

MINISTERUL EDUCAȚIEI ȘI CERCETĂRII ȘTIINȚIFICE



**FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
CATEDRA CALCULATOARE**

FUNDAMENTAL PROGRAMMING TECHNIQUES

ASSIGNMENT 2

Queues simulator

Student: Torzsa Gabriela

An studiu: II, semestrul II

Grupa: 30226

CUPRINS

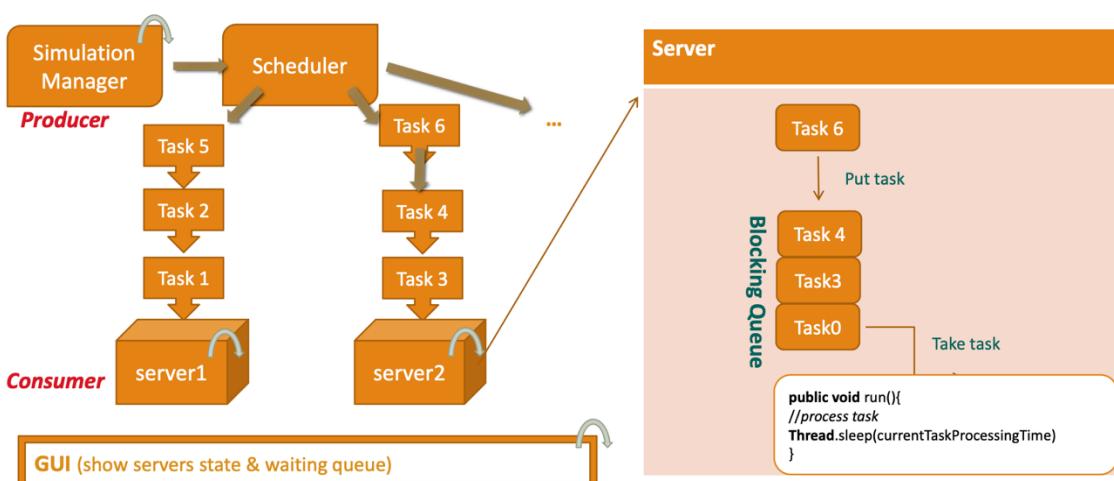
- I. Obiectivul temei
- II. Analiza problemei, modelare, scenarii, cazuri de utilizare
- III. Proiectare
- IV. Implementare
- V. Rezultate
- VI. Concluzii
- VII. Bibliografie

I. Obiectivul temei

Tema proiectului a fost să implementăm o aplicație care are ca scop analiza cozilor și performanța acestora pe baza unor algoritmi de determinare a timpului minim de aşteptare a clientilor și repartizării acestora într-un mod optim. Fiecare client este caracterizat de un număr de identificare, timpul la care acesta intră în coadă și timpul necesar servirii serviciului acestuia într-un timp maxim de simulare de definit de către utilizator.

Cozile sunt folosite pentru a oferi unui client un loc în care acesta să poată aștepta servirea sau îndeplinirea unui serviciu. Optimizarea cozilor se bazează pe minimizarea timpului de așteptare a clientului până la momentul servirii acestuia. Există mai multe metode de minimizare, precum: adăugarea mai multor ghișee (cozi), această soluție însă având ca dezavantaj un număr ridicat de costuri, sau repartizarea clientilor în funcție de coada cea mai scurtă sau coada a cărui timp de așteptare este cel mai scurt.

II. Analiza problemei, modelare, scenarii, cazuri de utilizare



Diagramă preluată din suportul de prezentare a temei

Elementele principale necesare aplicației sunt un *simulation manager* care reprezintă creierul întregului program, un *scheduler* care împarte clienții (task-urile) în cozi și *serverele* care tratează clienții și serviciile acestora.

Aplicația trebuie să lucreze cu mai multe fire de execuție (threaduri) pentru optimizare, adică se vor folosi fire de execuție pentru lansarea cozilor, pentru a procesa clienții și pentru distribuirea acestora la coada optimă, pe tot parcursul simulării.

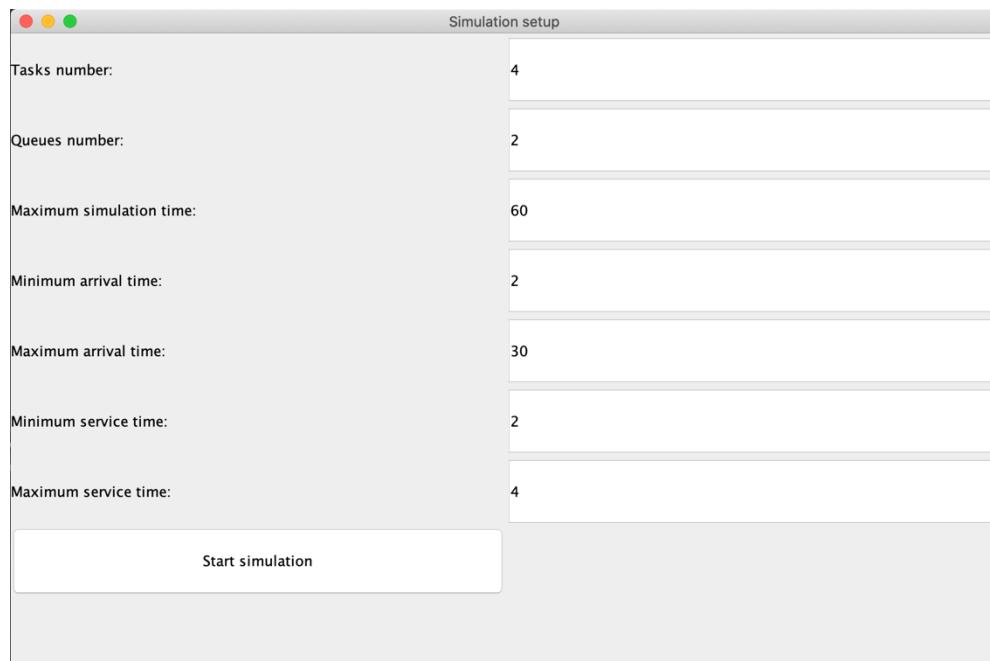
Cu ajutorul unei interfețe grafice, utilizatorul poate introduce valorile necesare simulării, și după pornirea acesteia, el poate vizualiza în timp real evoluția cozilor. Elementele necesare sunt: numărul de clienți care trebuie serviti, numărul de cozi, perioada de simulare, timpul minim și maxim în care un client poate intra în coadă și timpul minim și maxim necesar servirii unui client. După introducerea acestor valori, utilizatorul va putea vizualiza, într-un mod interactiv, cum clienții intră și ies din cozi.

III. Proiectare

Aplcația este proiectată în diferite pachete, cu diferite clase, fiecare clasă având un scop bine definit. Prezentarea acestor clase o voi face în detaliu ulterior în partea de *Implementare*. Aplicația este formată din 6 pachete:

1. *file* – care conține clase de fișiere și management-ul acestora –
2. *simulation* – care conține pachetele *model* și *strategy*, primul conținând clase *model*, iar al doilea metodele de strategie pentru alegerea cozii în care va fi repartizat clientul; în pachetul *simulation* se află clasele care reprezintă defapt simularea;
3. *ui* – care conține clasele pentru interfața grafică
4. *utils* – în care avem clasa de comparare a clienților în funcție de momentul în care aceștia trebuie să intre în coadă.
5. pachetul *default* – care conține toate pachetele menționate mai sus și clasa principală *main*

