DOCUMENTATIE

TEMA 2

NUME STUDENT:CIONTE SERGIU IONUT

GRUPA:30228

CUPRINS

1.OBIECTIVUL TEMEI

1. OBIECTIVUL PRINCIPAL

Design-ul si implementarea unei aplicatii care sa analizeze sistemele bazate pe cozi simuland o serie de N clienti pentru servire, intrand in Q cozi, asteptand, fiind serviti si la final parasind cozile si calcularea timpului de asteptare mediu, al timpului de servire mediu si peak hour.

2.OBIECTIVE SECUNDARE

-Analiza problemei si identificare cerintelor

-Design-ul si simularea aplicatiei

-Implementarea simularii aplicatiei

-Testarea simularii aplicatiei

2.ANALIZA PROBLEMEI, MODELARE, SCENARII, CAZURI DE UTILIZARE

CERINTE FUNCTIONALE

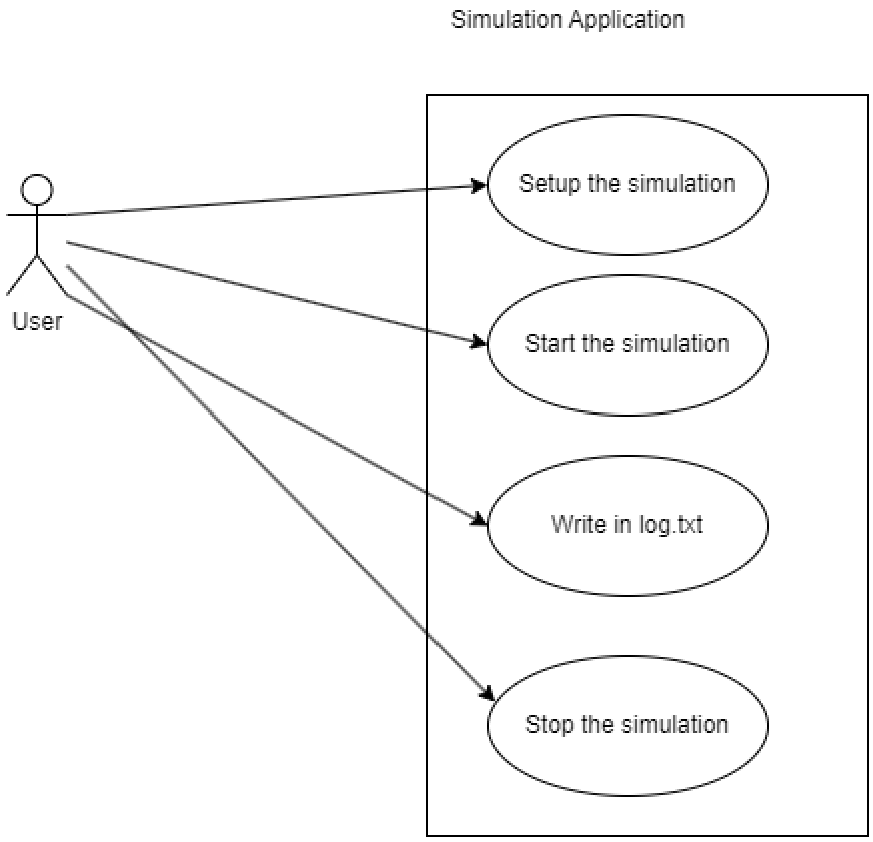
-simularea aplicatiei ar trebui sa permita utilizatorului sa initializeze simularea;

-simularea aplicatiei ar trebui sa permita utilizatorului sa inceapa simularea;

-simularea aplicatiei ar trebui sa afiseze in timp real evolutia cozilor;

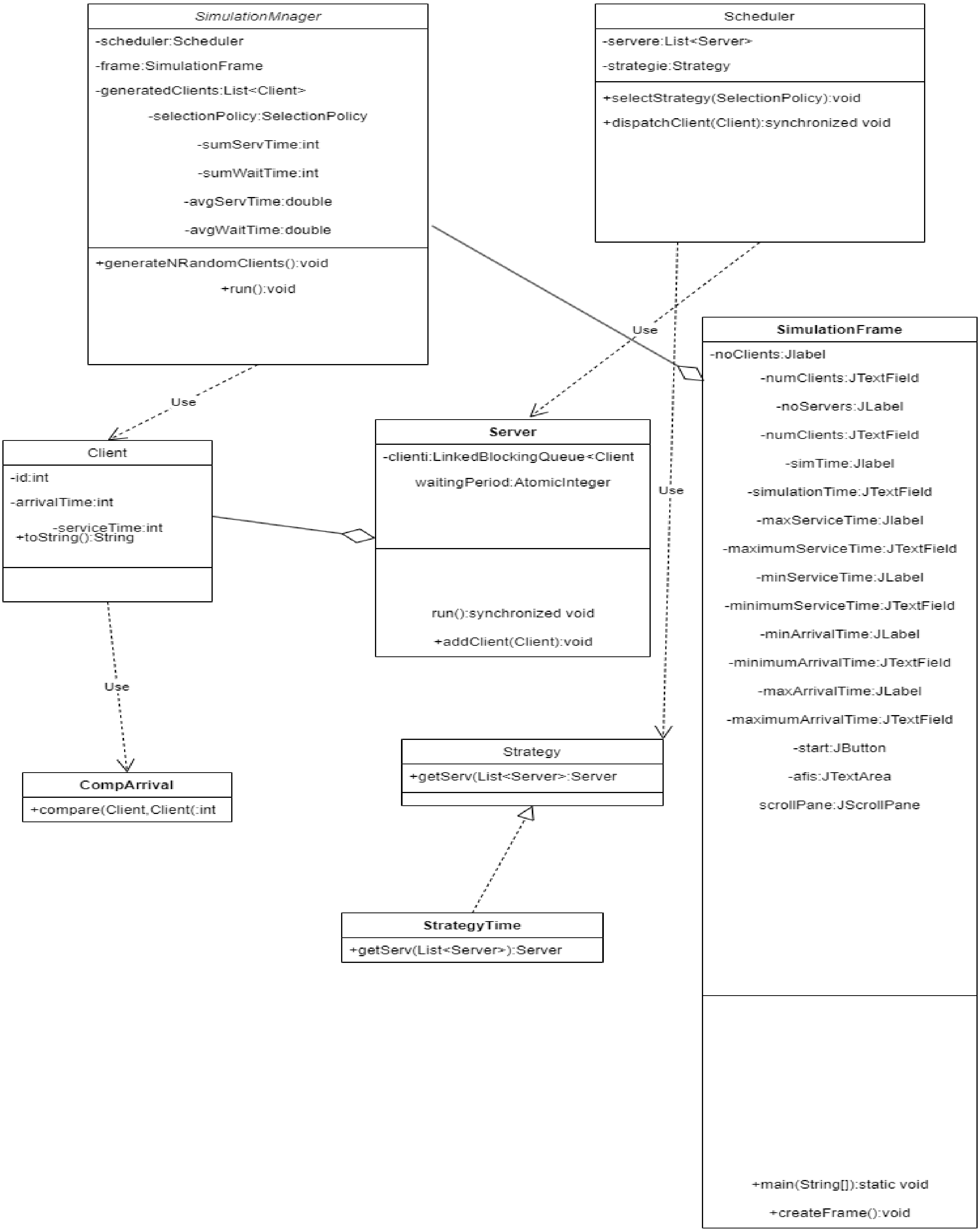
CERINTE NONFUNCTIONALE

-aplicatia ar trebui sa fie intuitiva si usor de folosit de catre utilizator;



