

**FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE**

**CATEDRA CALCULATOARE**

**Simulator - cozi de asteptare**

DOCUMENTAȚIE

Tirlea Maria Cristina

Grupa 30225 | An 2 semestrul 2

Cuprins

**1 . ................................................................................... Obiectiv**

**2................................................................... Studiul probelemei**

**3...................................................................................Proiectare**

**3.1..............................................................Diagrame UML**

**4.............................................................................Implementare**

**4.1............................................................................... Clase**

**4.2.............................................................................Metode**

**4.3.................................................................................GUI**

**5.....................................................................................Rezultate**

**6…………………………………………………….....Concluzii**

**7.................................................................................Bibliografie**

1. Obiectiv

Scopul acestui proiect este de a simula comportamentul unor cozi de asteptare, pentru a reliefa modul în care acestea pot fi create și manipulate prin intermediul thread-urilorsi a paradigmei programării concurente, orientate pe obiecte. Se prezinta astfel un numar de cozi de asteptare La fiecare coada in parte se proceseaza un numar de clienti, care se aseaza, la un anumit moment, la coada cu timpul de asteptare cel mai scurt, asteapta sa ajunga in fata cozii pentru a fi procesat, odata ajuns in fata este procesat pentru un interval de timp de procesare specific, dupa care paraseste coada.

Ca oricare alt sistem de calcul, si acesta prezinta date de intrare si date de iesire, colectate prin intermediul unei interfete grafice care permite atat introducerea parametrilor de intrare a simulatorului, cat si afisarea in mod real a simularii si a rezultatelor obtinute.

Pasii urmati pentru indeplinirea obiectivului:

* Impartirea sarcinilor de implementat intr-un pattern architectural aproximativ asemanator cu Model-View-Controller, precum si in pattern-ul Singleton
* Realizarea interfetei prin intermediul careia se colecteaza datale de intrare necesare simularii si se afiseaza in timp real simularea in sine
* Descompunerea problemei prin definirea claselor necesare reprezentarii cozilor de asteptare si a clientilor
* Realizarea functinalitatii componentelor din interfata prin implementarea unei clase de control (Controller)
* Implementarea unei clase principale care manageriaza procesul de simulare
* Implementarea unei clase utilitare pentru scrierea concurenta a update-urilor intr-un fisier de output si in componentele din interfata care afiseaza simularea
* Pregatirea rezultatelor pentru a fi prezentate utilizatorului in interfata
* Testarea rezultatelor obtinute pe trei cazuri, prezentate in acest document, in capitolul

1. Analiza problemei

# Cerinte functionale

- Cozile sunt de cele mai multe ori utilizate pentru a modela probleme din lumea reală. Obiectivul principal al unei cozi este acela de a oferi un loc pentru un „client“ să aștepte înainte de a primi un „serviciu“. Managementul unei cozi urmareste minimizarea cantității timpului de asteptare a „clientii“ în coada înainte ca acestea sa fie servit.

- Orice coadă poate fi văzută ca o pereche casă de servire - client care așteaptă la coadă, această corespondență fiind modelată după următorul aspect : fiecare coadă are clienți care trebuie procesați.

- Clientul poate să aleagă ( în funcție de numărul de case de servire puse la dispoziție) cărei case de servire să fie asociat. Modul de asociere depinde de cum este văzută problema:

- Un client se aseaza, la un anumit moment in timp, in coada unde exista deja cei mai putini clienti inaintea sa. Asteapta in coada pana sunt seviti pe rand fiecare dintre clietii aflati inaintea sa, urmand sa ajunga chiar el in fata cozii. Odata ajuns, petrece in fata cozii o perioada de procesare, timp in care este servit, la finalul careia, paraseste coada. Acest proces se desfasoara intr-un anumit interval de simulare determinat.

# Cazuri de utilizare/ Use-cases

**Use-case:** pregatirea aplicatiei pentru simulare

**Actor principal:** utilizator

**Scenariul principal de succes:**

1. Utilizatorul introduce parametrii de intrare necesari simularii

2. Utilizatorul selectează butonul de "Submit"

3. Aplicatia initializeaza datele de intrare cu valorile preluate de la utilizator

4. Optional, utilizatorul omite acest use-case deoarece campurile datelor de intrare sunt presetate cu niste valori implicite la lansarea ferestrei

