

# **DOCUMENTATIE**

## **TEMA 2**

**NUME STUDENT:CIONTE SERGIU IONUT**

**GRUPA:30228**

# CUPRINS

1. OBIECTIVUL TEMEI
2. ANALIZA PROBLEMEI, MODELARE, SCENARII, CAZURI DE UTILIZARE
3. PEOIWCTARE
4. IMPLEMENTARE
5. REZULTATE
6. CONCLUZII
7. BIBLIOGRAFIE

## 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

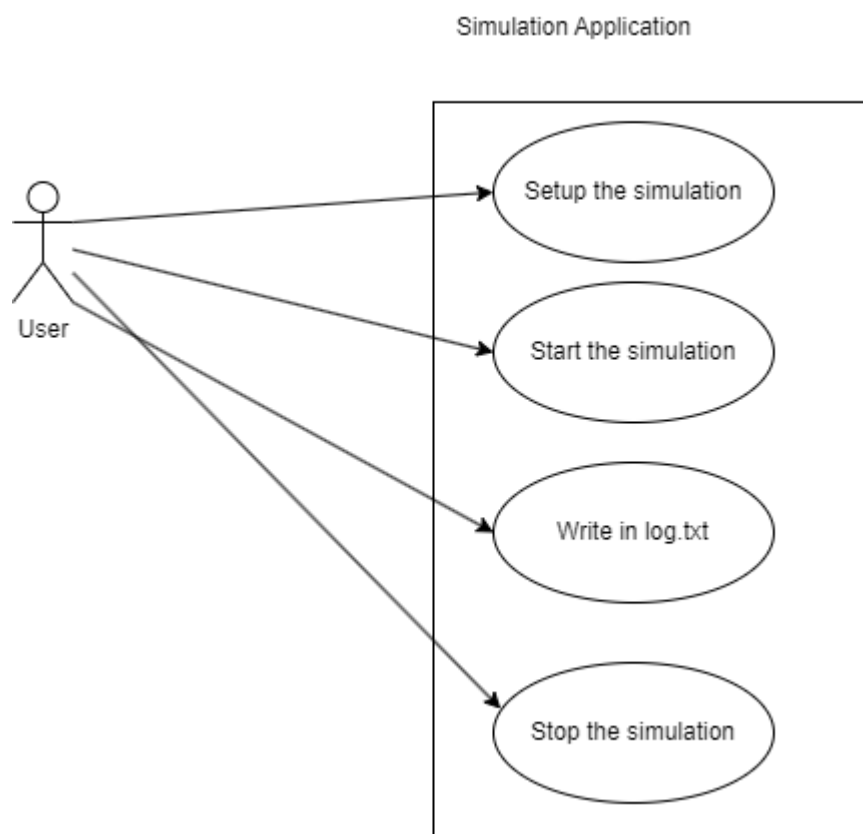
## 2.ANALIZA PROBLEMEI, MODELARE, SCENARII, CAZURI DE UTILIZARE

## 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;

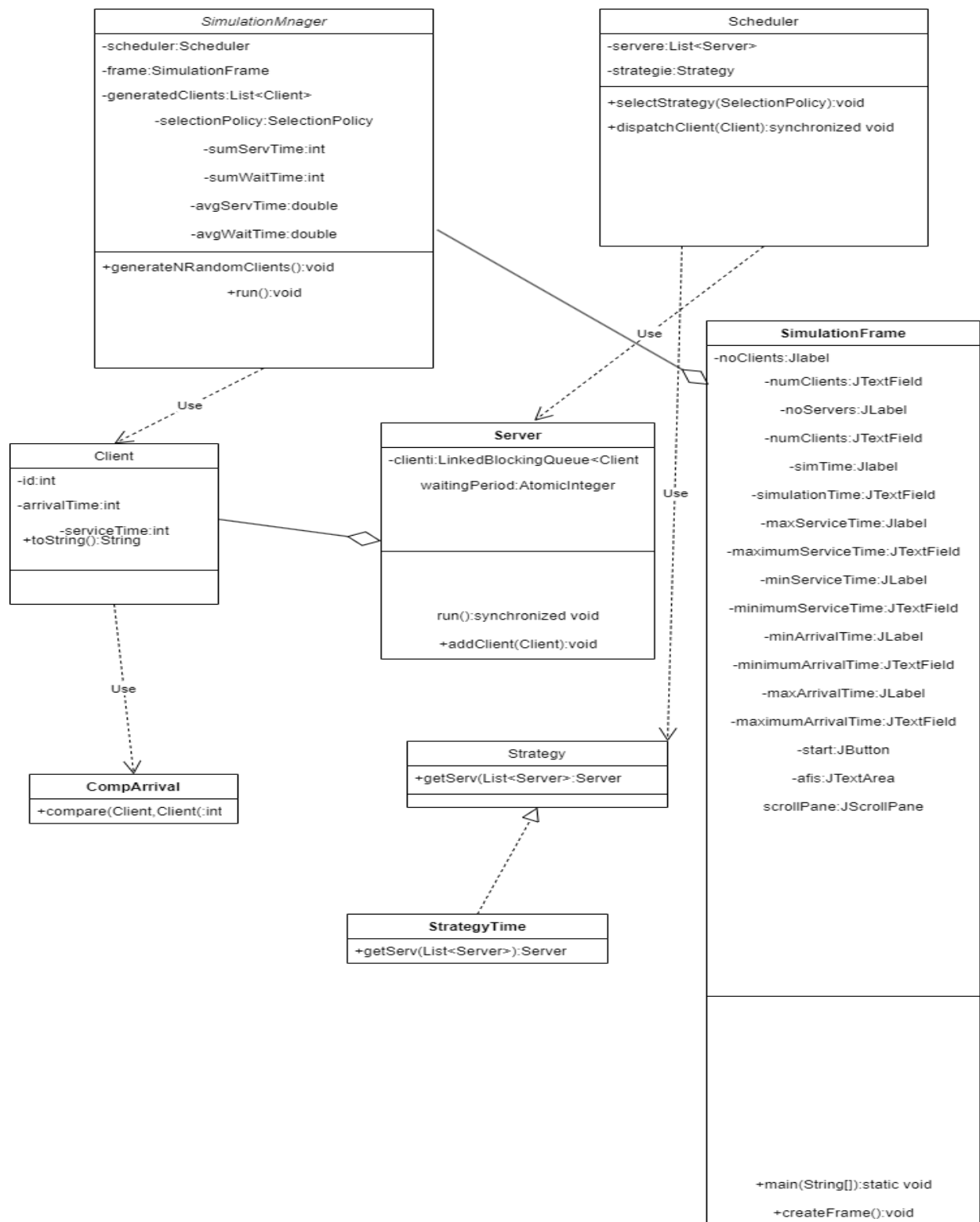
## CERINTE NONFUNCTIONALE

- 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;

## 4.IMPLEMENTARE

```
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 Client , care reprezinta task-urile avem cele 3 attribute care descriu un client si anume id,arrivalTime,serviceTime;

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

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

```
public synchronized void run() {
    while(true) {
        try {
            for(Client c:clienti) {
                if(c.getServiceTime()==0)
                {c.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;

```

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;