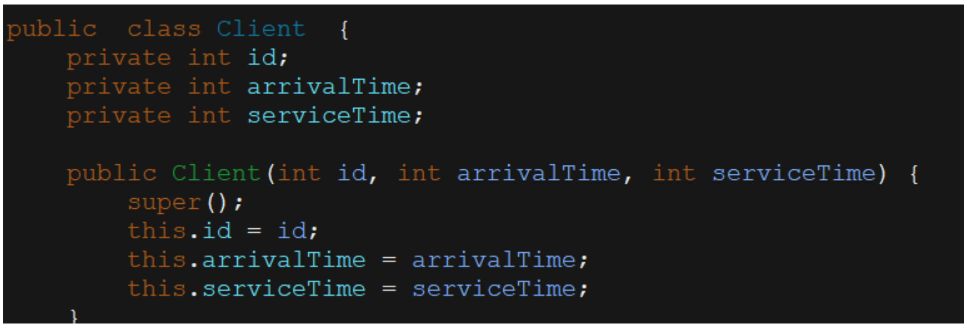
3.PROIECTARE

Diagrama de clase



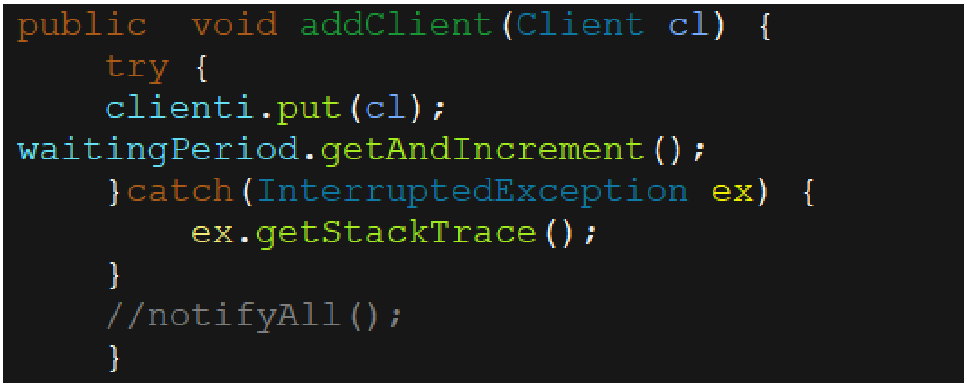
-Au fost folosite structurile LinkedBlocking Queue si AtomicInteger pentru a sincroniza corespunzator thread-urile;

<u>4.IMPLEMENTARE</u>



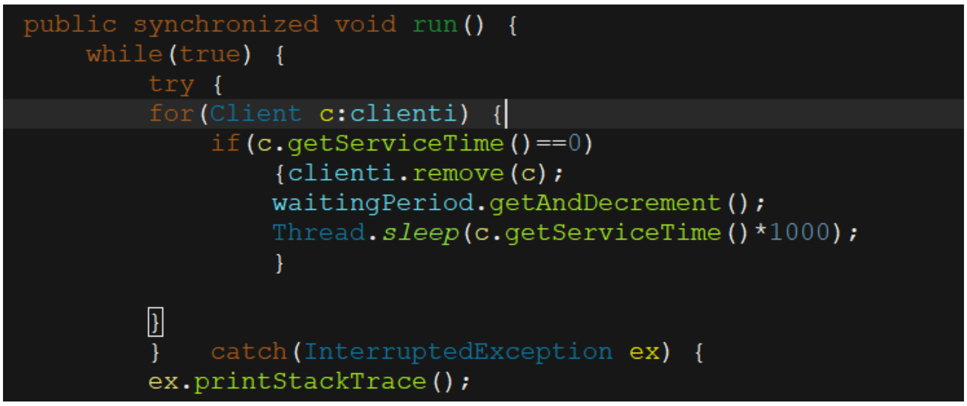
```
public  class Client  {
    private int id;
    private int arrivalTime;
    private int serviceTime;

    public Client(int id, int arrivalTime, int serviceTime) {
        super();
        this.id = id;
        this.arrivalTime = arrivalTime;
        this.serviceTime = serviceTime;
    }
```

In clasa <u>Client</u> , care reprezinta task-urile avem cele 3 atribute care descriu un client si anume id,arrivalTime,serviceTime;



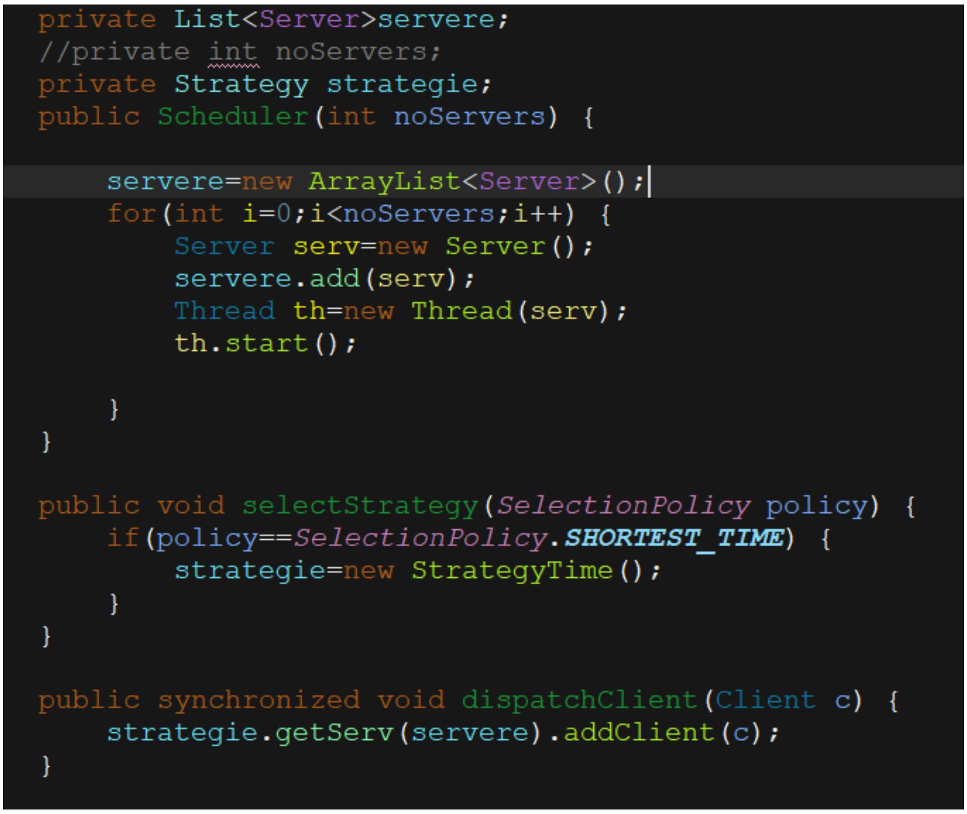
```
public  void addClient(Client cl)  {
    try {
    clienti.put(cl);
waitingPeriod.getAndIncrement();
    }catch(InterruptedException ex)  {
        ex.getStackTrace();
    }
    //notifyAll();
    }
```