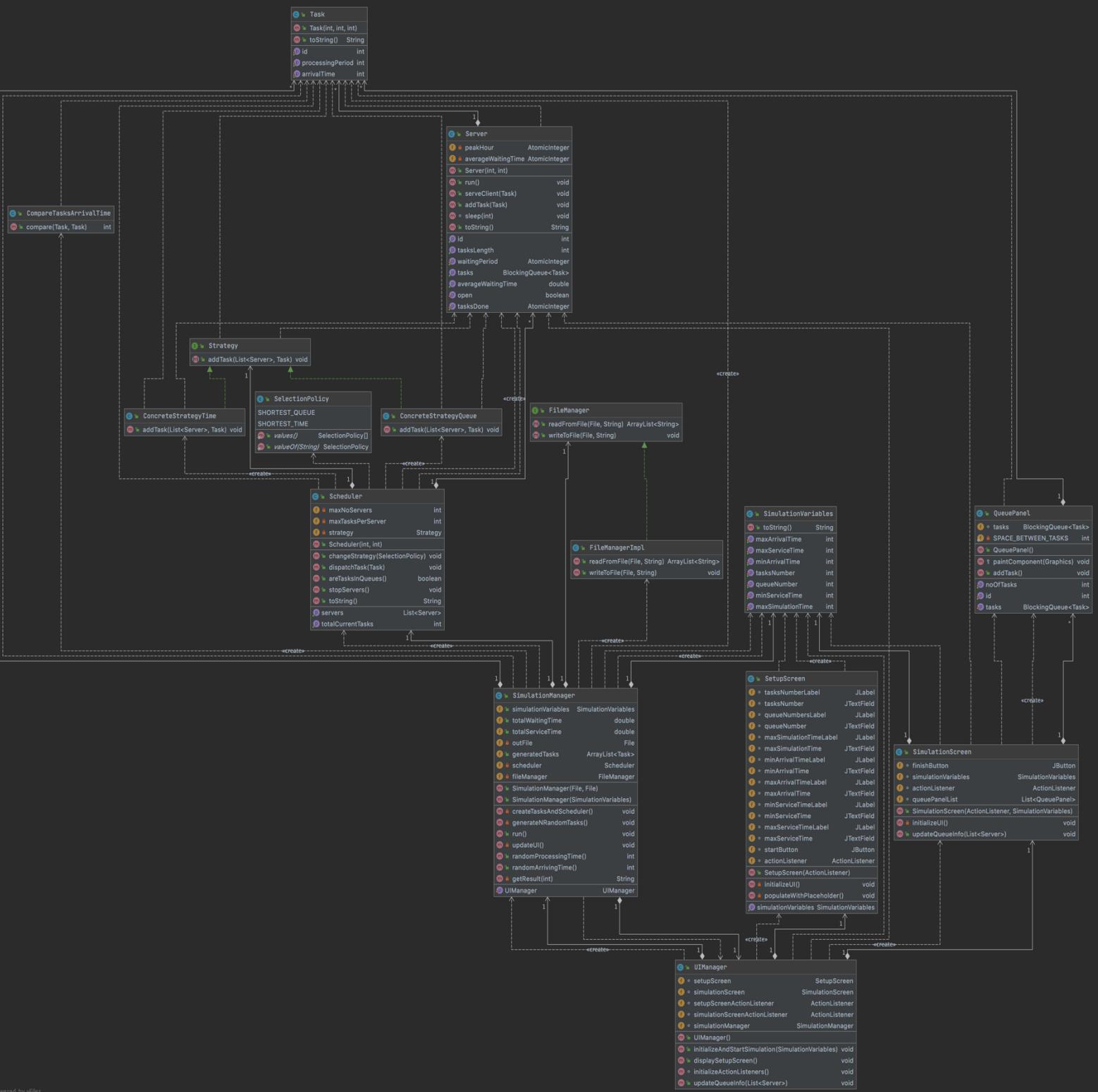
Interfața grafică (fereastra pentru setarea variabilelor de simulare)



Interfață grafică (fereastra de afișare a evoluției simulării)



Diagrama UML



IV. Implementare

Aplicația este structurată în diferite clase care au o funcționalitate foarte bine definită, conținând strict metodele de care aceste clase au nevoie pentru funcționarea corectă a aplicației finale.

1. Programul începe în clasa principală *Main* în care vom putea apela fie metoda pentru rularea programului cu citirea datelor dintr-un fișier de intare și afișarea evoluției cozilor într-un fișier de ieșire, fie metoda pentru rularea aplicației în interfață grafică.

```
1 import simulation.SimulationManager;
2 import ui.UIManager;
3
4 import java.io.File;
5
6 public class Main {
7     public static void main(String args[]) {
8         // startFileSimulation();
9         startUISimulation();
10    }
11
12    static void startFileSimulation() {
13        try {
14            File inputFile = new File("in-test-1.txt");
15            File outputFile = new File("out-test-1.txt");
16
17            SimulationManager gen = new SimulationManager(inputFile, outputFile);
18            Thread simThread = new Thread(gen);
19            simThread.start();
20        } catch (Exception ex) {
21            System.out.println(ex.getMessage());
22        }
23    }
24
25    static void startUISimulation() {
26        try {
27            UIManager uiManager = new UIManager();
28            uiManager.setVisible(true);
29        } catch (Exception ex) {
30            System.out.println(ex.getMessage());
31        }
32    }
33
34 }
```

Fisierele de intare/iesire

UIManager care se va acupa de afisarea elementelor intr-o fereastra

2. Clasa *SimulationManager* reprezintă creierul întregii aplicații. Acesta conține câte o instanță a fiecărei clase, adică știe despre fiecare clasă în parte. Această clasă conține câte un constructor pentru fiecare metodă de rulare a aplicației, fie varianta cu citirea din fișiere, fie varianta cu citirea variabilelor introduse de către utilizator în interfață grafică. Această clasă reprezentând creierul programului, implementează metodele:

- de creare a *scheduler-ului* și *task-urilor*, prin generarea aleatoare de n clienți și crearea *schedulerului* cu un număr de cozi și clienți specificat fie în fișier, fie de către utilizator în interfață. La generarea aleatoare de n clienți, se va genera în mod aleator timpul de la care clientul poate intra în coadă și timpul necesar servirii acestuia.

```
private void createTasksAndScheduler(){
    generateNRandomTasks();
    scheduler = new Scheduler(simulationVariables.getQueueNumber(), simulationVariables.getTasksNumber());
}

private void generateNRandomTasks() {
    generatedTasks = new ArrayList<>(simulationVariables.getTasksNumber());
    for (int i = 0; i < simulationVariables.getTasksNumber(); i++) {
        Task t = new Task(i, randomArrivingTime(), randomProcessingTime());
        generatedTasks.add(t);
    }
    // Sort chronologically for arrivingTime
    Collections.sort(generatedTasks, new CompareTasksArrivalTime());
}
```

```
public int randomProcessingTime() {
    return (simulationVariables.getMinServiceTime() + (int) (Math.random() * (simulationVariables.getMaxServiceTime() -
        simulationVariables.getMinServiceTime())));
}

public int randomArrivingTime() {
    return (simulationVariables.getMinArrivalTime() + (int) (Math.random() * (simulationVariables.getMaxArrivalTime() -
        simulationVariables.getMinArrivalTime())));
}
```

- metoda *run()* în care este implementată evoluția cozii în timp real. Condiția rulării simulării este:
 - ca timpul curent al simulării să fie mai mic decât timpul maxim de rulare al simulării, timp cunoscut încă de la începerea simulării.
 - și fie *scheduler-ul* mai are clienți în cozi, fie că încă mai sunt clienți așteaptă intrarea în coada optimă

În cazul în care această condiție este îndeplinită, cât timp încă mai sunt clienți care așteptă și timpul la care primul client trebuie să intre în coadă este egal cu timpul curent al simulării, *scheduler-u*l va distribui fiecare client la coada optimă în funcție de strategia de alegere a cozii. Astfel, numărul de clienți procesați va crește la fiecare distribuire, timpul total de servire va crește cu valoarea timpului de servire necesar fiecărui client. În final, clientul care a fost deja procesat, va fi eliminat.

Tot în această metodă se verifică dacă:

- există un fișier de ieșire și se scrie rezultatul simulării, în cazul în care condiția este *true* (*adevărată*)
- există *uiManager* (instanța clasei *UIManager* care creează interfața grafică). În cazul în care condiția este *adevărată*, se va deschide fereastra de interacțiune a aplicației cu utilizatorul.

Într-un bloc *try { } catch { }* vom *adormi (sleep)* firul de execuție și vom incrementa timpul curent al simulării.

```
while (currentTime.get() < simulationVariables.getMaxSimulationTime() &&
       (this.scheduler.areTasksInQueues() || !generatedTasks.isEmpty())) {
    System.out.println("--- Time: " + currentTime.get() + " --- Remaining: " + this.generatedTasks.size() + " ---");
    while (!generatedTasks.isEmpty() && generatedTasks.get(0).getArrivalTime() == currentTime.get()) {
        scheduler.dispatchTask(generatedTasks.get(0));
        nrProcessedClients++;
        totalServiceTime += generatedTasks.get(0).getProcessingPeriod();
        generatedTasks.remove(generatedTasks.get(0));
    }

    if(this.outFile != null) {
        String result = getResult(currentTime.get());
        fileManager.writeToFile(this.outFile, result);
    }

    if(this.scheduler.getTotalCurrentTasks() > maxTasksConcurrently){
        System.out.println(this.scheduler.getTotalCurrentTasks());
        maxTasksConcurrently = this.scheduler.getTotalCurrentTasks();
        peakHour = currentTime.get();
    }
}

if(this.uiManager != null){
    updateUI();
}

try {
    Thread.sleep( millis: 100 );
    currentTime.incrementAndGet();
} catch (InterruptedException e) {
    e.printStackTrace();
}
}
```

În finalul metodei vom calcula timpul mediu de servire per simulare și timpul la care cozile au avut cei mai mulți clienti și vom afișa datele în fișierul de ieșire.

```
if(this.outFile != null) {
    String formatting ="----- Results ----- \n";
    fileManager.writeToFile(this.outFile, formatting);
    String serviceData = String.format("Average service time: %s\n", (totalServiceTime / nrProcessedClients));
    fileManager.writeToFile(this.outFile, serviceData);
    String peakData = String.format("Peak hour: %s\n", peakHour);
    fileManager.writeToFile(this.outFile, peakData);
}
if(this.uiManager != null){
    updateUI();
}
this.scheduler.stopServers();
}
```

Actualizarea interfeței grafice se va face prin metoda *updateUI* care va modifica și va modifica în timp real evoluția cozilor.