**Secvență alternativă:** Utlizatorul nu apasa butonul “Submit”

- Simularea nu va porni in acest caz

- Scenariul revine la pasul 1

**Use-case:** Simularea comportamentului specific cozilor de servire

**Actor principal:** utilizator

**Scenariul principal de succes:**

1. Utilizatorul respecta pasii mentionati in use-case-ul anterior

2. Utilizatorul selectează operațiunea de “Start simulation”

3. Aplicatia incepe afisarea secventiala a etapelor de simulare la fiecare moment

**Use-case:** Terminarea fortata a simularii si afisarea rezultatelor obtinute pana in acel moment

**Actor principal:** utilizator

**Scenariul principal de succes:**

1. Utilizatorul respecta pasii mentionati in use-case-ul anterior

2. Utilizatorul selectează operațiunea de “End simulation”

3. Aplicatia afiseaza rezultatele simularii pana la momentul opririi sale

**Secvență alternativă:** Utilizatorul nu selectează butonul de “End simulation”

- Simularea continua conform strategiei implementate pana cand timpul ajunge la valoarea tMAXsimulation, specificata la primul use-case

- Aplicatia afiseaza rezultatele simularii

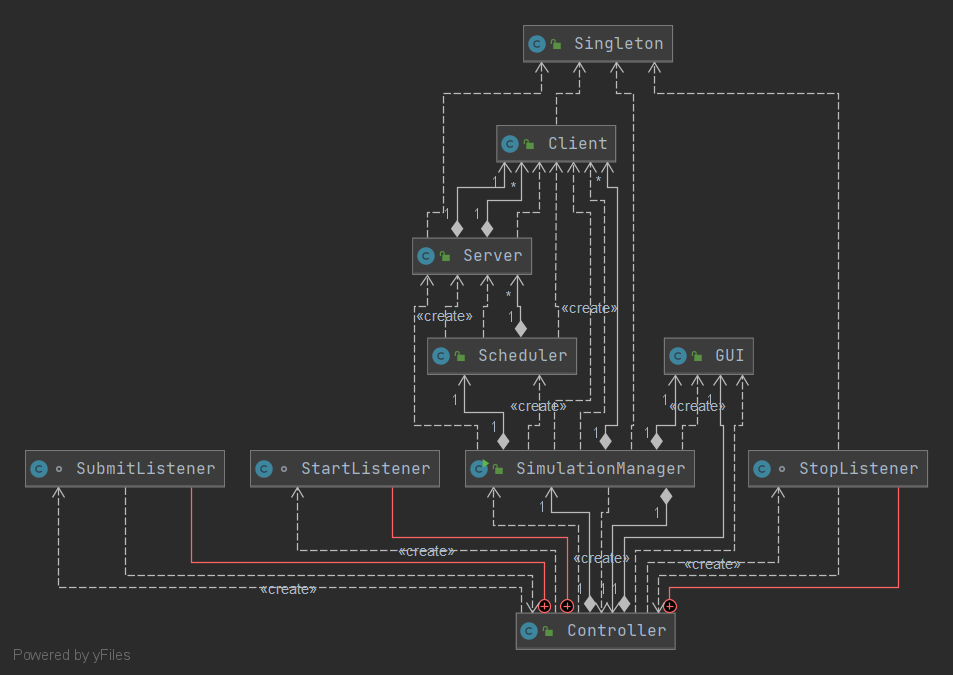
1. Proiectare

# Diagrama UML

Unified Modeling Language sau UML este un limbaj standard pentru descrierea de modele si specificatii pentru software. UML a fost dezvoltat pentru reprezentarea complexității programelor orientate pe obiect, al căror fundament este structurarea problemelor in clase, și instanțele acestora (numite și obiecte). Cu toate acestea, datorită eficienței și clarității în reprezentarea unor elemente abstracte, UML este utilizat si dincolo de domeniul IT.

Diagrama de clase oferă o notaţie grafică pentru reprezentarea:

* claselor - entităţi ce au caracteristici comune
* relațiilor - relațiile dintre două sau mai multe clase

**Diagrama UML de clase**:

****

**Proiectarea OOP a aplicatiei:**

Simulatorul de cozi este structurat pe baza pattern-ului architectural Model-View-Controller (MVC) divizat in 3 zone functionale: procesarea datelor (Simulation Manager, Client, Server, Scheduler), colectarea datelor de intrare, respectiv prezentarea catre utlizator a datelor de iesire (GUI) si o parte care se ocupa de comunicarea dintre cele doua componente mentionate anterior (Controller si Singleton) .