In clasa avem metoda addClient care va adauga clientul in severul respectiv



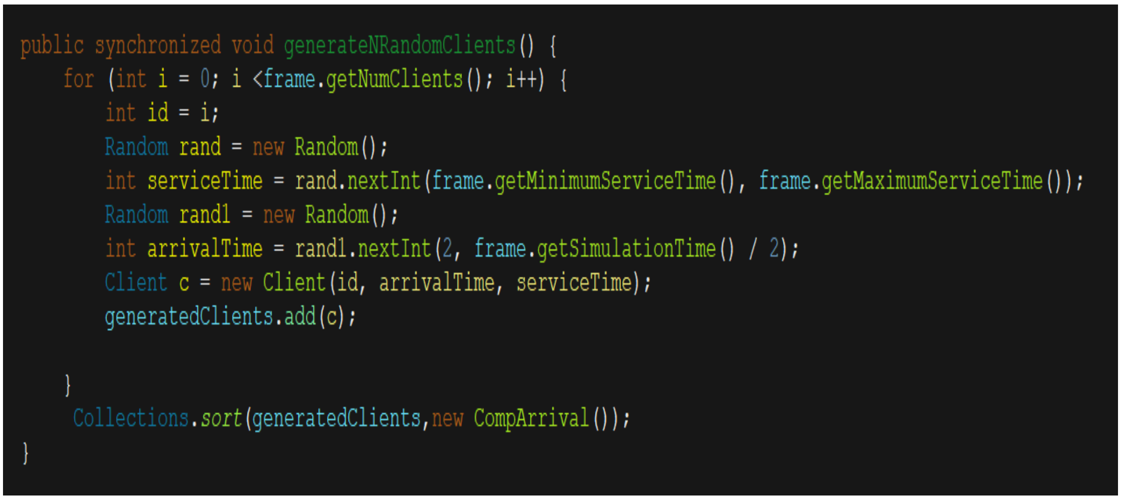
```
 public synchronized void run()  {
     while(true)  {
         try  {
         for(Client  c:clienti)  {
             if(c.getServiceTime()==0)
                 {clienti.remove(c);
                 waitingPeriod.getAndDecrement();
                 Thread.sleep(c.getServiceTime()*1000);
                 }

         }
         }    catch(InterruptedException  ex)  {
         ex.printStackTrace();
```

Precum si metoda run necesara deoarece clasa Server implementeaza Runnable, totodata aceasta metoda stergand clientul din coada pentru serviceTime=0;



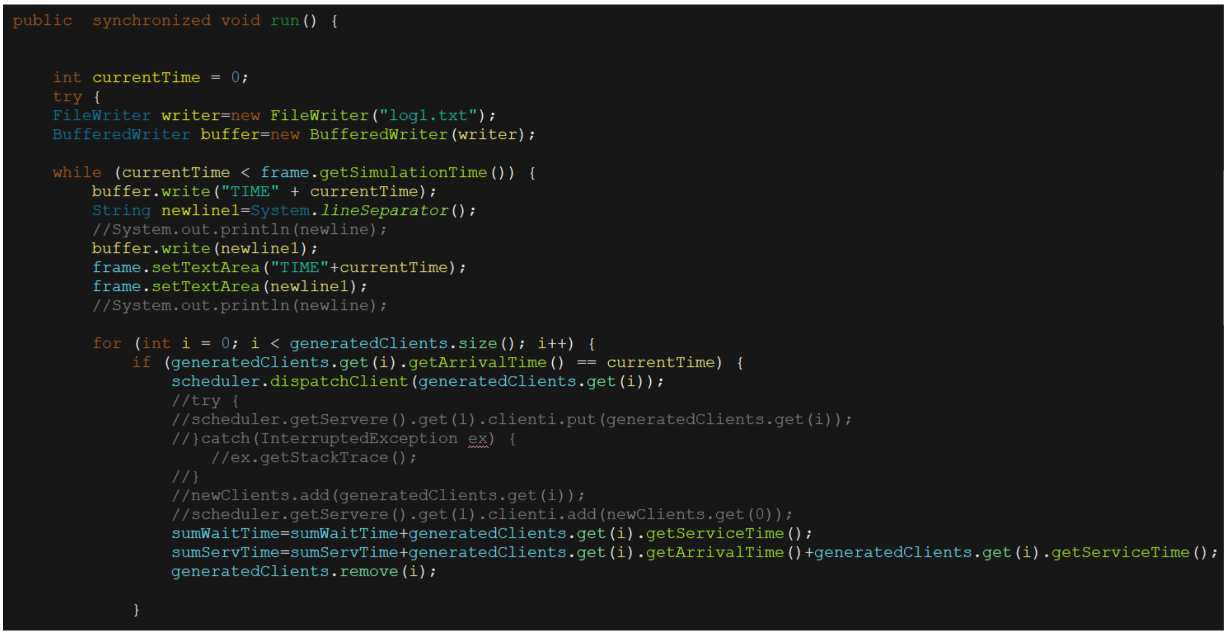
```java
private List<Server>servere;
//private int noServers;
private Strategy strategie;
public Scheduler(int noServers) {

    servere=new ArrayList<Server>();
    for(int i=0;i<noServers;i++) {
        Server serv=new Server();
        servere.add(serv);
        Thread th=new Thread(serv);
        th.start();

    }
}

public void selectStrategy(SelectionPolicy policy) {
    if(policy==SelectionPolicy.SHORTEST_TIME) {
        strategie=new StrategyTime();
    }
}

public synchronized void dispatchClient(Client c) {
    strategie.getServ(servere).addClient(c);
}
```

In clasa _initializam cate un thread pentru fiecare coada selectam strategia(in cazul de fata avem doar strategia shortest_time si apelam metoda getServ care ne returneaza coada la care trebuie sa fie pus clientul;