3. Clasa *Scheduler* se ocupă de împărțirea clientilor (task-urilor) în cozi în funcție de strategia aleasă.

```
public void changeStrategy(SelectionPolicy policy) {  
    // apply strategy pattern to instantiate the strategy with the concrete strategy corresponding to policy  
    if (policy == SelectionPolicy.SHORTEST_QUEUE)  
        strategy = new ConcreteStrategyQueue();  
    else if (policy == SelectionPolicy.SHORTEST_TIME)  
        strategy = new ConcreteStrategyTime();  
}  
  
public void dispatchTask(Task t) {  
    strategy.addTask(servers, t);  
}
```

Cele două strategii sunt implementate în pachetul *strategy* care conține o interfață în care avem o metodă abstractă pentru adăugarea task-ului în funcție de strategie

```
public interface Strategy {  
    void addTask(List<Server> servers, Task t);  
}
```

și cele două clase reprezentând strategiile:

- în funcție de cea mai scurtă coadă (se verifică lungimea fiecărei cozi (server), primul client care urmează să intre în coadă va fi repartizat la coada cu cei mai puțini clienți)

```
public class ConcreteStrategyQueue implements Strategy {  
    public void addTask(List<Server> servers, Task t) {  
        int minQueue = 32000;  
        Server serverWithShortestQueue = servers.get(0);  
        for (Server server:servers) {  
            int serverQueueLength = server.getTasksLength();  
            if(serverQueueLength < minQueue){  
                minQueue = serverQueueLength;  
                serverWithShortestQueue = server;  
            }  
        }  
        serverWithShortestQueue.setOpen(true);  
        serverWithShortestQueue.addTask(t);  
    }  
}
```

- în funcție de timpul de așteptare cel mai scurt (se verifică timpul de așteptare total al fiecarei cozi (server), primul client care urmează să intre în coadă va fi repartizat la coada care urmează să se termine prima)

```

public class ConcreteStrategyTime implements Strategy {
    public void addTask(List<Server> servers, Task t) {
        long minTime = 320000;
        Server serverWithShortestWaitingTime = servers.get(0);
        for (Server server : servers) {
            int serverQueueWaitingTime = server.getWaitingPeriod().get();
            if (serverQueueWaitingTime < minTime) {
                minTime = serverQueueWaitingTime;
                serverWithShortestWaitingTime = server;
            }
        }
        serverWithShortestWaitingTime.setOpen(true);
        serverWithShortestWaitingTime.addTask(t);
    }
}

```

```

public enum SelectionPolicy {
    SHORTEST_QUEUE, SHORTEST_TIME
}

```

Clasa *Scheduler* trebuie să se ocupe de oprirea serverelor la finalul aplicației și conține o metodă în care calculează numărul total de clienți (task-uri) aflați în momentul curent al simulării, metodă folosită pentru calcularea *peak hour-ului* (în clasa *SimulationManager*).

```

public void stopServers(){
    for (Server server : servers){
        server.setOpen(false);
    }
}

public int getTotalCurrentTasks(){
    int totalTasks = 0;
    for (Server q : servers) {
        totalTasks += q.getTasksLength();
    }
    return totalTasks;
}

```

În constructorul clasei *Scheduler* se creează *noServers* instanțe de servere, fiecare având un ID și un număr maxim de clienți care pot intra în coadă. După adăugarea serverelor într-o listă de servere, vom porni threadul asociat fiecărui server (coadă).

```
public Scheduler(int noServers, int noTasks) {
    this.changeStrategy(SelectionPolicy.SHORTEST_TIME);
    this.maxNoServers = noServers;
    this.maxTasksPerServer = noTasks;
    this.servers = new ArrayList<>(noServers);
    for (int i = 0; i < noServers; i++) {
        Server server = new Server(i, maxTasksPerServer);
        servers.add(server);
        // Add to thread list
        Thread serverThread = new Thread(server);
        serverThread.start();
    }
}
```

Strategia aleasă în acest caz este cea bazată pe coada care se termină cel mai repede. (pentru a modifica strategia, vom înlocui *SHORTEST_TIME* cu *SHORTEST_QUEUE*)

4. Clasa *Server* se ocupă de tot ce ține de procesarea unui client. Constructorul clasei creează câte un server (coadă) formată dintr-un ID și un număr maxim de clienți per server și initializează toate valorile cu valoarea 0. Clasa implementează metoda *run()* care se ocupă de servirea fiecărui client, atât timp cât există clienți de servit.

La fiecare perioadă a rulării simulării, după ce un client a intrat în coadă, firul de executie va dormi *seconds * 100ms*, iar clientului trebuie să își decrementeze timpul de servire cu 1. În cazul în care timpul de servire a ajuns la 0, înseamnă că timpul necesar servirii clientului s-a terminat și acesta trebuie scos din coadă.

```
public void run() {
    while (isOpen) {
        Task nextTask = tasks.peek();
        if (nextTask != null) {
            serveClient(nextTask);
        }
    }
}

public void serveClient(Task client) {
    sleep(seconds: 1); // Red arrow points here
    client.setProcessingPeriod(client.getProcessingPeriod() - 1);

    if (client.getProcessingPeriod() <= 0) {
        tasks.remove(client);
    }
    System.out.println("Queue #" + this.getId() + " :" + this.tasks.toString());
    this.waitingPeriod.decrementAndGet();
}

void sleep(int seconds) {
    try {
        Thread.sleep(millis: 100 * seconds);
    } catch (InterruptedException e) {
        e.printStackTrace();
    }
}
```

La fiecare adăugare a unui task, timpul de așteptare total al cozii în care este adăugat task-ul va crește cu valoarea timpului necesar servirii acestuia.

```
public void addTask(Task newTask) {
    this.tasks.add(newTask);
    this.waitingPeriod.addAndGet(newTask.getProcessingPeriod());
}

//      this.averageWaitingTime = this.waitingPeriod / this.tasksDone.get()
```

Metoda *toString* din clasa *Server* se ocupă de afișarea în fișierul de ieșire a evoluției cozilor în timp real.

```
public String toString() {
    String result;
    if (tasks.peek() == null || tasks.peek().getProcessingPeriod() == 0) {
        result = "closed";
    } else {
        result = tasks.toString();
    }
    return result;
}
```

```
public String toString() {
    return "(" + id + "," + arrivalTime + "," + processingPeriod + ")";
}
```

Metodă implementată în clasa *Task* pentru afișarea sub forma (id, arrivalTime, processingPeriod) a fiecărui client (task)

5. Clasa *Task* se află în pachetul *model* deoarece task-ul (clientul) reprezintă un model, adică această clasă doar descrie atributele unui client și îi initializează valorile printr-un constructor.

```
public class Task {
    private int id;
    private int arrivalTime;
    private int processingPeriod; // t_service

    public Task(int id, int arrivalTime, int processingPeriod) {
        this.id = id;
        this.arrivalTime = arrivalTime;
        this.processingPeriod = processingPeriod;
    }
}
```

Afișarea simulării am implementat-o în două metode:

1. Prin citirea variabilelor dintr-un fișier de intrare și afișarea rezultatului simulării într-un fișier de ieșire
2. Prin citirea variabilelor introduse de către utilizator în interfață grafică și vizualizarea evoluției simulării tot într-o interfață grafică

1. Metoda utilizând fișiere

În pachetul *file* am implementat o interfață care conține declararea metodelor de citire și scriere din/în fișiere (*FileManager*). Iar în clasa *FileManagerImpl* am scris corpul acestor metode. În metoda *readFromFile* se citesc datele din fișier și se adaugă într-o listă.

```
public class FileManagerImpl implements FileManager {  
    @Override  
    public ArrayList<String> readFromFile(File file, String delimiter) throws IllegalArgumentException {  
        ArrayList<String> fileResult = new ArrayList<>();  
        try {  
            Scanner sc = new Scanner(file);  
            while (sc.hasNextLine()) {  
                String nextLine = sc.nextLine();  
                String[] splittedLine = nextLine.split( regex: "\\|");  
  
                if(splittedLine.length > 0){  
                    for (String value:splittedLine) {  
                        fileResult.add(value);  
                    }  
                }else{  
                    fileResult.add(nextLine);  
                }  
            }  
            sc.close();  
        } catch (IllegalArgumentException | FileNotFoundException ex) {  
            throw new IllegalArgumentException("File does not exists!");  
        }  
  
        return fileResult;  
    }  
}
```

Metoda *writeToFile* va scrie în fișier rezultatul simulării.

```
@Override  
public void writeToFile(File file, String data) throws IllegalArgumentException {  
    FileWriter fWriter;  
    try {  
        fWriter = new FileWriter(file.getAbsoluteFile(), append: true);  
        fWriter.write(data);  
        fWriter.close();  
    } catch (Exception ex) {  
        System.out.println("Error while trying to write to the file: \n" + ex.getMessage());  
    }  
}
```

2. Metoda utilizând interfață grafică (GUI)

În pachetul *ui* am declarat 4 clase:

- *UIManager*
- *SetupScreen*
- *SimulationScreen*
- *QueuePanel*

Clasa *UIManager* are un constructor în care dimensionăm fereastra de afișare, o facem vizibilă, inițializăm ascultătorii butoanelor și afișam fereastra de pornire în care utilizatorul introduce valorile pentru variabilele necesare simulării.

```
public UIManager() throws HeadlessException {
    setTitle("Simulation setup");
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    setPreferredSize(new Dimension(width: 900, height: 600));
    pack();
    setLocationRelativeTo(null);
    setVisible(true);

    initializeActionListeners();
    displaySetupScreen();
}
```

