```
/**
 * A class that simulates priority algo.
 */

import org.omg.PortableInterceptor.INACTIVE;

import javax.sound.midi.Soundbank;
import java.io.*;
import java.util.*;

public class pnp {

    private Map<Integer, Integer> mpp1 = new HashMap<>(
);
    private Map<Integer, Integer> mpp2 = new HashMap<>(
);
    private Map<Integer, Integer> mpp6 = new HashMap<>(
);

    private String algo;

    /**
     * Constructor
     * @param mpp1 pid-->timestamp
     * @param mpp2 pid-->cpuburst
     * @param mpp6 pid-->priority
     * @param algo name of the algorithm
     */
    public pnp(Map<Integer, Integer> mpp1, Map<Integer,
 Integer> mpp2,
                Map<Integer, Integer> mpp6,  String
algo)
    {
        this.mpp1.putAll(mpp1);
        this.mpp2.putAll(mpp2);
        this.mpp6.putAll(mpp6);
        this.algo = algo;



    }
    /**
     * A method to output required tables in file
     */


    public void fileString()
    {
        try
        {
            FileWriter fileWriter = new FileWriter("
pnp_out.txt");
            PrintWriter printwr = new PrintWriter(
fileWriter);
            Map<Integer, Integer> mpp3 = new HashMap<>(
);
            Map<Integer, Integer> mpp4 = new HashMap<>(
);

            int clock = 0;
            printwr.println("CPU sheduling algorithm: "
+algo);
            printwr.println("Total number of CPU
requests: "+mpp1.size());
            printwr.println(
"-----------------------------------------------------
---");
            String mi = "";
            Iterator<Integer> iter = mpp6.keySet().
iterator();
            int coun = 0;
            int cou = 0;
            int mini = 0;


            while(iter.hasNext()) {

                int itemd = iter.next();

                for (int i = 0; i < mpp6.size(); i++) {
                    if (mpp6.get(i) <= mpp6.get(mini))
{
                        if (mpp6.get(i) == mpp6.get(
mini)) {
                            if (i < mini) {
                                mini = i;
                            }
                        } else {
                            mini = i;
                        }
                    }


                }


                mpp4.put(cou, mpp2.get(mini));
                cou++;


                printwr.println("Clock: " + clock);
                printwr.println("Pending CPU request(s
):");
                printwr.println(mini + " " + mpp1.get(
mini) + " " + mpp2.get(mini)
                        +" "+mpp6.get(mini));

                clock += mpp2.get(mini);

                mpp3.put(coun, clock);
                coun++;


                mi = mini + " " + mpp1.get(mini) + " "
+ mpp2.get(mini) +" "+mpp6.get(mini);
                mpp1.put(mini, Integer.MAX_VALUE);
                mpp2.put(mini, Integer.MAX_VALUE);
                mpp6.put(mini, Integer.MAX_VALUE);

                Iterator<Integer> iter1 = mpp1.keySet()
.iterator();
```

```
                while (iter1.hasNext()) {

                    int item2 = iter1.next();
                    if (mpp1.get(item2) < Integer.
MAX_VALUE) {
                        printwr.println(item2 + " " +
mpp1.get(item2) + " " + mpp2.get(item2)
                                +" "+mpp6.get(item2));
                    }
                }
                printwr.println();
                printwr.println("CPU Request serviced
during this clock interval: " + mi);
                printwr.println(
"-------------------------------------------------------
---");
            }




            printwr.println("Turn-Around Time
Computations");
            printwr.println();
            int count = 0;

            double sum = 0;
            for(int i = 0;i<mpp1.size();i++)
            {
                printwr.println("TAT("+i+") = "+mpp3.
get(i));
                sum+=mpp3.get(i);
                count++;
            }

            printwr.println();
            printwr.println("Average TAT = "+(sum/count
));
            printwr.println(
"-------------------------------------------------------
```

```
---");

            printwr.println("Wait Time Computations");
            printwr.println();

            int count1 = 0;
            double sum1 = 0;

            for(int i = 0;i<mpp1.size();i++)
            {
                printwr.println("WT("+i+") = "+(mpp3.