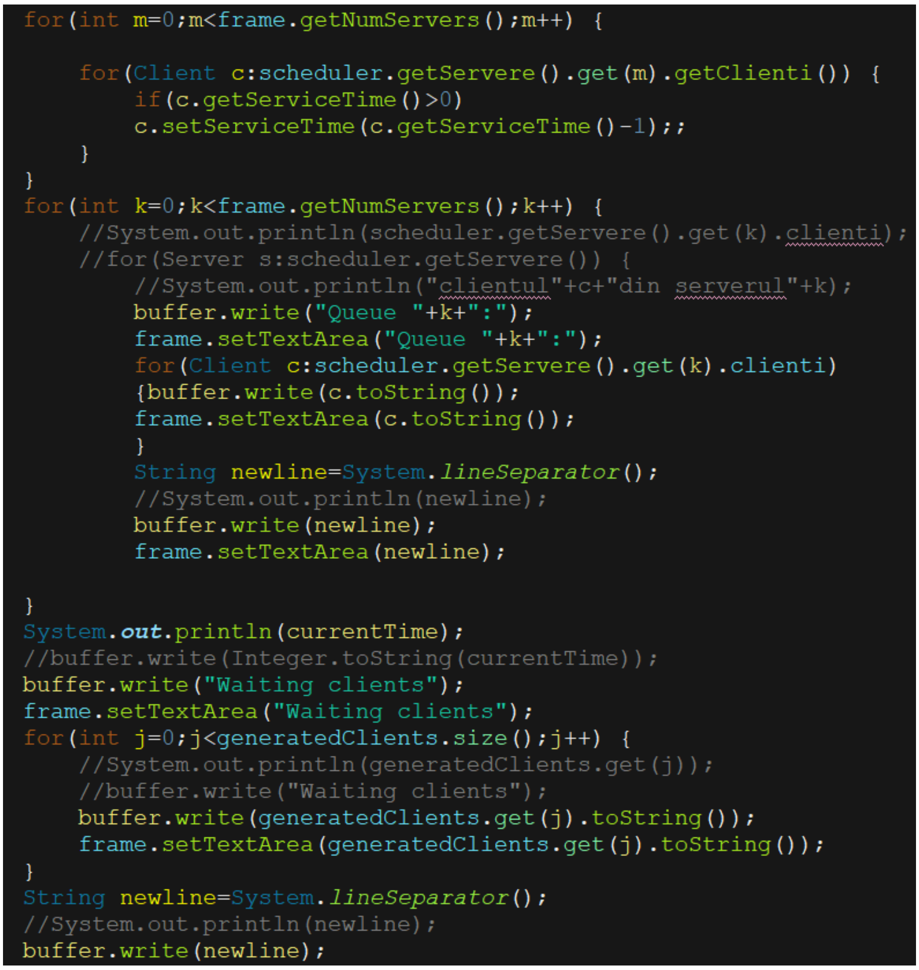


```java
public synchronized void generateNRandomClients() {
    for (int i = 0; i <frame.getNumClients(); i++) {
        int id = i;
        Random rand = new Random();
        int serviceTime = rand.nextInt(frame.getMinimumServiceTime(), frame.getMaximumServiceTime());
        Random rand1 = new Random();
        int arrivalTime = rand1.nextInt(2, frame.getSimulationTime() / 2);
        Client c = new Client(id, arrivalTime, serviceTime);
        generatedClients.add(c);

    }
     Collections.sort(generatedClients,new CompArrival());
}
```

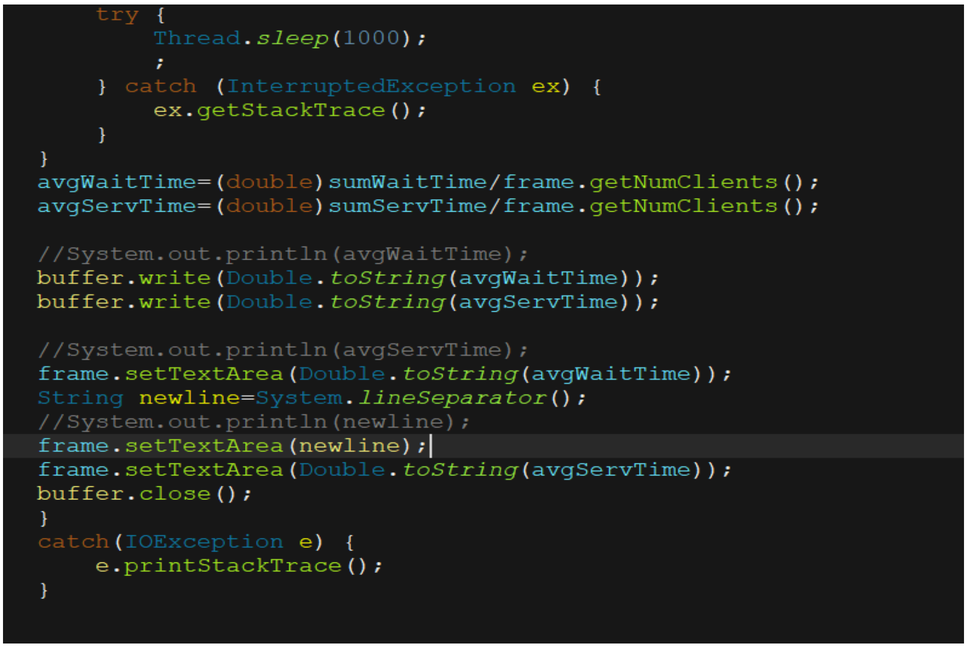
In SimulationManager avem metoda generateNRandomClients care va genera n clienti random conform informatiilor extrase din interfata;



```java
public  synchronized void run() {


    int currentTime = 0;
    try {
    FileWriter writer=new FileWriter("log1.txt");
    BufferedWriter buffer=new BufferedWriter(writer);

    while (currentTime < frame.getSimulationTime()) {
        buffer.write("TIME" + currentTime);
        String newline1=System.lineSeparator();
        //System.out.println(newline);
        buffer.write(newline1);
        frame.setTextArea("TIME"+currentTime);
        frame.setTextArea(newline1);
        //System.out.println(newline);

        for (int i = 0; i < generatedClients.size(); i++) {
            if (generatedClients.get(i).getArrivalTime() == currentTime) {
                scheduler.dispatchClient(generatedClients.get(i));
                //try {
                //scheduler.getServere().get(1).clienti.put(generatedClients.get(i));
                //}catch(InterruptedException ex) {
                    //ex.getStackTrace();
                //}
                //newClients.add(generatedClients.get(i));
                //scheduler.getServere().get(1).clienti.add(newClients.get(0));
                sumWaitTime=sumWaitTime+generatedClients.get(i).getServiceTime();
                sumServTime=sumServTime+generatedClients.get(i).getArrivalTime()+generatedClients.get(i).getServiceTime();
                generatedClients.remove(i);

            }
```