Aplicația se va deschide prin afișarea unei ferestre numite *Simulation setup* în care utilizatorul va trebui să introducă datele, iar după apăsarea butonului de pornire a simulării (*Start simulation*), se va deschide o altă fereastră în care se va putea vizualiza evoluția cozilor la fiecare moment de timp până la terminarea timpului maxim de simulare (dat de utilizator). Clasa de *UIManager* va gestiona atât evenimentele din fereastra de setare a variabilelor, cât și a "animației" cozilor, din fereastra de afișare a simulării.

```

public UIManager() throws HeadlessException {
    setTitle("Simulation setup");
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    setPreferredSize(new Dimension(width: 900, height: 600));
    pack();
    setLocationRelativeTo(null);
    setVisible(true);

    initializeActionListeners();
    displaySetupScreen();
}

public void initializeAndStartSimulation(SimulationVariables simulationVariables){
    setTitle("Simulation running...");
    this.simulationScreen = new SimulationScreen(simulationScreenActionListener, simulationVariables);
    this.remove(setupScreen);
    this.add(simulationScreen);
    this.revalidate();

    this.simulationManager = new SimulationManager(simulationVariables);
    this.simulationManager.setUIManager(this);
    Thread simThread = new Thread(simulationManager);
    simThread.start();
}

public void displaySetupScreen(){
    setTitle("Simulation setup");
    this.setupScreen = new SetupScreen(setupScreenActionListener);
    if(simulationScreen != null){
        this.remove(simulationScreen);
    }
    this.add(setupScreen);
    this.revalidate();
}

```

1. Fereastra de setup (*SetupScreen*)

Are un constructor care initializează toate componentele swing. By default, am setat ca în momentul rulării simulării cu interfața grafică, să se populeze câmpurile cu primele valori menționate în suportul pentru temă. Valorile de simulare se vor extrage din câmpuri prin metoda *getSimulationVariables*.

```

public SimulationVariables getSimulationVariables() {
    SimulationVariables simulationVariables = new SimulationVariables();
    simulationVariables.setTasksNumber(Integer.parseInt(tasksNumber.getText()));
    simulationVariables.setQueueNumber(Integer.parseInt(queueNumber.getText()));
    simulationVariables.setMaxSimulationTime(Integer.parseInt(maxSimulationTime.getText()));
    simulationVariables.setMinArrivalTime(Integer.parseInt(minArrivalTime.getText()));
    simulationVariables.setMaxArrivalTime(Integer.parseInt(maxArrivalTime.getText()));
    simulationVariables.setMinServiceTime(Integer.parseInt(minServiceTime.getText()));
    simulationVariables.setMaxServiceTime(Integer.parseInt(maxServiceTime.getText()));
    return simulationVariables;
}

```

Clasa *SimulationVariables* reprezintă și ea doar un model, deoarece ea conține doar variabilele necesare simulării și metodele mutatoare și accesoare pentru acestea.

2. Fereastra de simulare (*SimulationScreen*)

După introducerea datelor în field-uri și apăsarea butonului de pornire a simulării, se va deschide fereastra în care va porni animația simulării cozilor.

Fiecare coadă va reprezenta o linie în simulare, formată dintr-un dreptunghi gri (reprezentând coada) și buline portocalii (reprezentând task-urile). Am creat o clasă specială, clasa *QueuePanel*, care se va ocupa doar de desenarea unei astfel de linii. În funcție de care este numărul de cozi introdus de user, atâtea linii se vor crea. Fiecare linie va avea un *id* și un număr de task-uri. Metoda *paintComponent* va crea un dreptunghi, iar după acesta se vor desena pe ecran

```
@Override  
protected void paintComponent(Graphics g) {  
    super.paintComponent(g); // do your superclass's painting routine first, and then paint on top of it.  
    g.setColor(Color.DARK_GRAY);  
    g.fillRect( x: 0, y: 0, width: 25, height: 50 );  
  
    for(int i = 0; i < this.tasks.size(); i++){  
        g.setColor(Color.ORANGE);  
        g.fillOval( x: 50 + SPACE_BETWEEN_TASKS * i, y: 18, width: 15, height: 15 );  
    }  
}
```

atâtea cercuri câte task-uri sunt la momentul curent al simulării.

Actualizarea stării fiecarei cozi o va face *SimulationManager-ul* prin metoda de actualizare a datelor cozii.

```
public void updateQueueInfo(List<Server> servers) {  
    for (int i = 0; i < queuePanelList.size(); i++) {  
        queuePanelList.get(i).setTasks(servers.get(i).getTasks());  
    }  
}
```

După terminarea simlării, se poate reveni la fereastra de setare a variabilelor prin apăsarea butonului *Finish simulation*. Astfel utilizatorul va putea introduce alte date și simula din nou cu datele respective. Terminarea și inchiderea aplicației se va face prin apăsarea butonului de *exit* al ferestrei.

V. Rezultate

Rezultatele simulării care s-au scris în fișierul de ieșire pentru cele 3 cazuri cerute în cerința temei:

Test 1	Test 2	Test 3
$N = 4$ $Q = 2$ $t_{simulation}^{MAX} = 60 \text{ seconds}$ $[t_{arrival}^{MIN}, t_{arrival}^{MAX}] = [2, 30]$ $[t_{service}^{MIN}, t_{service}^{MAX}] = [2, 4]$	$N = 50$ $Q = 5$ $t_{simulation}^{MAX} = 60 \text{ seconds}$ $[t_{arrival}^{MIN}, t_{arrival}^{MAX}] = [2, 40]$ $[t_{service}^{MIN}, t_{service}^{MAX}] = [1, 7]$	$N = 1000$ $Q = 20$ $t_{simulation}^{MAX} = 200 \text{ seconds}$ $[t_{arrival}^{MIN}, t_{arrival}^{MAX}] = [10, 100]$ $[t_{service}^{MIN}, t_{service}^{MAX}] = [3, 9]$

Test 1:

```

Time: 0
Waiting clients: (0,9,2) (1,13,2) (2,14,2) (3,20,2)
Queue 0: closed
Queue 1: closed

Time: 1
Waiting clients: (0,9,2) (1,13,2) (2,14,2) (3,20,2)
Queue 0: closed
Queue 1: closed

Time: 2
Waiting clients: (0,9,2) (1,13,2) (2,14,2) (3,20,2)
Queue 0: closed
Queue 1: closed

Time: 3
Waiting clients: (0,9,2) (1,13,2) (2,14,2) (3,20,2)
Queue 0: closed
Queue 1: closed

Time: 4
Waiting clients: (0,9,2) (1,13,2) (2,14,2) (3,20,2)
Queue 0: closed
Queue 1: closed

Time: 5
Waiting clients: (0,9,2) (1,13,2) (2,14,2) (3,20,2)
Queue 0: closed
Queue 1: closed

Time: 6
Waiting clients: (0,9,2) (1,13,2) (2,14,2) (3,20,2)
Queue 0: closed
Queue 1: closed

Time: 7
Waiting clients: (0,9,2) (1,13,2) (2,14,2) (3,20,2)
Queue 0: closed
Queue 1: closed

Time: 8
Waiting clients: (0,9,2) (1,13,2) (2,14,2) (3,20,2)

```

```
Queue 0: closed
Queue 1: closed

Time: 9
Waiting clients: (1,13,2) (2,14,2) (3,20,2)
Queue 0: [(0,9,2)]
Queue 1: closed

Time: 10
Waiting clients: (1,13,2) (2,14,2) (3,20,2)
Queue 0: [(0,9,2)]
Queue 1: closed

Time: 11
Waiting clients: (1,13,2) (2,14,2) (3,20,2)
Queue 0: [(0,9,1)]
Queue 1: closed

Time: 12
Waiting clients: (1,13,2) (2,14,2) (3,20,2)
Queue 0: closed
Queue 1: closed

Time: 13
Waiting clients: (2,14,2) (3,20,2)
Queue 0: [(1,13,2)]
Queue 1: closed

Time: 14
Waiting clients: (3,20,2)
Queue 0: [(1,13,2)]
Queue 1: [(2,14,2)]

Time: 15
Waiting clients: (3,20,2)
Queue 0: [(1,13,1)]
Queue 1: [(2,14,2)]

Time: 16
Waiting clients: (3,20,2)
Queue 0: closed
Queue 1: closed

Time: 17
Waiting clients: (3,20,2)
Queue 0: closed
Queue 1: closed

Time: 18
Waiting clients: (3,20,2)
Queue 0: closed
Queue 1: closed

Time: 19
Waiting clients: (3,20,2)
Queue 0: closed
Queue 1: closed

Time: 20
Waiting clients:
Queue 0: [(3,20,2)]
```

```

Queue 1: closed

Time: 21
Waiting clients:
Queue 0: [(3,20,1)]
Queue 1: closed

----- Results -----
Average service time: 2.0
Peak hour: 14