get(i)-mpp4.get(i)));
                sum1+=(mpp3.get(i)-mpp4.get(i));
                count1++;
            }

            printwr.println();
            printwr.println("Average WT = "+(sum1/
count1));

            printwr.close();



        }
        catch (Exception e)
        {
            System.out.println("File can't be OPENED/
READ");
        }

    }




}
```

```java
/**
 * Class for simulating First come first serve
 */

import org.omg.PortableInterceptor.INACTIVE;

import javax.sound.midi.Soundbank;
import java.io.*;
import java.util.*;

public class Fcfs {

    private Map<Integer, Integer> mpp1 = new HashMap<>(
);
    private Map<Integer, Integer> mpp2 = new HashMap<>(
);

    private String algo;

    /**
     * Constructor
     * @param mpp1 pid-->timestamp
     * @param mpp2 pid-->cpuburst
     * @param algo name of algorithm
     */
    public Fcfs(Map<Integer, Integer> mpp1, Map<Integer
, Integer> mpp2, String algo)
    {
       this.mpp1.putAll(mpp1);
        this.mpp2.putAll(mpp2);
        this.algo = algo;




    }

    /**
     * A method to output required tables in file
     */
    public void fileString()
    {
        try
        {
            FileWriter fileWriter = new FileWriter("
fcfs_out.txt");
            PrintWriter printwr = new PrintWriter(
fileWriter);
            Map<Integer, Integer> mpp3 = new HashMap<>(
);
            Map<Integer, Integer> mpp4 = new HashMap<>(
);

            int clock = 0;
            printwr.println("CPU sheduling algorithm: "
+algo);
            printwr.println("Total number of CPU
requests: "+mpp1.size());
            printwr.println(
"------------------------------------------------------
---");
            String mi = "";
            Iterator<Integer> iter = mpp1.keySet().
iterator();
            int coun = 0;
            int cou = 0;
            int mini = 0;

            while(iter.hasNext())
            {


                int itemd = iter.next();

                for (int i = 0; i < mpp1.size(); i++) {

                    if (mpp1.get(i) <= mpp1.get(mini))
{
                        if (mpp1.get(i) == mpp1.get(
mini)) {
                            if (i < mini) {
                                mini = i;
                            }
                        } else {
                            mini = i;
                        }
                    }



                }

                mpp4.put(cou,mpp2.get(mini));
                cou++;

                printwr.println("Clock: "+clock);
                printwr.println("Pending CPU request(s
):");
                printwr.println(mini+" "+mpp1.get(mini)
+" "+mpp2.get(mini));

                clock+=mpp2.get(mini);

                mpp3.put(coun, clock);
                coun++;



                        mi = mini+" "+mpp1.get(mini)+
" "+mpp2.get(mini);
                mpp1.put(mini,Integer.MAX_VALUE);
                mpp2.put(mini,Integer.MAX_VALUE);

                Iterator<Integer> iter1 = mpp1.keySet()
.iterator();


                while(iter1.hasNext())
                {

                    int item2 = iter1.next();
                    if(mpp1.get(item2)<Integer.
MAX_VALUE)
                    {
                        printwr.println(item2 + " " +
mpp1.get(item2) + " " + mpp2.get(item2));
                    }
                }
                printwr.println();
                printwr.println("CPU Request serviced
during this clock interval: "+mi);
                printwr.println(
"-------------------------------------------------------
---");




            }

            printwr.println("Turn-Around Time
Computations");
            printwr.println();
            int count = 0;
            double sum = 0;
            for(int i = 0;i<mpp1.size();i++)
            {
                printwr.println("TAT("+i+") = "+mpp3.
get(i));
                sum+=mpp3.get(i);
                count++;
            }

            printwr.println();
            printwr.println("Average TAT = "+(sum/count
));
            printwr.println(
"-------------------------------------------------------
---");

            printwr.println("Wait Time Computations");
            printwr.println();

            int count1 = 0;
            double sum1 = 0;

            for(int i = 0;i<mpp1.size();i++)
            {
                printwr.println("WT("+i+") = "+(mpp3.