```

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;

```

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 newline = System.lineSeparator();
            //System.out.println(newline);
            buffer.write(newline);
            frame.setTextArea("TIME"+currentTime);
            frame.setTextArea(newline);
            //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.getServer().get(1).clienti.put(generatedClients.get(i));
                    //} catch (InterruptedException ex) {
                    //    ex.printStackTrace();
                    //}
                    //newClients.add(generatedClients.get(i));
                    //scheduler.getServer().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.printStackTrace();
        }
    }
    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 attributele de mai sus si metoda createFrame care va crea frame-ul ce va contine attributele 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

TIME0  
Queue 0:  
Queue 1:  
Queue 2:  
Queue 3:  
Waiting clients(9,2,3)(25,2,6)(27,2,1)(38,2,1)(4,3,4)(34,3,3)(45,3,4)(12,4,5)  
TIME1  
Queue 0:  
Queue 1:  
Queue 2:  
Queue 3:  
Waiting clients(9,2,3)(25,2,6)(27,2,1)(38,2,1)(4,3,4)(34,3,3)(45,3,4)(12,4,5)  
TIME2  
Queue 0:(9,2,2)  
Queue 1:(25,2,5)  
Queue 2:  
Queue 3:  
Waiting clients(4,3,4)(34,3,3)(45,3,4)(12,4,5)(21,4,3)(11,6,3)(13,6,1)(14,6,2)  
TIME3  
Queue 0:(9,2,1)(45,3,3)  
Queue 1:(25,2,4)  
Queue 2:(4,3,3)  
Queue 3:(34,3,2)  
Waiting clients(12,4,5)(21,4,3)(11,6,3)(13,6,1)(14,6,2)(48,7,2)(36,8,3)(18,9,3)  
TIME4  
Queue 0:(45,3,2)  
Queue 1:(25,2,3)(12,4,4)  
Queue 2:(4,3,2)(21,4,2)  
Queue 3:(34,3,1)  
Waiting clients(11,6,3)(13,6,1)(14,6,2)(48,7,2)(36,8,3)(18,9,3)(35,9,2)(41,9,2)

TIME5  
Queue 0:(45,3,1)  
Queue 1:(25,2,2)(12,4,3)  
Queue 2:(4,3,1)(21,4,1)  
Queue 3:  
Waiting clients(11,6,3)(13,6,1)(14,6,2)(48,7,2)(36,8,3)(18,9,3)(35,9,2)(41,10,6)(0,11,6)(19,12,2)(23,12,2)(44,12,2)(30,13,5)(17,14,1)

TIME6  
Queue 0:  
Queue 1:(25,2,1)(12,4,2)  
Queue 2:  