```

Test 2:

```

Time: 0
Waiting clients:
(10,3,3) (33,3,6) (28,4,2) (3,5,4) (12,5,4) (2,6,4) (15,6,1) (23,6,2) (45,6,5) (49,6
,6) (6,7,6) (48,7,6) (17,8,5) (5,11,4) (8,12,5) (34,13,3) (41,13,1) (35,15,4) (19,16
,6) (30,16,6) (38,17,3) (43,17,3) (36,19,5) (37,19,6) (9,20,6) (13,20,5) (25,21,3) (
42,21,3) (7,22,6) (22,23,1) (46,23,3) (27,24,5) (29,24,5) (4,26,1) (14,26,5) (16,28
,3) (31,29,1) (32,29,2) (26,30,2) (0,31,6) (47,31,5) (20,34,1) (39,35,4) (18,37,4) (
11,38,4) (24,38,3) (44,38,2) (1,39,2) (21,39,3) (40,39,4)
Queue 0: closed
Queue 1: closed
Queue 2: closed
Queue 3: closed
Queue 4: closed

Time: 1
Waiting clients:
(10,3,3) (33,3,6) (28,4,2) (3,5,4) (12,5,4) (2,6,4) (15,6,1) (23,6,2) (45,6,5) (49,6
,6) (6,7,6) (48,7,6) (17,8,5) (5,11,4) (8,12,5) (34,13,3) (41,13,1) (35,15,4) (19,16
,6) (30,16,6) (38,17,3) (43,17,3) (36,19,5) (37,19,6) (9,20,6) (13,20,5) (25,21,3) (
42,21,3) (7,22,6) (22,23,1) (46,23,3) (27,24,5) (29,24,5) (4,26,1) (14,26,5) (16,28
,3) (31,29,1) (32,29,2) (26,30,2) (0,31,6) (47,31,5) (20,34,1) (39,35,4) (18,37,4) (
11,38,4) (24,38,3) (44,38,2) (1,39,2) (21,39,3) (40,39,4)
Queue 0: closed
Queue 1: closed
Queue 2: closed
Queue 3: closed
Queue 4: closed

Time: 2
Waiting clients:
(10,3,3) (33,3,6) (28,4,2) (3,5,4) (12,5,4) (2,6,4) (15,6,1) (23,6,2) (45,6,5) (49,6
,6) (6,7,6) (48,7,6) (17,8,5) (5,11,4) (8,12,5) (34,13,3) (41,13,1) (35,15,4) (19,16
,6) (30,16,6) (38,17,3) (43,17,3) (36,19,5) (37,19,6) (9,20,6) (13,20,5) (25,21,3) (
42,21,3) (7,22,6) (22,23,1) (46,23,3) (27,24,5) (29,24,5) (4,26,1) (14,26,5) (16,28
,3) (31,29,1) (32,29,2) (26,30,2) (0,31,6) (47,31,5) (20,34,1) (39,35,4) (18,37,4) (
11,38,4) (24,38,3) (44,38,2) (1,39,2) (21,39,3) (40,39,4)
Queue 0: closed
Queue 1: closed
Queue 2: closed
Queue 3: closed
Queue 4: closed

Time: 3
Waiting clients:
(28,4,2) (3,5,4) (12,5,4) (2,6,4) (15,6,1) (23,6,2) (45,6,5) (49,6,6) (6,7,6) (48,7
,6) (17,8,5) (5,11,4) (8,12,5) (34,13,3) (41,13,1) (35,15,4) (19,16,6) (30,16,6) (38,

```

```
17,3) (43,17,3) (36,19,5) (37,19,6) (9,20,6) (13,20,5) (25,21,3) (42,21,3) (7,22,6)
(22,23,1) (46,23,3) (27,24,5) (29,24,5) (4,26,1) (14,26,5) (16,28,3) (31,29,1) (32,
29,2) (26,30,2) (0,31,6) (47,31,5) (20,34,1) (39,35,4) (18,37,4) (11,38,4) (24,38,3
) (44,38,2) (1,39,2) (21,39,3) (40,39,4)
Queue 0: [(10,3,3)]
Queue 1: [(33,3,6)]
Queue 2: closed
Queue 3: closed
Queue 4: closed

Time: 4
Waiting clients:
(3,5,4) (12,5,4) (2,6,4) (15,6,1) (23,6,2) (45,6,5) (49,6,6) (6,7,6) (48,7,6) (17,8,
5) (5,11,4) (8,12,5) (34,13,3) (41,13,1) (35,15,4) (19,16,6) (30,16,6) (38,17,3) (43
,17,3) (36,19,5) (37,19,6) (9,20,6) (13,20,5) (25,21,3) (42,21,3) (7,22,6) (22,23,1
) (46,23,3) (27,24,5) (29,24,5) (4,26,1) (14,26,5) (16,28,3) (31,29,1) (32,29,2) (26
,30,2) (0,31,6) (47,31,5) (20,34,1) (39,35,4) (18,37,4) (11,38,4) (24,38,3) (44,38,
2) (1,39,2) (21,39,3) (40,39,4)
Queue 0: [(10,3,2)]
Queue 1: [(33,3,5)]
Queue 2: [(28,4,2)]
Queue 3: closed
Queue 4: closed

Time: 5
Waiting clients:
(2,6,4) (15,6,1) (23,6,2) (45,6,5) (49,6,6) (6,7,6) (48,7,6) (17,8,5) (5,11,4) (8,12
,5) (34,13,3) (41,13,1) (35,15,4) (19,16,6) (30,16,6) (38,17,3) (43,17,3) (36,19,5)
(37,19,6) (9,20,6) (13,20,5) (25,21,3) (42,21,3) (7,22,6) (22,23,1) (46,23,3) (27,2
4,5) (29,24,5) (4,26,1) (14,26,5) (16,28,3) (31,29,1) (32,29,2) (26,30,2) (0,31,6) (
47,31,5) (20,34,1) (39,35,4) (18,37,4) (11,38,4) (24,38,3) (44,38,2) (1,39,2) (21,3
9,3) (40,39,4)
Queue 0: [(10,3,1)]
Queue 1: [(33,3,4)]
Queue 2: [(28,4,1)]
Queue 3: [(3,5,4)]
Queue 4: [(12,5,4)]

Time: 6
Waiting clients:
(6,7,6) (48,7,6) (17,8,5) (5,11,4) (8,12,5) (34,13,3) (41,13,1) (35,15,4) (19,16,6)
(30,16,6) (38,17,3) (43,17,3) (36,19,5) (37,19,6) (9,20,6) (13,20,5) (25,21,3) (42,
21,3) (7,22,6) (22,23,1) (46,23,3) (27,24,5) (29,24,5) (4,26,1) (14,26,5) (16,28,3)
(31,29,1) (32,29,2) (26,30,2) (0,31,6) (47,31,5) (20,34,1) (39,35,4) (18,37,4) (11,
38,4) (24,38,3) (44,38,2) (1,39,2) (21,39,3) (40,39,4)
Queue 0: [(2,6,4)]
Queue 1: [(33,3,3), (45,6,5)]
Queue 2: [(15,6,1), (23,6,2), (49,6,6)]
Queue 3: [(3,5,3)]
Queue 4: [(12,5,3)]

Time: 7
Waiting clients:
(17,8,5) (5,11,4) (8,12,5) (34,13,3) (41,13,1) (35,15,4) (19,16,6) (30,16,6) (38,17
,3) (43,17,3) (36,19,5) (37,19,6) (9,20,6) (13,20,5) (25,21,3) (42,21,3) (7,22,6) (2
,23,1) (46,23,3) (27,24,5) (29,24,5) (4,26,1) (14,26,5) (16,28,3) (31,29,1) (32,29
,2) (26,30,2) (0,31,6) (47,31,5) (20,34,1) (39,35,4) (18,37,4) (11,38,4) (24,38,3) (
44,38,2) (1,39,2) (21,39,3) (40,39,4)
Queue 0: [(2,6,4)]
Queue 1: [(33,3,2), (45,6,5)]
Queue 2: [(23,6,2), (49,6,6)]
```

```

Queue 3: [(3,5,2), (6,7,6)]
Queue 4: [(12,5,3), (48,7,6)]

Time: 8
Waiting clients:
(5,11,4) (8,12,5) (34,13,3) (41,13,1) (35,15,4) (19,16,6) (30,16,6) (38,17,3) (43,1
7,3) (36,19,5) (37,19,6) (9,20,6) (13,20,5) (25,21,3) (42,21,3) (7,22,6) (22,23,1) (
46,23,3) (27,24,5) (29,24,5) (4,26,1) (14,26,5) (16,28,3) (31,29,1) (32,29,2) (26,3
0,2) (0,31,6) (47,31,5) (20,34,1) (39,35,4) (18,37,4) (11,38,4) (24,38,3) (44,38,2)
(1,39,2) (21,39,3) (40,39,4)
Queue 0: [(2,6,2), (17,8,5)]
Queue 1: [(33,3,1), (45,6,5)]
Queue 2: [(23,6,1), (49,6,6)]
Queue 3: [(3,5,1), (6,7,6)]
Queue 4: [(12,5,1), (48,7,6)]

Time: 9
Waiting clients:
(5,11,4) (8,12,5) (34,13,3) (41,13,1) (35,15,4) (19,16,6) (30,16,6) (38,17,3) (43,1
7,3) (36,19,5) (37,19,6) (9,20,6) (13,20,5) (25,21,3) (42,21,3) (7,22,6) (22,23,1) (
46,23,3) (27,24,5) (29,24,5) (4,26,1) (14,26,5) (16,28,3) (31,29,1) (32,29,2) (26,3
0,2) (0,31,6) (47,31,5) (20,34,1) (39,35,4) (18,37,4) (11,38,4) (24,38,3) (44,38,2)
(1,39,2) (21,39,3) (40,39,4)
Queue 0: [(2,6,1), (17,8,5)]
Queue 1: [(45,6,5)]
Queue 2: [(49,6,6)]
Queue 3: [(6,7,6)]
Queue 4: [(48,7,6)]

Time: 10
Waiting clients:
(5,11,4) (8,12,5) (34,13,3) (41,13,1) (35,15,4) (19,16,6) (30,16,6) (38,17,3) (43,1
7,3) (36,19,5) (37,19,6) (9,20,6) (13,20,5) (25,21,3) (42,21,3) (7,22,6) (22,23,1) (
46,23,3) (27,24,5) (29,24,5) (4,26,1) (14,26,5) (16,28,3) (31,29,1) (32,29,2) (26,3
0,2) (0,31,6) (47,31,5) (20,34,1) (39,35,4) (18,37,4) (11,38,4) (24,38,3) (44,38,2)
(1,39,2) (21,39,3) (40,39,4)
Queue 0: [(17,8,5)]
Queue 1: [(45,6,4)]
Queue 2: [(49,6,5)]
Queue 3: [(6,7,5)]
Queue 4: [(48,7,5)]

Time: 11
Waiting clients:
(8,12,5) (34,13,3) (41,13,1) (35,15,4) (19,16,6) (30,16,6) (38,17,3) (43,17,3) (36,
19,5) (37,19,6) (9,20,6) (13,20,5) (25,21,3) (42,21,3) (7,22,6) (22,23,1) (46,23,3)
(27,24,5) (29,24,5) (4,26,1) (14,26,5) (16,28,3) (31,29,1) (32,29,2) (26,30,2) (0,3
1,6) (47,31,5) (20,34,1) (39,35,4) (18,37,4) (11,38,4) (24,38,3) (44,38,2) (1,39,2)
(21,39,3) (40,39,4)
Queue 0: [(17,8,4)]
Queue 1: [(45,6,3), (5,11,4)]
Queue 2: [(49,6,4)]
Queue 3: [(6,7,4)]
Queue 4: [(48,7,4)]

Time: 12
Waiting clients:
(34,13,3) (41,13,1) (35,15,4) (19,16,6) (30,16,6) (38,17,3) (43,17,3) (36,19,5) (37
,19,6) (9,20,6) (13,20,5) (25,21,3) (42,21,3) (7,22,6) (22,23,1) (46,23,3) (27,24,5)
(29,24,5) (4,26,1) (14,26,5) (16,28,3) (31,29,1) (32,29,2) (26,30,2) (0,31,6) (47,
31,5) (20,34,1) (39,35,4) (18,37,4) (11,38,4) (24,38,3) (44,38,2) (1,39,2) (21,39,3)