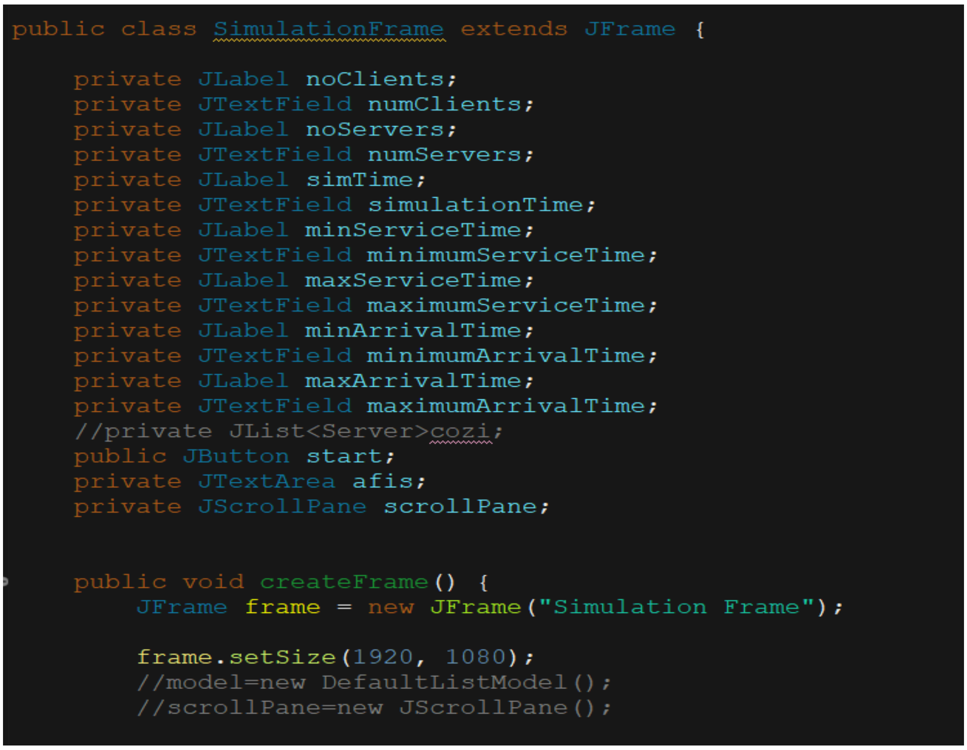


```
for(int m=0;m<frame.getNumServers();m++) {

    for(Client c:scheduler.getServere().get(m).getClienti()) {
        if(c.getServiceTime()>0)
        c.setServiceTime(c.getServiceTime()-1);;
    }
}
for(int k=0;k<frame.getNumServers();k++) {
    //System.out.println(scheduler.getServere().get(k).clienti);
    //for(Server s:scheduler.getServere()) {
        //System.out.println("clientul"+c+"din serverul"+k);
        buffer.write("Queue "+k+":");
        frame.setTextArea("Queue "+k+":");
        for(Client c:scheduler.getServere().get(k).clienti)
        {buffer.write(c.toString());
        frame.setTextArea(c.toString());
        }
        String newline=System.lineSeparator();
        //System.out.println(newline);
        buffer.write(newline);
        frame.setTextArea(newline);

}
System.out.println(currentTime);
//buffer.write(Integer.toString(currentTime));
buffer.write("Waiting clients");
frame.setTextArea("Waiting clients");
for(int j=0;j<generatedClients.size();j++) {
    //System.out.println(generatedClients.get(j));
    //buffer.write("Waiting clients");
    buffer.write(generatedClients.get(j).toString());
    frame.setTextArea(generatedClients.get(j).toString());
}
String newline=System.lineSeparator();
//System.out.println(newline);
buffer.write(newline);
```



```
      try {
          Thread.sleep(1000);
          ;
      } catch (InterruptedException ex) {
          ex.getStackTrace();
      }
 }
 avgWaitTime=(double)sumWaitTime/frame.getNumClients();
 avgServTime=(double)sumServTime/frame.getNumClients();

 //System.out.println(avgWaitTime);
 buffer.write(Double.toString(avgWaitTime));
 buffer.write(Double.toString(avgServTime));

 //System.out.println(avgServTime);
 frame.setTextArea(Double.toString(avgWaitTime));
 String newline=System.lineSeparator();
 //System.out.println(newline);
 frame.setTextArea(newline);
 frame.setTextArea(Double.toString(avgServTime));
 buffer.close();
 }
 catch(IOException e) {
     e.printStackTrace();
 }
```

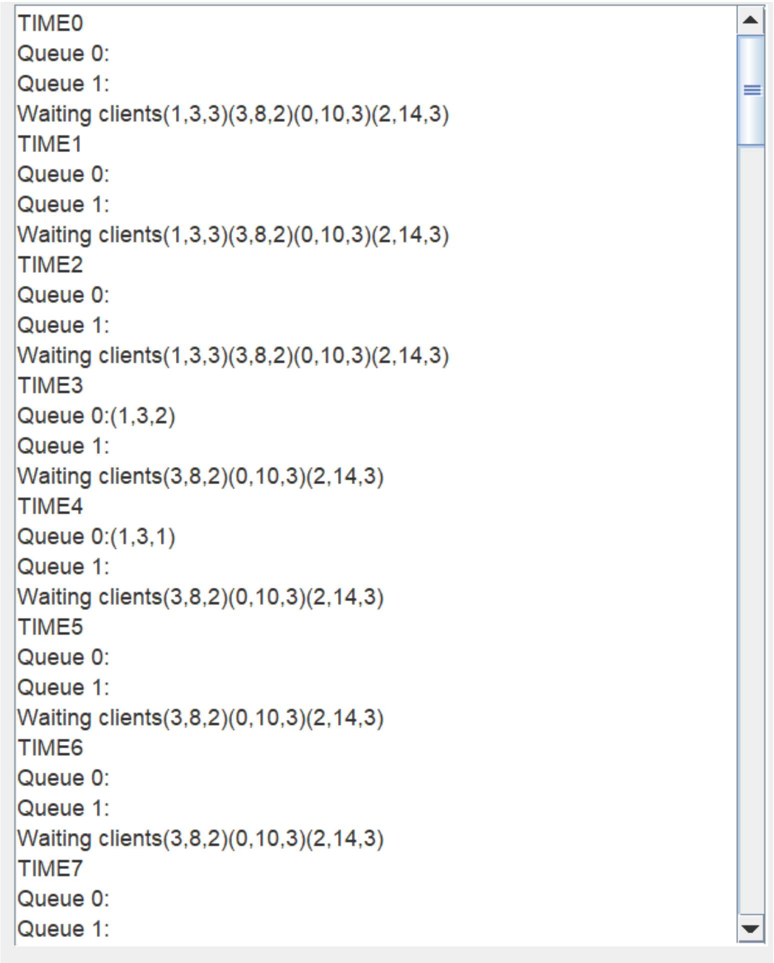
Metoda run prezentata extrage informatiile din interfata, calculeaza average waitingTime si average Service Time si scrie informatiile necesare in fisier, respectiv in JTextArea din interfata.



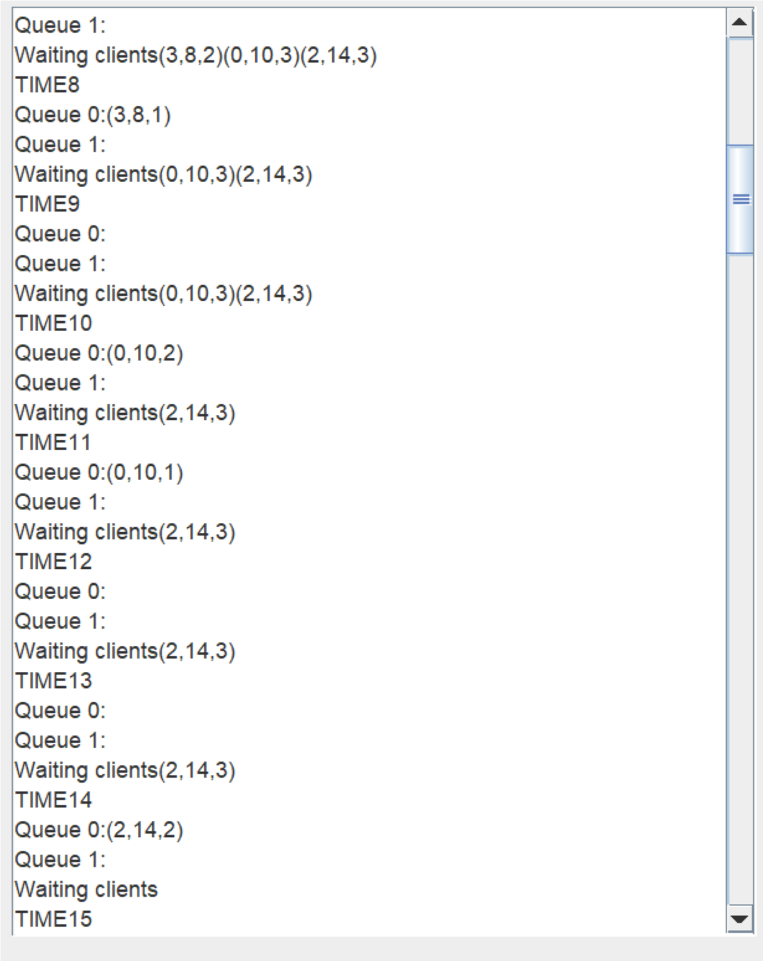
```
public class SimulationFrame extends JFrame {

    private JLabel noClients;
    private JTextField numClients;
    private JLabel noServers;
    private JTextField numServers;
    private JLabel simTime;
    private JTextField simulationTime;
    private JLabel minServiceTime;
    private JTextField minimumServiceTime;
    private JLabel maxServiceTime;
    private JTextField maximumServiceTime;
    private JLabel minArrivalTime;
    private JTextField minimumArrivalTime;
    private JLabel maxArrivalTime;
    private JTextField maximumArrivalTime;
    //private JList<Server>cozi;
    public JButton start;
    private JTextArea afis;
    private JScrollPane scrollPane;


    public void createFrame() {
        JFrame frame = new JFrame("Simulation Frame");

        frame.setSize(1920, 1080);
        //model=new DefaultListModel();
        //scrollPane=new JScrollPane();
```

