i] = sq;
}
if(n == count)
break;
}
System.out.print("---------------------------------------------------------------------------------");
System.out.print("\nProcess\t     Burst Time\t      Turnaround Time\t         Waiting Time\n");
System.out.print("---------------------------------------------------------------------------------");
for(i=0;i<n;i++)
{
wt[i]=tat[i]-bt[i];
awt=awt+wt[i];
atat=atat+tat[i];
System.out.print("\n "+(i+1)+"\t "+bt[i]+"\t\t "+tat[i]+"\t\t "+wt[i]+"\n");
}
awt=awt/n;
atat=atat/n;
System.out.println("\nAverage waiting Time = "+awt+"\n");
System.out.println("Average turnaround time = "+atat);
}
}
```

5. Priority

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

public class Priority {
```

```
    public static void main(String[] args) {

        System.out.println("*** Priority Scheduling ***");

        System.out.print("Enter Number of Process: ");
        Scanner sc = new Scanner(System.in);
        int numberOfProcess = sc.nextInt();
        String process[] = new String[numberOfProcess];

        int p = 1;
        for (int i = 0; i < numberOfProcess; i++) {
            process[i] = "P" + p;
            p++;
        }

        System.out.println(Arrays.toString(process));

        System.out.print("Enter Burst Time for " + numberOfProcess + " process: ");

        int burstTime[] = new int[numberOfProcess];
        for (int i = 0; i < numberOfProcess; i++) {
            burstTime[i] = sc.nextInt();
        }

        System.out.println(Arrays.toString(burstTime));

        System.out.print("Enter Priority for " + numberOfProcess + " process: ");

        int priority[] = new int[numberOfProcess];
        for (int i = 0; i < numberOfProcess; i++) {
            priority[i] = sc.nextInt();
        }

        System.out.println(Arrays.toString(priority));

// Sorting process & burst time by priority
int temp;
String temp2;
for (int i = 0; i < numberOfProcess - 1; i++) {
    for (int j = 0; j < numberOfProcess - 1; j++) {
        if (priority[j] > priority[j + 1]) {
            temp = priority[j];
             priority[j] = priority[j + 1];
```

```
                priority[j + 1] = temp;

                temp = burstTime[j];
                burstTime[j] = burstTime[j + 1];
                burstTime[j + 1] = temp;

                temp2 = process[j];
                process[j] = process[j + 1];
                process[j + 1] = temp2;

            }
          }
        }

        int TAT[] = new int[numberOfProcess + 1];
        int waitingTime[] = new int[numberOfProcess + 1];

    // Calculating Waiting Time & Turn Around Time
        for (int i = 0; i < numberOfProcess; i++) {
          TAT[i] = burstTime[i] + waitingTime[i];
          waitingTime[i + 1] = TAT[i];
        }

        int totalWT = 0;
        int totalTAT = 0;
        double avgWT;
        double avgTAT;

        System.out.println("Process    BT     WT      TAT");
        for (int i = 0; i < numberOfProcess; i++) {

          System.out.println(process[i] + "         " + burstTime[i] + "      " + waitingTime[i] + "        " +
    (TAT[i]));
          totalTAT += (waitingTime[i] + burstTime[i]);
          totalWT += waitingTime[i];

        }

        avgWT = totalWT / (double) numberOfProcess;
        avgTAT = totalTAT / (double) numberOfProcess;

        System.out.println("\n Average Wating Time: " + avgWT);
        System.out.println(" Average Turn Around Time: " + avgTAT);
```

```
    }

}
```