```

```

) (40,39,4)
Queue 0: [(17,8,3), (8,12,5)]
Queue 1: [(45,6,2), (5,11,4)]
Queue 2: [(49,6,3)]
Queue 3: [(6,7,3)]
Queue 4: [(48,7,3)]

Time: 13
Waiting clients:
(35,15,4) (19,16,6) (30,16,6) (38,17,3) (43,17,3) (36,19,5) (37,19,6) (9,20,6) (13,
20,5) (25,21,3) (42,21,3) (7,22,6) (22,23,1) (46,23,3) (27,24,5) (29,24,5) (4,26,1)
(14,26,5) (16,28,3) (31,29,1) (32,29,2) (26,30,2) (0,31,6) (47,31,5) (20,34,1) (39,
35,4) (18,37,4) (11,38,4) (24,38,3) (44,38,2) (1,39,2) (21,39,3) (40,39,4)
Queue 0: [(17,8,2), (8,12,5)]
Queue 1: [(45,6,1), (5,11,4)]
Queue 2: [(49,6,2), (34,13,3)]
Queue 3: [(6,7,2), (41,13,1)]
Queue 4: [(48,7,2)]

Time: 14
Waiting clients:
(35,15,4) (19,16,6) (30,16,6) (38,17,3) (43,17,3) (36,19,5) (37,19,6) (9,20,6) (13,
20,5) (25,21,3) (42,21,3) (7,22,6) (22,23,1) (46,23,3) (27,24,5) (29,24,5) (4,26,1)
(14,26,5) (16,28,3) (31,29,1) (32,29,2) (26,30,2) (0,31,6) (47,31,5) (20,34,1) (39,
35,4) (18,37,4) (11,38,4) (24,38,3) (44,38,2) (1,39,2) (21,39,3) (40,39,4)
Queue 0: [(17,8,1), (8,12,5)]
Queue 1: [(5,11,4)]
Queue 2: [(49,6,1), (34,13,3)]
Queue 3: [(6,7,1), (41,13,1)]
Queue 4: [(48,7,1)]

Time: 15
Waiting clients:
(19,16,6) (30,16,6) (38,17,3) (43,17,3) (36,19,5) (37,19,6) (9,20,6) (13,20,5) (25,
21,3) (42,21,3) (7,22,6) (22,23,1) (46,23,3) (27,24,5) (29,24,5) (4,26,1) (14,26,5)
(16,28,3) (31,29,1) (32,29,2) (26,30,2) (0,31,6) (47,31,5) (20,34,1) (39,35,4) (18,
37,4) (11,38,4) (24,38,3) (44,38,2) (1,39,2) (21,39,3) (40,39,4)
Queue 0: [(8,12,5)]
Queue 1: [(5,11,3)]
Queue 2: [(34,13,3)]
Queue 3: [(41,13,1)]
Queue 4: [(35,15,4)]

Time: 16
Waiting clients:
(38,17,3) (43,17,3) (36,19,5) (37,19,6) (9,20,6) (13,20,5) (25,21,3) (42,21,3) (7,2
2,6) (22,23,1) (46,23,3) (27,24,5) (29,24,5) (4,26,1) (14,26,5) (16,28,3) (31,29,1)
(32,29,2) (26,30,2) (0,31,6) (47,31,5) (20,34,1) (39,35,4) (18,37,4) (11,38,4) (24,
38,3) (44,38,2) (1,39,2) (21,39,3) (40,39,4)
Queue 0: [(8,12,4)]
Queue 1: [(5,11,2), (30,16,6)]
Queue 2: [(34,13,2)]
Queue 3: [(19,16,6)]
Queue 4: [(35,15,3)]

Time: 17
Waiting clients:
(36,19,5) (37,19,6) (9,20,6) (13,20,5) (25,21,3) (42,21,3) (7,22,6) (22,23,1) (46,2
3,3) (27,24,5) (29,24,5) (4,26,1) (14,26,5) (16,28,3) (31,29,1) (32,29,2) (26,30,2)
(0,31,6) (47,31,5) (20,34,1) (39,35,4) (18,37,4) (11,38,4) (24,38,3) (44,38,2) (1,3
9,2) (21,39,3) (40,39,4)