In clasa SimulationFrame avem atributele de mai sus si metoda createFrame care va crea frame-ul ce va contine atributele specificate.

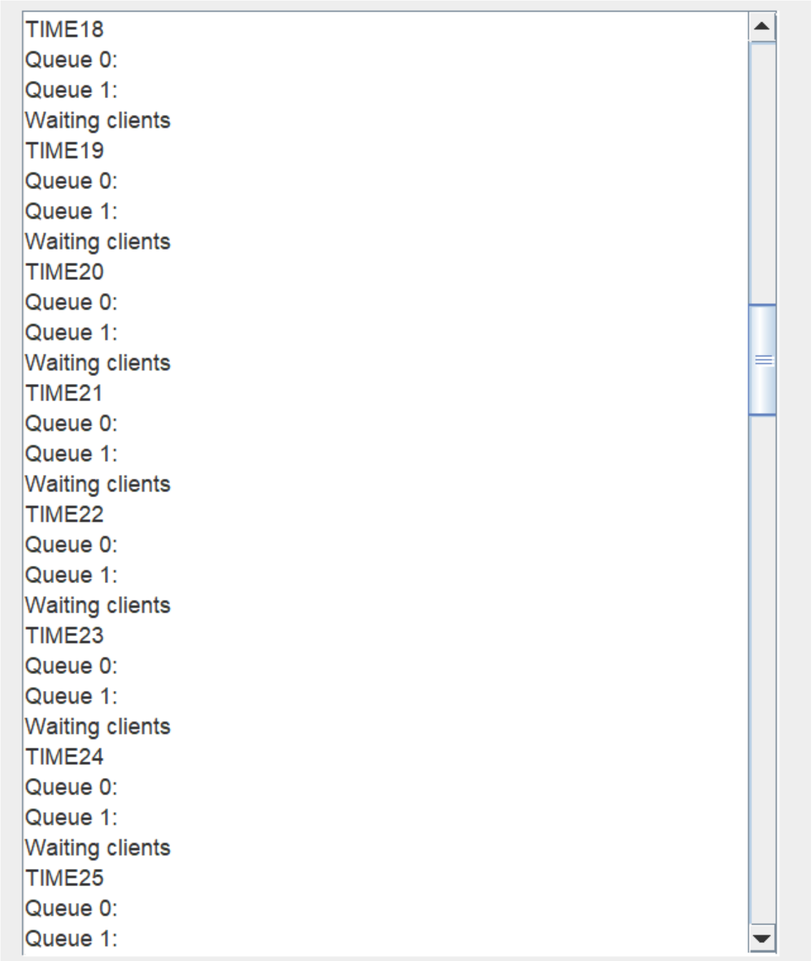
5.REZULTATE



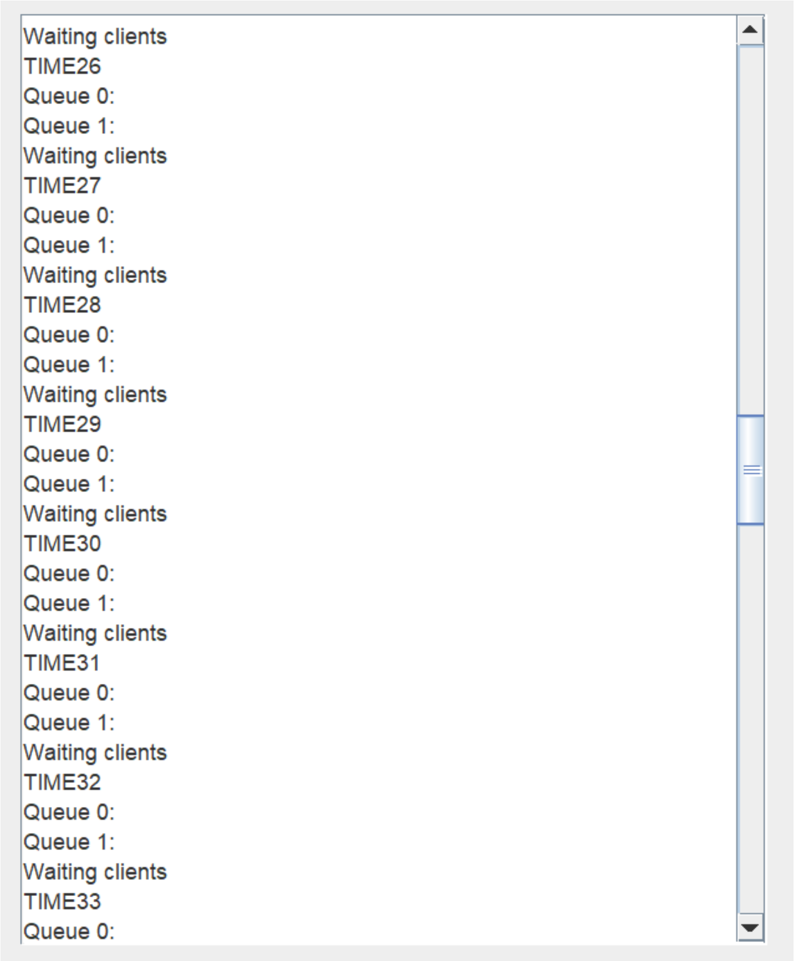
```
TIME0
Queue 0:
Queue 1:
Waiting clients(1,3,3)(3,8,2)(0,10,3)(2,14,3)
TIME1
Queue 0:
Queue 1:
Waiting clients(1,3,3)(3,8,2)(0,10,3)(2,14,3)
TIME2
Queue 0:
Queue 1:
Waiting clients(1,3,3)(3,8,2)(0,10,3)(2,14,3)
TIME3
Queue 0:(1,3,2)
Queue 1:
Waiting clients(3,8,2)(0,10,3)(2,14,3)
TIME4
Queue 0:(1,3,1)
Queue 1:
Waiting clients(3,8,2)(0,10,3)(2,14,3)
TIME5
Queue 0:
Queue 1:
Waiting clients(3,8,2)(0,10,3)(2,14,3)
TIME6
Queue 0:
Queue 1:
Waiting clients(3,8,2)(0,10,3)(2,14,3)
TIME7
Queue 0:
Queue 1:
```