Queue 3:(11,6,2)(14,6,1)  
Waiting clients(48,7,2)(36,8,3)(18,9,3)(35,9,2)(41,10,6)(0,11,6)(19,12,2)(23,12,2)(44,12,2)(30,13,5)(17,14,1)

TIME7  
Queue 0:(48,7,1)  
Queue 1:(12,4,1)  
Queue 2:  
Queue 3:(11,6,1)  
Waiting clients(36,8,3)(18,9,3)(35,9,2)(41,10,6)(0,11,6)(19,12,2)(23,12,2)(44,12,2)(30,13,5)(17,14,1)

TIME8  
Queue 0:  
Queue 1:  
Queue 2:(36,8,2)  
Queue 3:  
Waiting clients(18,9,3)(35,9,2)(41,10,6)(0,11,6)(19,12,2)(23,12,2)(44,12,2)(30,13,5)(17,14,1)

TIME9  
Queue 0:(18,9,2)  
Queue 1:(35,9,1)  
Queue 2:(36,8,1)  
Queue 3:  
Waiting clients(41,10,6)(0,11,6)(19,12,2)(23,12,2)(44,12,2)(30,13,5)(17,14,1)

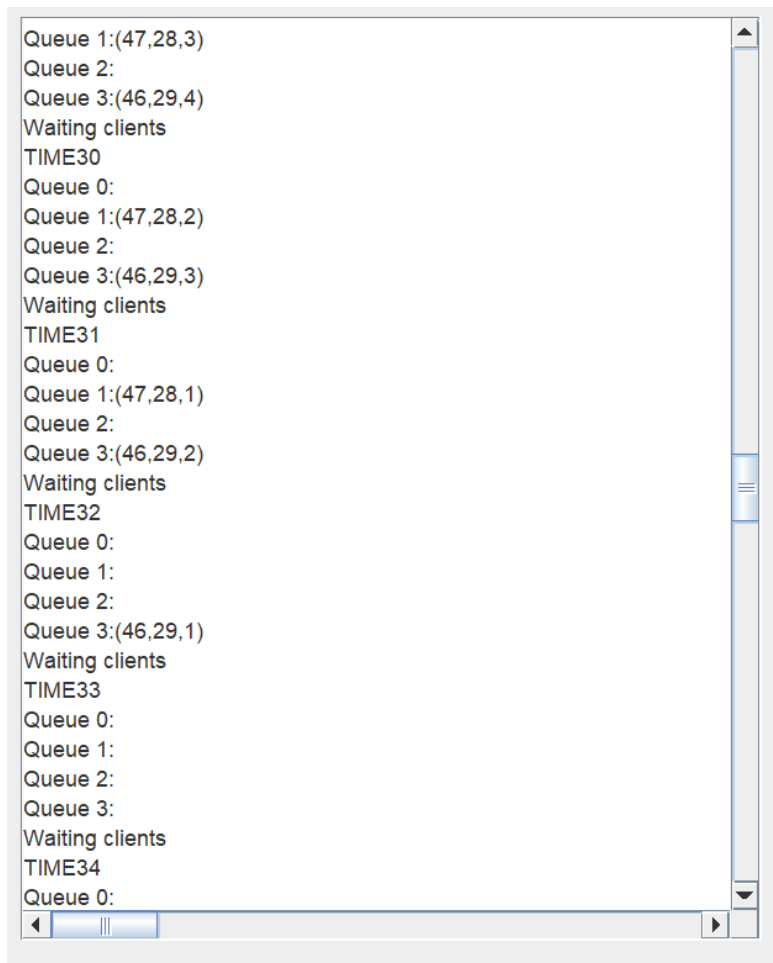
Waiting clients(41,10,6)(0,11,6)(19,12,2)(23,12,2)(44,12,2)(30,13,5)(17,14,4)  
TIME10  
Queue 0:(18,9,1)  
Queue 1:  
Queue 2:  
Queue 3:(41,10,5)  
Waiting clients(0,11,6)(19,12,2)(23,12,2)(44,12,2)(30,13,5)(17,14,4)(24,14,6)(1,15,6)  
TIME11  
Queue 0:  
Queue 1:(0,11,5)  
Queue 2:  
Queue 3:(41,10,4)  
Waiting clients(19,12,2)(23,12,2)(44,12,2)(30,13,5)(17,14,4)(24,14,6)(1,15,6)  
TIME12  
Queue 0:(19,12,1)(44,12,1)  
Queue 1:(0,11,4)  
Queue 2:(23,12,1)  
Queue 3:(41,10,3)  
Waiting clients(30,13,5)(17,14,4)(24,14,6)(1,15,6)(31,15,3)(3,17,5)(32,17,6)(42,17,6)  
TIME13  
Queue 0:  
Queue 1:(0,11,3)(30,13,4)  
Queue 2:  
Queue 3:(41,10,2)  
Waiting clients(17,14,4)(24,14,6)(1,15,6)(31,15,3)(3,17,5)(32,17,6)(42,17,6)  
TIME14  
Queue 0:(17,14,3)  
Queue 1:(0,11,2)(30,13,3)  
Queue 2:(24,14,5)  
Queue 3:(41,10,1)

Queue 1:(0,11,2)(30,13,3)  
Queue 2:(24,14,5)  
Queue 3:(41,10,1)  
Waiting clients(1,15,6)(31,15,3)(3,17,5)(32,17,6)(42,17,3)(26,18,3)(16,19,6)  
TIME15  
Queue 0:(17,14,2)(1,15,5)  
Queue 1:(0,11,1)(30,13,2)  
Queue 2:(24,14,4)(31,15,2)  
Queue 3:  
Waiting clients(3,17,5)(32,17,6)(42,17,3)(26,18,3)(16,19,6)(8,20,1)(5,21,4)  
TIME16  
Queue 0:(17,14,1)(1,15,4)  
Queue 1:(30,13,1)  
Queue 2:(24,14,3)(31,15,1)  
Queue 3:  
Waiting clients(3,17,5)(32,17,6)(42,17,3)(26,18,3)(16,19,6)(8,20,1)(5,21,4)  
TIME17  
Queue 0:(1,15,3)  
Queue 1:(32,17,5)  