```

```

Queue 0: [(8,12,3)]
Queue 1: [(5,11,1), (30,16,6)]
Queue 2: [(34,13,1), (38,17,3)]
Queue 3: [(19,16,5)]
Queue 4: [(35,15,2), (43,17,3)]

Time: 18
Waiting clients:
(36,19,5) (37,19,6) (9,20,6) (13,20,5) (25,21,3) (42,21,3) (7,22,6) (22,23,1) (46,2
3,3) (27,24,5) (29,24,5) (4,26,1) (14,26,5) (16,28,3) (31,29,1) (32,29,2) (26,30,2)
(0,31,6) (47,31,5) (20,34,1) (39,35,4) (18,37,4) (11,38,4) (24,38,3) (44,38,2) (1,3
9,2) (21,39,3) (40,39,4)
Queue 0: [(8,12,2)]
Queue 1: [(30,16,6)]
Queue 2: [(38,17,3)]
Queue 3: [(19,16,4)]
Queue 4: [(35,15,1), (43,17,3)]

Time: 19
Waiting clients:
(9,20,6) (13,20,5) (25,21,3) (42,21,3) (7,22,6) (22,23,1) (46,23,3) (27,24,5) (29,2
4,5) (4,26,1) (14,26,5) (16,28,3) (31,29,1) (32,29,2) (26,30,2) (0,31,6) (47,31,5) (
20,34,1) (39,35,4) (18,37,4) (11,38,4) (24,38,3) (44,38,2) (1,39,2) (21,39,3) (40,3
9,4)
Queue 0: [(8,12,1), (36,19,5)]
Queue 1: [(30,16,5)]
Queue 2: [(38,17,2), (37,19,6)]
Queue 3: [(19,16,3)]
Queue 4: [(43,17,3)]

Time: 20
Waiting clients:
(25,21,3) (42,21,3) (7,22,6) (22,23,1) (46,23,3) (27,24,5) (29,24,5) (4,26,1) (14,2
6,5) (16,28,3) (31,29,1) (32,29,2) (26,30,2) (0,31,6) (47,31,5) (20,34,1) (39,35,4)
(18,37,4) (11,38,4) (24,38,3) (44,38,2) (1,39,2) (21,39,3) (40,39,4)
Queue 0: [(36,19,5)]
Queue 1: [(30,16,4)]
Queue 2: [(38,17,1), (37,19,6)]
Queue 3: [(19,16,2), (9,20,6)]
Queue 4: [(43,17,2), (13,20,5)]

Time: 21
Waiting clients:
(7,22,6) (22,23,1) (46,23,3) (27,24,5) (29,24,5) (4,26,1) (14,26,5) (16,28,3) (31,2
9,1) (32,29,2) (26,30,2) (0,31,6) (47,31,5) (20,34,1) (39,35,4) (18,37,4) (11,38,4)
(24,38,3) (44,38,2) (1,39,2) (21,39,3) (40,39,4)
Queue 0: [(36,19,4), (42,21,3)]
Queue 1: [(30,16,3), (25,21,3)]
Queue 2: [(37,19,6)]
Queue 3: [(19,16,1), (9,20,6)]
Queue 4: [(43,17,1), (13,20,5)]

Time: 22
Waiting clients:
(22,23,1) (46,23,3) (27,24,5) (29,24,5) (4,26,1) (14,26,5) (16,28,3) (31,29,1) (32,
29,2) (26,30,2) (0,31,6) (47,31,5) (20,34,1) (39,35,4) (18,37,4) (11,38,4) (24,38,3)
(44,38,2) (1,39,2) (21,39,3) (40,39,4)
Queue 0: [(36,19,3), (42,21,3)]
Queue 1: [(30,16,2), (25,21,3), (7,22,6)]
Queue 2: [(37,19,5)]
Queue 3: [(9,20,6)]

```

```
Queue 4: [(13,20,5)]  
  
Time: 23  
Waiting clients:  
(27,24,5) (29,24,5) (4,26,1) (14,26,5) (16,28,3) (31,29,1) (32,29,2) (26,30,2) (0,31,6)  
(47,31,5) (20,34,1) (39,35,4) (18,37,4) (11,38,4) (24,38,3) (44,38,2) (1,39,2)  
(21,39,3) (40,39,4)  
Queue 0: [(36,19,2), (42,21,3)]  
Queue 1: [(30,16,1), (25,21,3), (7,22,6)]  
Queue 2: [(37,19,4), (22,23,1)]  
Queue 3: [(9,20,5)]  
Queue 4: [(13,20,4), (46,23,3)]  
  
Time: 24  
Waiting clients:  
(4,26,1) (14,26,5) (16,28,3) (31,29,1) (32,29,2) (26,30,2) (0,31,6) (47,31,5) (20,34,1)  
(39,35,4) (18,37,4) (11,38,4) (24,38,3) (44,38,2) (1,39,2) (21,39,3) (40,39,4)  
Queue 0: [(36,19,1), (42,21,3), (27,24,5)]  
Queue 1: [(25,21,3), (7,22,6)]  
Queue 2: [(37,19,3), (22,23,1), (29,24,5)]  
Queue 3: [(9,20,4)]  
Queue 4: [(13,20,3), (46,23,3)]  
  
Time: 25  
Waiting clients:  
(4,26,1) (14,26,5) (16,28,3) (31,29,1) (32,29,2) (26,30,2) (0,31,6) (47,31,5) (20,34,1)  
(39,35,4) (18,37,4) (11,38,4) (24,38,3) (44,38,2) (1,39,2) (21,39,3) (40,39,4)  
Queue 0: [(42,21,3), (27,24,5)]  
Queue 1: [(25,21,2), (7,22,6)]  
Queue 2: [(37,19,2), (22,23,1), (29,24,5)]  
Queue 3: [(9,20,3)]  
Queue 4: [(13,20,2), (46,23,3)]  
  
Time: 26  
Waiting clients:  
(16,28,3) (31,29,1) (32,29,2) (26,30,2) (0,31,6) (47,31,5) (20,34,1) (39,35,4) (18,37,4)  
(11,38,4) (24,38,3) (44,38,2) (1,39,2) (21,39,3) (40,39,4)  
Queue 0: [(42,21,2), (27,24,5)]  
Queue 1: [(25,21,1), (7,22,6)]  
Queue 2: [(37,19,1), (22,23,1), (29,24,5)]  
Queue 3: [(9,20,2), (4,26,1), (14,26,5)]  
Queue 4: [(13,20,1), (46,23,3)]  
  
Time: 27  
Waiting clients:  
(16,28,3) (31,29,1) (32,29,2) (26,30,2) (0,31,6) (47,31,5) (20,34,1) (39,35,4) (18,37,4)  
(11,38,4) (24,38,3) (44,38,2) (1,39,2) (21,39,3) (40,39,4)  
Queue 0: [(42,21,1), (27,24,5)]  
Queue 1: [(7,22,6)]  
Queue 2: [(22,23,1), (29,24,5)]  
Queue 3: [(9,20,1), (4,26,1), (14,26,5)]  
Queue 4: [(46,23,3)]  
  
Time: 28  
Waiting clients:  
(31,29,1) (32,29,2) (26,30,2) (0,31,6) (47,31,5) (20,34,1) (39,35,4) (18,37,4) (11,38,4)  
(24,38,3) (44,38,2) (1,39,2) (21,39,3) (40,39,4)  
Queue 0: [(27,24,5)]  
Queue 1: [(7,22,5)]  
Queue 2: [(29,24,5)]  
Queue 3: [(4,26,1), (14,26,5)]
```

```
Queue 4: [(46,23,2), (16,28,3)]  
  
Time: 29  
Waiting clients:  
(26,30,2) (0,31,6) (47,31,5) (20,34,1) (39,35,4) (18,37,4) (11,38,4) (24,38,3) (44,  
38,2) (1,39,2) (21,39,3) (40,39,4)  
Queue 0: [(27,24,4), (31,29,1)]  
Queue 1: [(7,22,4), (32,29,2)]  
Queue 2: [(29,24,4)]  
Queue 3: [(14,26,5)]  
Queue 4: [(46,23,1), (16,28,3)]  
  
Time: 30  
Waiting clients:  
(0,31,6) (47,31,5) (20,34,1) (39,35,4) (18,37,4) (11,38,4) (24,38,3) (44,38,2) (1,3  
9,2) (21,39,3) (40,39,4)  
Queue 0: [(27,24,3), (31,29,1)]  
Queue 1: [(7,22,3), (32,29,2)]  
Queue 2: [(29,24,3), (26,30,2)]  
Queue 3: [(14,26,4)]  
Queue 4: [(16,28,3)]  
  
Time: 31  
Waiting clients:  
(20,34,1) (39,35,4) (18,37,4) (11,38,4) (24,38,3) (44,38,2) (1,39,2) (21,39,3) (40,  
39,4)  
Queue 0: [(27,24,2), (31,29,1), (47,31,5)]  
Queue 1: [(7,22,2), (32,29,2)]  
Queue 2: [(29,24,2), (26,30,2)]  
Queue 3: [(14,26,3)]  
Queue 4: [(16,28,2), (0,31,6)]  
  
Time: 32  
Waiting clients:  
(20,34,1) (39,35,4) (18,37,4) (11,38,4) (24,38,3) (44,38,2) (1,39,2) (21,39,3) (40,  
39,4)  
Queue 0: [(27,24,1), (31,29,1), (47,31,5)]  
Queue 1: [(7,22,1), (32,29,2)]  
Queue 2: [(29,24,1), (26,30,2)]  
Queue 3: [(14,26,2)]  
Queue 4: [(16,28,1), (0,31,6)]  
  
Time: 33  
Waiting clients:  
(20,34,1) (39,35,4) (18,37,4) (11,38,4) (24,38,3) (44,38,2) (1,39,2) (21,39,3) (40,  
39,4)  
Queue 0: [(31,29,1), (47,31,5)]  
Queue 1: [(32,29,2)]  
Queue 2: [(26,30,2)]  
Queue 3: [(14,26,1)]  
Queue 4: [(0,31,6)]  
  
Time: 34  
Waiting clients:  
(39,35,4) (18,37,4) (11,38,4) (24,38,3) (44,38,2) (1,39,2) (21,39,3) (40,39,4)  
Queue 0: [(47,31,5)]  
Queue 1: [(32,29,1)]  
Queue 2: [(26,30,1)]  
Queue 3: [(20,34,1)]  
Queue 4: [(0,31,5)]
```

```
Time: 35
Waiting clients:
(18,37,4) (11,38,4) (24,38,3) (44,38,2) (1,39,2) (21,39,3) (40,39,4)
Queue 0: [(47,31,4)]
Queue 1: [(39,35,4)]
Queue 2: closed
Queue 3: closed
Queue 4: [(0,31,4)]

Time: 36
Waiting clients:
(18,37,4) (11,38,4) (24,38,3) (44,38,2) (1,39,2) (21,39,3) (40,39,4)
Queue 0: [(47,31,3)]
Queue 1: [(39,35,3)]
Queue 2: closed
Queue 3: closed
Queue 4: [(0,31,3)]

Time: 37
Waiting clients: (11,38,4) (24,38,3) (44,38,2) (1,39,2) (21,39,3) (40,39,4)
Queue 0: [(47,31,2)]
Queue 1: [(39,35,2)]
Queue 2: [(18,37,4)]
Queue 3: closed
Queue 4: [(0,31,2)]

Time: 38
Waiting clients: (1,39,2) (21,39,3) (40,39,4)
Queue 0: [(47,31,1), (24,38,3)]
Queue 1: [(39,35,1), (44,38,2)]
Queue 2: [(18,37,3)]
Queue 3: [(11,38,4)]
Queue 4: [(0,31,1)]

Time: 39
Waiting clients:
Queue 0: [(24,38,3)]
Queue 1: [(44,38,2), (21,39,3)]
Queue 2: [(18,37,2), (40,39,4)]
Queue 3: [(11,38,3)]
Queue 4: [(1,39,2)]

Time: 40
Waiting clients:
Queue 0: [(24,38,2)]
Queue 1: [(44,38,1), (21,39,3)]
Queue 2: [(18,37,1), (40,39,4)]
Queue 3: [(11,38,2)]
Queue 4: [(1,39,1)]

Time: 41
Waiting clients:
Queue 0: [(24,38,1)]
Queue 1: [(21,39,3)]
Queue 2: [(40,39,4)]
Queue 3: [(11,38,1)]
Queue 4: closed

Time: 42
Waiting clients:
Queue 0: closed
```

```

Queue 1: [(21,39,2)]
Queue 2: [(40,39,3)]
Queue 3: closed
Queue 4: closed

Time: 43
Waiting clients:
Queue 0: closed
Queue 1: [(21,39,1)]
Queue 2: [(40,39,2)]
Queue 3: closed
Queue 4: closed

Time: 44
Waiting clients:
Queue 0: closed
Queue 1: closed
Queue 2: [(40,39,1)]
Queue 3: closed
Queue 4: closed

----- Results -----
Average service time: 3.78
Peak hour: 26