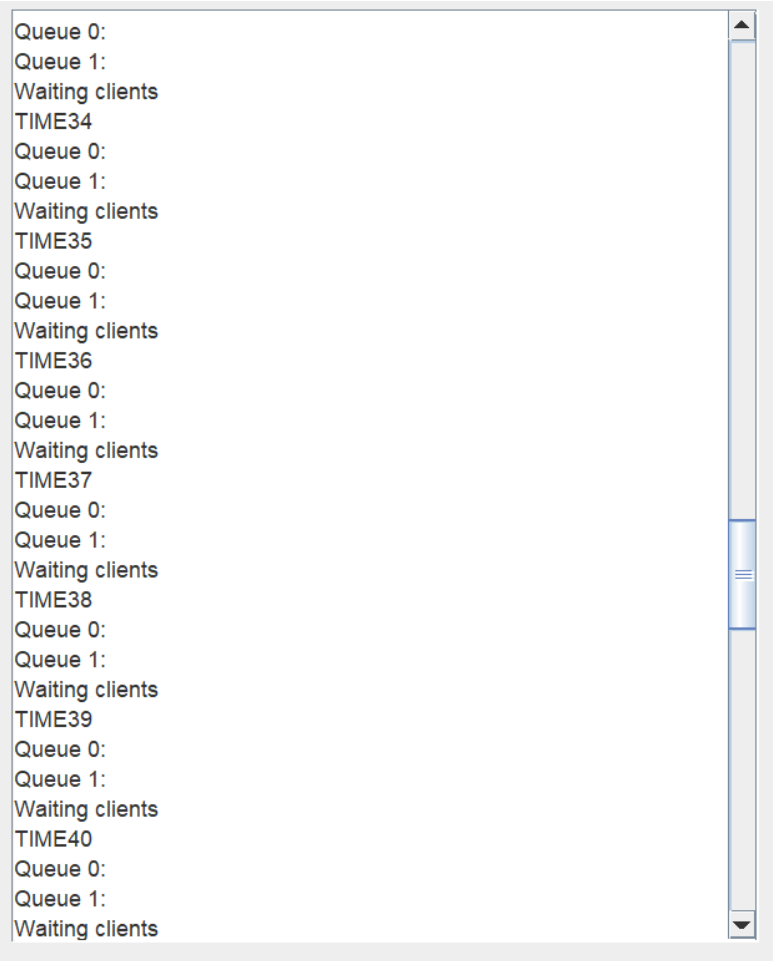
```
Queue 1:
Waiting clients(3,8,2)(0,10,3)(2,14,3)
TIME8
Queue 0:(3,8,1)
Queue 1:
Waiting clients(0,10,3)(2,14,3)
TIME9
Queue 0:
Queue 1:
Waiting clients(0,10,3)(2,14,3)
TIME10
Queue 0:(0,10,2)
Queue 1:
Waiting clients(2,14,3)
TIME11
Queue 0:(0,10,1)
Queue 1:
Waiting clients(2,14,3)
TIME12
Queue 0:
Queue 1:
Waiting clients(2,14,3)
TIME13
Queue 0:
Queue 1:
Waiting clients(2,14,3)
TIME14
Queue 0:(2,14,2)
Queue 1:
Waiting clients
TIME15
```



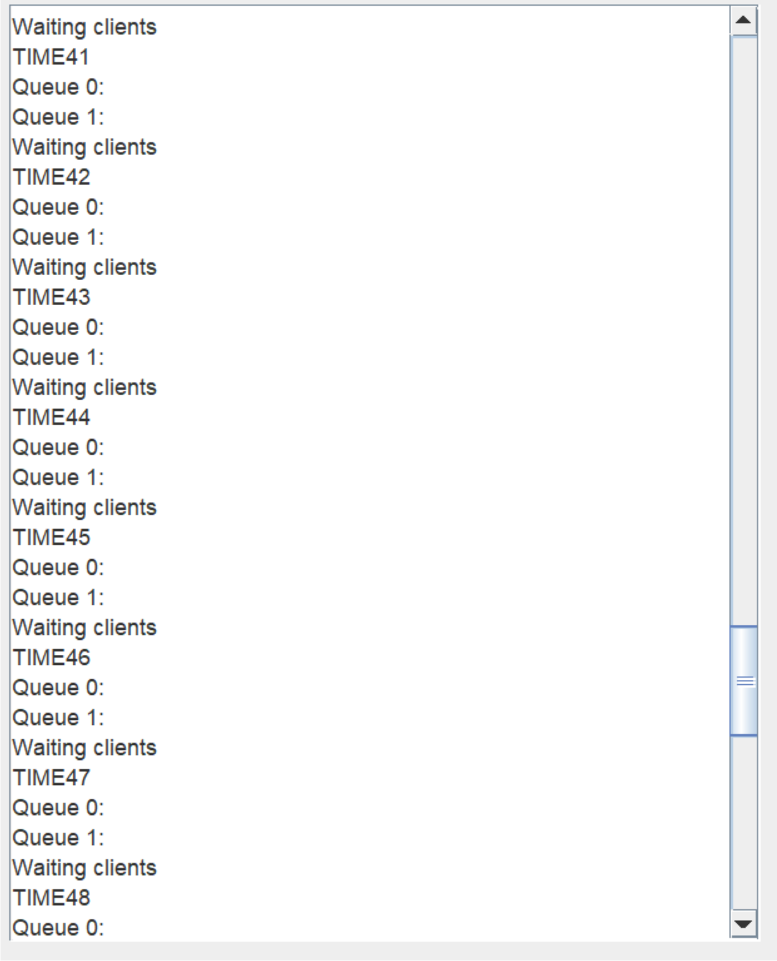
```
TIME18
Queue 0:
Queue 1:
Waiting clients
TIME19
Queue 0:
Queue 1:
Waiting clients
TIME20
Queue 0:
Queue 1:
Waiting clients
TIME21
Queue 0:
Queue 1:
Waiting clients
TIME22
Queue 0:
Queue 1:
Waiting clients
TIME23
Queue 0:
Queue 1:
Waiting clients
TIME24
Queue 0:
Queue 1:
Waiting clients
TIME25
Queue 0:
Queue 1:
```



