**ALGORITHM 2: Write a Program to demonstrate the use of SJF CPU Scheduling algorithm in java.**

import java.util.\*;

public class SJF

{

public static void main(String[] args)

{

Scanner sc = new Scanner(System.in);

System.out.print("Enter number of processes: ");

int n = sc.nextInt();

int[] burstTime = new int[n];

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

{

System.out.print("Enter burst time for process " + (i+1) + ": ");

burstTime[i] = sc.nextInt();

}

List<Process> processes = new ArrayList<>();

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

{

processes.add(new Process(i+1, burstTime[i]));

}

Collections.sort(processes, new Comparator<Process>()

{

@Override

public int compare(Process p1, Process p2)

{

if (p1.getBurstTime() == p2.getBurstTime())

{

return Integer.compare(p1.getId(), p2.getId());

}

return Integer.compare(p1.getBurstTime(), p2.getBurstTime());

}

});

int[] waitingTime = new int[n];

int[] turnaroundTime = new int[n];

int time = 0; // the current time

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

{

Process p = processes.get(i);

waitingTime[p.getId()-1] = time;

time += p.getBurstTime();

turnaroundTime[p.getId()-1] = time;

}

System.out.println("\nProcess\tBurst Time\tWaiting Time\tTurnaround Time");

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

{

Process p = processes.get(i);

System.out.println(p.getId() + "\t" + p.getBurstTime() + "\t\t" + waitingTime[p.getId()-1] + "\t\t" + turnaroundTime[p.getId()-1]);

}

}

static class Process

{

private int id;

private int burstTime;

public Process(int id, int burstTime)

{

this.id = id;

this.burstTime = burstTime;

}

public int getId()

{

return id;

}

public int getBurstTime()

{

return burstTime;

}

}

}

OUTPUT :-

