**Write a java program to implement round robin scheduling algorithm**

**package**Thuresday;

**import**java.util.Scanner;

**publicclass** GFG

{

// Method to find the waiting time for all

// processes

**staticvoid**findWaitingTime(**int**processes[], **int**n,

**int**bt[], **int**wt[], **int**quantum)

{

// Make a copy of burst times bt[] to store remaining

// burst times.

**int**rem\_bt[] = **newint**[n];

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

rem\_bt[i] = bt[i];

**int**t = 0; // Current time

// Keep traversing processes in round robin manner

// until all of them are not done.

**while**(**true**)

{

**boolean**done = **true**;

// Traverse all processes one by one repeatedly

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

{

// If burst time of a process is greater than 0

// then only need to process further

**if** (rem\_bt[i] > 0)

{

done = **false**; // There is a pending process

**if** (rem\_bt[i] >quantum)

{

// Increase the value of t i.e. shows

// how much time a process has been processed

t += quantum;

// Decrease the burst\_time of current process

// by quantum

rem\_bt[i] -= quantum;

}

// If burst time is smaller than or equal to

// quantum. Last cycle for this process

**else**

{

// Increase the value of t i.e. shows

// how much time a process has been processed

t = t + rem\_bt[i];

// Waiting time is current time minus time

// used by this process

wt[i] = t - bt[i];

// As the process gets fully executed

// make its remaining burst time = 0

rem\_bt[i] = 0;

}

}

}

// If all processes are done

**if** (done == **true**)

**break**;

}

}

// Method to calculate turn around time

**staticvoid**findTurnAroundTime(**int**processes[], **int**n,

**int**bt[], **int**wt[], **int**tat[])

{

// calculating turnaround time by adding

// bt[i] + wt[i]

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

tat[i] = bt[i] + wt[i];}

// Method to calculate average time

**staticvoid**findavgTime(**int**processes[], **int**n, **int**bt[],

**int**quantum)

{

**int**wt[] = **newint**[n], tat[] = **newint**[n];

**int**total\_wt = 0, total\_tat = 0;

// Function to find waiting time of all processes

*findWaitingTime*(processes, n, bt, wt, quantum);

// Function to find turn around time for all processes

*findTurnAroundTime*(processes, n, bt, wt, tat);

// Display processes along with all details

System.***out***.println("Processes " + " Burst time " +

" Waiting time " + " Turn around time");

// Calculate total waiting time and total turn

// around time

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

{

total\_wt = total\_wt + wt[i];

total\_tat = total\_tat + tat[i];

System.***out***.println(" " + (i+1) + "\t\t" + bt[i] +"\t " +

wt[i] +"\t\t " + tat[i]);

}

System.***out***.println("Average waiting time = " +

(**float**)total\_wt / (**float**)n);

System.***out***.println("Average turn around time = " +

(**float**)total\_tat / (**float**)n);

}

// Driver Method

**publicstaticvoid** main(String[] args)

{

// process id's

Scanner s = **new** Scanner(System.***in***);

System.***out***.print("Enter no. of processes:");

**int**n = s.nextInt();

**int**processes[] = **newint**[n];

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

{

processes[i] = i;

}

**int**burst\_time[]= **newint**[n];

System.***out***.print("Enter burst time:");

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

{

burst\_time[i] = s.nextInt();

}

// Time quantum

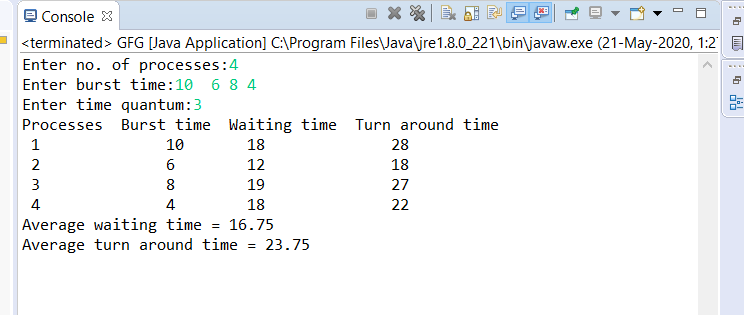
System.***out***.print("Enter time quantum:");

**int**quantum = s.nextInt();

*findavgTime*(processes, n, burst\_time, quantum);

}

}

**OUTPUT**