Queue 2:(24,14,2)  
Queue 3:(3,17,4)(42,17,2)  
Waiting clients(26,18,3)(16,19,6)(8,20,1)(5,21,4)(2,22,1)(15,22,6)(37,22,1)  
TIME18  
Queue 0:(1,15,2)(26,18,2)  
Queue 1:(32,17,4)  
Queue 2:(24,14,1)  
Queue 3:(3,17,3)(42,17,1)  
Waiting clients(16,19,6)(8,20,1)(5,21,4)(2,22,1)(15,22,6)(37,22,1)(43,22,1)  
TIME19  
Queue 0:(1,15,1)(26,18,1)

Waiting clients(8,20,1)(5,21,4)(2,22,1)(15,22,6)(37,22,1)(43,22,4)(20,23,4)  
TIME20  
Queue 0:  
Queue 1:(32,17,2)(16,19,4)  
Queue 2:  
Queue 3:(3,17,1)  
Waiting clients(5,21,4)(2,22,1)(15,22,6)(37,22,1)(43,22,4)(20,23,4)(29,24,6)  
TIME21  
Queue 0:(5,21,3)  
Queue 1:(32,17,1)(16,19,3)  
Queue 2:  
Queue 3:  
Waiting clients(2,22,1)(15,22,6)(37,22,1)(43,22,4)(20,23,4)(29,24,6)(7,25,5)  
TIME22  
Queue 0:(5,21,2)  
Queue 1:(16,19,2)  
Queue 2:(43,22,3)  
Queue 3:(15,22,5)  
Waiting clients(20,23,4)(29,24,6)(7,25,5)(10,25,2)(39,25,2)(40,26,2)(49,26,4)  
TIME23  
Queue 0:(5,21,1)(20,23,3)  
Queue 1:(16,19,1)  
Queue 2:(43,22,2)  
Queue 3:(15,22,4)  
Waiting clients(29,24,6)(7,25,5)(10,25,2)(39,25,2)(40,26,2)(49,26,4)(28,17,1)  
TIME24  
Queue 0:(20,23,2)  
Queue 1:(29,24,5)  
Queue 2:(43,22,1)  
Queue 3:(15,22,3)

Waiting clients(7,25,5)(10,25,2)(39,25,2)(40,26,2)(49,26,4)(28,27,3)(22,2  
TIME25  
Queue 0:(20,23,1)(7,25,4)  
Queue 1:(29,24,4)(10,25,1)  
Queue 2:(39,25,1)  
Queue 3:(15,22,2)  
Waiting clients(40,26,2)(49,26,4)(28,27,3)(22,28,1)(47,28,5)(6,29,1)(33,2  
TIME26  
Queue 0:(7,25,3)  
Queue 1:(29,24,3)  
Queue 2:(40,26,1)  
Queue 3:(15,22,1)(49,26,3)  
Waiting clients(28,27,3)(22,28,1)(47,28,5)(6,29,1)(33,29,1)(46,29,5)  
TIME27  
Queue 0:(7,25,2)(28,27,2)  
Queue 1:(29,24,2)  
Queue 2:  
Queue 3:(49,26,2)  
Waiting clients(22,28,1)(47,28,5)(6,29,1)(33,29,1)(46,29,5)  
TIME28  
Queue 0:(7,25,1)(28,27,1)  
Queue 1:(29,24,1)(47,28,4)  
Queue 2:  
Queue 3:(49,26,1)  
Waiting clients(6,29,1)(33,29,1)(46,29,5)  
TIME29  
Queue 0:  
Queue 1:(47,28,3)  
Queue 2:  
Queue 3:(46,29,4)





## 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>