get(i)-mpp4.get(i)));
                sum1+=(mpp3.get(i)-mpp4.get(i));
                count1++;
            }

            printwr.println();
            printwr.println("Average WT = "+(sum1/
count1));

            printwr.close();


        }
        catch (Exception e)
        {
            System.out.println("File can't be OPENED/
READ");
        }

    }






}
```

```java
/**
 * A class that simulates shortest job next
 */

import org.omg.PortableInterceptor.INACTIVE;

import javax.sound.midi.Soundbank;
import java.io.*;
import java.util.*;

public class Sjnp {

    private Map<Integer, Integer> mpp1 = new HashMap<>(
);
    private Map<Integer, Integer> mpp2 = new HashMap<>(
);

    private String algo;

    /**
     * Constructor
     * @param mpp1 pid-->timestamp
     * @param mpp2 pid-->cpuburst
     * @param algo name of the algorithm
     */
    public Sjnp(Map<Integer, Integer> mpp1, Map<Integer
, Integer> mpp2, String algo)
    {
        this.mpp1.putAll(mpp1);
        this.mpp2.putAll(mpp2);
        this.algo = algo;




    }

    /**
     * A method to output required tables in file
     */
    public void fileString()
    {
        try
        {
            FileWriter fileWriter = new FileWriter("
sjnp_out.txt");
            PrintWriter printwr = new PrintWriter(
fileWriter);
            Map<Integer, Integer> mpp3 = new HashMap<>(
);
            Map<Integer, Integer> mpp4 = new HashMap<>(
);

            int clock = 0;
            printwr.println("CPU sheduling algorithm: "
+algo);
            printwr.println("Total number of CPU
requests: "+mpp1.size());
            printwr.println(
"-------------------------------------------------------
---");
            String mi = "";
            Iterator<Integer> iter = mpp2.keySet().
iterator();
            int coun = 0;
            int cou = 0;
            int mini = 0;

            while(iter.hasNext()) {

                int itemd = iter.next();

                for (int i = 0; i < mpp2.size(); i++) {

                    if (mpp2.get(i) <= mpp2.get(mini))
{
                        if (mpp2.get(i) == mpp2.get(
mini)) {
                            if (i < mini) {
                                mini = i;
                            }
                        } else {
                            mini = i;
```

```
                         }
                    }



                }



                mpp4.put(cou, mpp2.get(mini));
                cou++;
                System.out.println(mini + " " + mpp2.
get(mini));

                printwr.println("Clock: " + clock);
                printwr.println("Pending CPU request(s
):");
                printwr.println(mini + " " + mpp1.get(
mini) + " " + mpp2.get(mini));

                clock += mpp2.get(mini);

                mpp3.put(coun, clock);
                coun++;



                mi = mini + " " + mpp1.get(mini) + " "
+ mpp2.get(mini);
                mpp1.put(mini, Integer.MAX_VALUE);
                mpp2.put(mini, Integer.MAX_VALUE);

                Iterator<Integer> iter1 = mpp1.keySet()
.iterator();

                while (iter1.hasNext()) {

                    int item2 = iter1.next();
                    if (mpp1.get(item2) < Integer.
MAX_VALUE) {
                        printwr.println(item2 + " " +
mpp1.get(item2) + " " + mpp2.get(item2));
                    }
                }
                printwr.println();
                printwr.println("CPU Request serviced 
during this clock interval: " + mi);
                printwr.println(
"------------------------------------------------------
---");
            }





            printwr.println("Turn-Around Time 
Computations");
            printwr.println();
            int count = 0;

            double sum = 0;
            for(int i = 0;i<mpp1.size();i++)
            {
                printwr.println("TAT("+i+") = "+mpp3.
get(i));
                sum+=mpp3.get(i);
                count++;
            }

            printwr.println();
            printwr.println("Average TAT = "+(sum/count
));
            printwr.println(
"------------------------------------------------------
---");

            printwr.println("Wait Time Computations");
            printwr.println();

            int count1 = 0;
            double sum1 = 0;

            for(int i = 0;i<mpp1.size();i++)
            {
                printwr.println("WT("+i+") = "+(mpp3.
get(i)-mpp4.get(i)));
                sum1+=(mpp3.get(i)-mpp4.get(i));
                count1++;
            }

            printwr.println();
            printwr.println("Average WT = "+(sum1/
count1));

            printwr.close();



        }
        catch (Exception e)
        {
            System.out.println("File can't be OPENED/
READ");
        }

    }





}
```