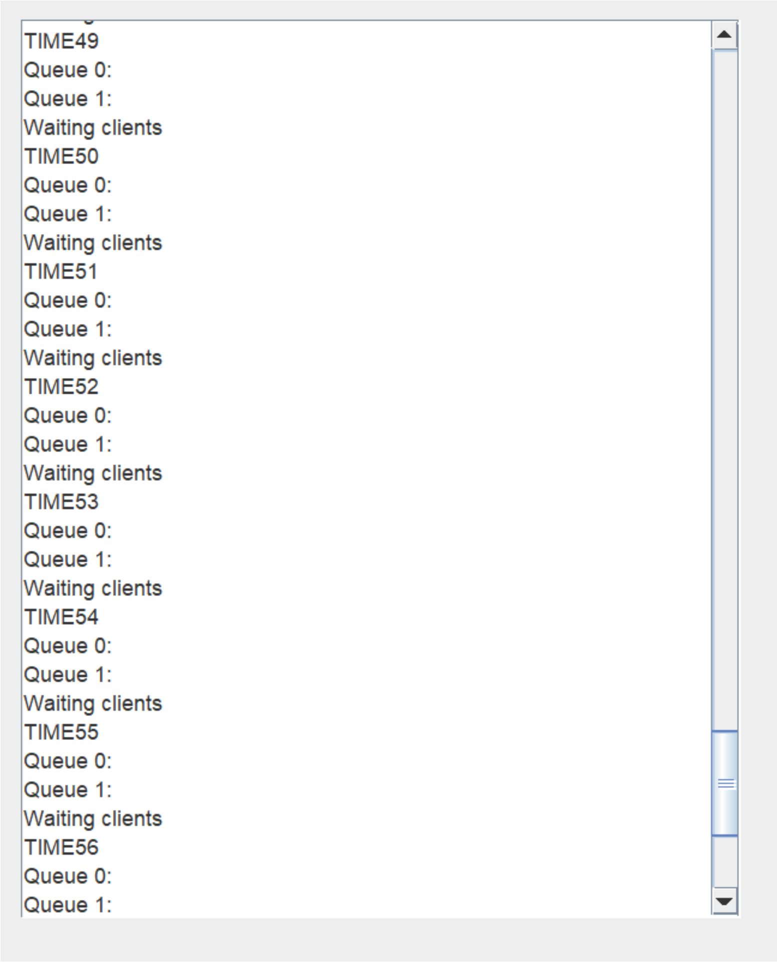
```
Waiting clients
TIME26
Queue 0:
Queue 1:
Waiting clients
TIME27
Queue 0:
Queue 1:
Waiting clients
TIME28
Queue 0:
Queue 1:
Waiting clients
TIME29
Queue 0:
Queue 1:
Waiting clients
TIME30
Queue 0:
Queue 1:
Waiting clients
TIME31
Queue 0:
Queue 1:
Waiting clients
TIME32
Queue 0:
Queue 1:
Waiting clients
TIME33
Queue 0:
```



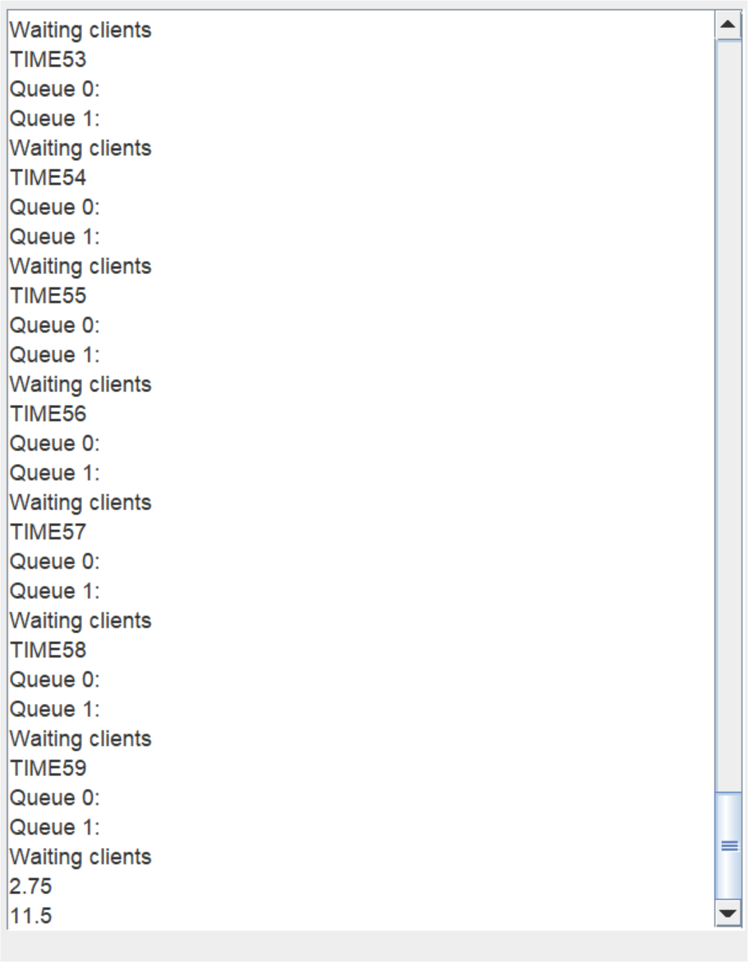
```
Queue 0:
Queue 1:
Waiting clients
TIME34
Queue 0:
Queue 1:
Waiting clients
TIME35
Queue 0:
Queue 1:
Waiting clients
TIME36
Queue 0:
Queue 1:
Waiting clients
TIME37
Queue 0:
Queue 1:
Waiting clients
TIME38
Queue 0:
Queue 1:
Waiting clients
TIME39
Queue 0:
Queue 1:
Waiting clients
TIME40
Queue 0:
Queue 1:
Waiting clients
```



```
Waiting clients
TIME41
Queue 0:
Queue 1:
Waiting clients
TIME42
Queue 0:
Queue 1:
Waiting clients
TIME43
Queue 0:
Queue 1:
Waiting clients
TIME44
Queue 0:
Queue 1:
Waiting clients
TIME45
Queue 0:
Queue 1:
Waiting clients
TIME46
Queue 0:
Queue 1:
Waiting clients
TIME47
Queue 0:
Queue 1:
Waiting clients
TIME48
Queue 0:
```



```
TIME49
Queue 0:
Queue 1:
Waiting clients
TIME50
Queue 0:
Queue 1:
Waiting clients
TIME51
Queue 0:
Queue 1:
Waiting clients
TIME52
Queue 0:
Queue 1:
Waiting clients
TIME53
Queue 0:
Queue 1:
Waiting clients
TIME54
Queue 0:
Queue 1:
Waiting clients
TIME55
Queue 0:
Queue 1:
Waiting clients
TIME56
Queue 0:
Queue 1:
```



```
Waiting clients
TIME53
Queue 0:
Queue 1:
Waiting clients
TIME54
Queue 0:
Queue 1:
Waiting clients
TIME55
Queue 0:
Queue 1:
Waiting clients
TIME56
Queue 0:
Queue 1:
Waiting clients
TIME57
Queue 0:
Queue 1:
Waiting clients
TIME58
Queue 0:
Queue 1:
Waiting clients
TIME59
Queue 0:
Queue 1:
Waiting clients
2.75
11.5
```

6.CONCLUZII

Tema presupune dezvoltarea cunostintelor despre thread-uri a metodelor de sincronizare, respectiv a scrierii in fisiere a unui log of events.

7.BIBLIOGRAFIE

Thread synchronization in java: https://www.javatpoint.com/synchronization-in-java

LinkedBlockingQueue in java:

https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/LinkedBlockingQueue.html