```

Test 3:

... datorită numărului foarte mare de clienți și a spațiului ocupat, am decis pun în prezentare doar starea cozii la timpul 199 al simulării.

```

Time: 199
Waiting clients:
Queue 0: [(232,72,2), (685,74,3), (590,75,3), (89,77,4), (456,78,7),
(787,80,7), (425,83,6), (54,85,3), (480,85,6), (32,87,8), (784,88,5),
(300,90,8), (332,92,6), (648,94,4), (195,96,6), (382,98,6)]
Queue 1: [(252,73,2), (757,74,6), (270,77,3), (46,78,5), (814,79,6),
(623,81,3), (613,82,5), (534,84,8), (218,86,5), (744,87,6), (802,88,7),
(591,90,8), (337,93,5), (102,95,7), (411,97,7)]
Queue 2: [(705,72,1), (671,74,3), (298,75,6), (629,77,3), (911,78,6),
(890,80,4), (222,82,7), (560,84,7), (828,85,6), (876,87,8), (354,89,5),
(627,90,3), (871,91,7), (437,94,3), (462,95,7), (679,97,4), (274,99,5)]
Queue 3: [(190,73,3), (184,75,6), (373,77,8), (271,80,3), (196,81,5),
(706,82,5), (573,84,5), (542,85,6), (70,87,8), (805,88,8), (701,90,7),
(432,93,8), (240,96,3), (117,97,7), (527,99,4)]
Queue 4: [(61,74,7), (8,76,6), (927,78,5), (453,80,8), (633,83,6),
(55,85,7), (375,86,4), (919,87,8), (431,89,6), (843,90,6), (817,92,5),
(712,94,8), (444,97,3), (397,98,4)]
Queue 5: [(850,72,5), (715,75,8), (954,78,4), (273,80,3), (264,81,5),
(997,82,4), (286,84,7), (620,85,7), (968,87,7), (971,88,6), (88,91,4),
(130,92,3), (518,93,7), (530,95,3), (691,96,3), (770,97,3), (484,98,3)]
Queue 6: [(496,73,5), (771,75,4), (426,77,8), (395,80,3), (423,81,3),
(257,82,8), (975,84,8), (682,86,3), (149,87,4), (360,88,8), (396,90,8),
(0,93,3), (933,93,7), (521,96,8), (343,99,3)]
Queue 7: [(888,72,4), (339,75,3), (17,76,8), (643,79,5), (439,81,8),
(93,84,4), (65,85,8), (835,86,3), (980,87,6), (809,88,7), (231,91,6),
(182,93,8), (957,95,3), (793,96,6), (864,98,6)]
Queue 8: [(122,74,3), (284,75,4), (79,76,8), (670,79,4), (974,80,5),
(519,82,6), (582,84,4), (377,85,5), (692,86,4), (996,87,7), (990,88,6),
(312,91,3), (63,92,6), (964,93,7), (563,96,4), (929,97,4), (510,99,3)]
Queue 9: [(907,72,4), (349,75,3), (114,76,6), (180,79,3), (840,79,7),
(345,82,7), (697,84,8), (269,86,5), (999,87,8), (440,89,4), (618,90,8),

```

```

(663,93,8), (322,96,7), (441,98,7)]
Queue 10: [(589,73,7), (223,76,5), (490,78,4), (43,80,3), (57,81,3),
(699,81,6), (101,84,6), (420,85,7), (84,87,6), (584,88,7), (450,90,7),
(403,92,5), (553,94,3), (528,95,6), (471,97,3), (482,98,4)]
Queue 11: [(595,73,4), (353,75,4), (365,77,4), (493,78,3), (689,79,6),
(509,81,3), (551,82,8), (78,85,5), (898,85,7), (209,88,4), (773,88,3),
(562,89,8), (86,92,3), (255,93,5), (854,94,3), (994,95,3), (901,96,8),
(713,99,4)]
Queue 12: [(152,72,2), (847,74,7), (473,77,3), (576,78,4), (121,80,4),
(488,81,3), (352,82,7), (912,84,3), (119,85,6), (804,86,4), (124,88,7),
(133,89,4), (503,90,6), (283,92,5), (987,93,6), (330,96,4), (567,97,4),
(108,99,6)]
Queue 13: [(402,74,7), (768,76,3), (636,77,8), (477,80,3), (536,81,8),
(321,84,8), (131,86,5), (336,87,8), (197,89,8), (831,91,7), (42,94,6),
(485,96,3), (188,97,7), (848,99,8)]
Queue 14: [(662,74,3), (285,75,8), (64,78,4), (727,79,7), (718,81,4),
(969,83,5), (29,85,8), (830,86,7), (579,88,3), (878,88,3), (44,90,5),
(604,91,8), (100,94,8), (906,96,8), (946,99,4)]
Queue 15: [(200,73,4), (451,75,5), (491,77,5), (307,79,8), (722,81,3),
(251,83,7), (116,85,8), (938,86,7), (588,88,5), (47,90,4), (443,91,5),
(656,92,6), (48,95,8), (668,97,7)]
Queue 16: [(720,73,6), (792,75,4), (646,77,8), (497,80,5), (407,82,7),
(939,84,3), (166,85,3), (669,85,8), (215,88,5), (949,88,8), (631,91,3),
(302,92,8), (148,95,6), (220,97,5), (146,99,5)]
Queue 17: [(247,73,6), (936,75,5), (344,78,3), (316,79,4), (541,80,7),
(350,83,6), (34,85,3), (459,85,7), (120,87,4), (290,88,7), (161,90,7),
(90,92,8), (83,95,7), (361,97,8)]
Queue 18: [(681,72,2), (53,75,5), (824,76,7), (619,79,8), (756,81,7),
(507,84,8), (216,86,5), (381,87,8), (204,89,6), (516,91,6), (256,93,6),
(346,95,8), (358,98,6)]
Queue 19: [(755,73,6), (948,75,3), (512,77,3), (661,78,6), (628,80,4),
(171,82,3), (372,83,8), (177,85,7), (958,86,5), (293,88,8), (374,90,3),
(533,91,5), (789,92,3), (726,93,4), (364,95,8), (359,98,3), (244,99,7)]

```

----- Results -----
Average service time: 5.506
Peak hour: 99

VI. Concluzii

Am observat că în funcție de datele de intrare, o strategie poate fi mai eficientă decât cealaltă. Aplicația poate fi îmbunătățită prin afișarea peak hour-ului și a timpului mediu de servire în interfața grafică, la momentul terminării simulării.

Pentru siguranța thredurilor (*thread safety*) am folosit variabile atomice, pentru a putea compune două acțiuni făcute asupra unui thread (d.e. `AtomicInteger i= new AtomicInteger(); i.getAndIncrement();`).

VII. Bibliografie

- https://didatec.sharepoint.com/sites/TP_Seria_2_2020_2021/Class%20Materials/Laborator/Tema%202/ASSIGNMENT_2_SUPPORT_PRESENTATION.pdf?CT=1617810301772&OR=ItemsView
- <https://docs.oracle.com/javase/tutorial/>

- <https://stackoverflow.com>
-