Alt pattern architectural utilizat in rezolvarea problemei este pattern-ul Singleton. Pattern-ul Singleton este utilizat pentru a restricționa numărul de instanțieri ale unei clase la un singur obiect, deci reprezintă o metodă de a folosi o singură instanță a unui obiect în aplicație. Motivul optiunii pentru abordare este de a asigura siguranta thread-urilor care scriu in mod concurrent in fisierul text de iesire si in componenta JTextArea a interfetei. Astfel, toate thread-urile vor scrie in acelasi fisier/ componenta de text.

Programarea concurenta:

În programarea concurentă, există două unități de bază de execuție: procese și fire.

Pe un computer rulează multe fire și procese active, chiar dacă există doar un procesor single-core. Acest lucru se realizează prin caracteristica OS de tranșare a timpului. În Java, programarea concurentă se referă la fire de executie.

Clasele care implementeaza thread-uri sunt Clasa Server (creaza thread-uri concurente, unul pentru fiecare coada) si clasa SimulationManager (thread-ul principal care ruleaza aplicatia in sine si adauga clientii in cozi).

Pentru asigurarea sigurantei thread-urilor am folosit concept precum:

* Atomicitate: declararea variabilelor instanta modificate concurrent de thread-uri ca fiind instante ale claselor AtomicInteger, AtomicBoolean sau AtomicReferance.
* Singleton pentru de a asigura siguranta thread-urilor care scriu in mod concurrent in fisierul text de iesire si in componenta JTextArea a interfetei. Astfel, toate thread-urile vor scrie in acelasi fisier/ componenta de text, utilizand metodele syncromized de writeTofile(), writeToGUI(), writeLineToFile().
* BlockingQueue pentru reprezentarea cozilor

Clasa SimulationManager imlementeaza thread-ul principal care porneste celelate thread-uri. Thread-ul principal incepe doar dupa initializarea datelor de intrare si transmiterea lor din interfata. Pana cand

1. Implementare

# . Clase

**Clasa Server**

Fiecare coada (Clasa Server) este de fapt un thread, fir de executie, in total ruland atatea thread-uri, cate cozi se simuleaza. Clasa Server implementeaza Runnable, iar fiecare thread in executie scoate cate un client din fata cozii, intrerupand executia curenta pentru un interval de timp egal cu timpul de procesare a clientului respectiv, timp in care se decrementeaza aceasta perioada de procesare iar modificarile aparute sunt afisate in timp real in interfata.

**Clasa SimulationManager**

Clasa SimulationManager imlementeaza thread-ul principal care porneste celelate thread-uri. Thread-ul principal incepe doar dupa initializarea datelor de intrare si transmiterea lor din interfata. Pana cand este apasat butonul de Submit si implicit butonul de start din interfata, thread-ul este pus in asteptare prin metoda Thread.sleep(). Metoda run() suprascrisa in clasa SimulationManager contorizeaza timpul de executie si foloseste metoda de strategie mplementata in clasa Scheduler pentru a adauga clienti in cozi pentru a fi procesati.

**Clasa Scheduler**

* Aceasta clasa implementeaza strategia de transmitere a clientilor/ task-urilor catre cozi dupa o anumita logica (clientul va intra in coada cu timpul de asteptare cel mai scurt).
* Tot in aceasta clasa sunt create instantele propriu-zise ale clasei Server, sub forma unei liste de cozi, fiecare element din aceasta lista fiind dat ca parametru unei instante a clasei Thread, iar fiecare din aceste instante este pornita cu metoda .start() in constructorul clasei Scheduler.

**Clasa Singleton**

Clasa Singleton asigura siguranta thread-urilor care scriu in mod concurrent in fisierul text de iesire si in componenta JTextArea a interfetei. Astfel, toate thread-urile vor scrie in acelasi fisier/ componenta de text, utilizand metodele syncromized de writeTofile(), writeToGUI(), writeLineToFile().

Am utilizat acasta abordare pentru a restricționa numărul de instanțieri ale unei clase la un singur obiect, deci reprezintă o metodă de a folosi o singură instanță a unui obiect în aplicație. Pentru asta am declarat constructorul si singura instanta ca fiind private si permit accesul la unica instanta cu metoda publica getInstance() care returneza instanta clasei.

**Clasa GUI**

* Incapsuleaza componentele care alcatuiesc interfata grafica (vederea).

**Clasa Controller**

* Clasa Controller primeste datele introduse de utilizator in interfata ca evenimente care denotă mișcarea mouse-ului, activarea butoanele mouse-ului sau de la tastatura.
* Permite interactiunea intre utilizator si sistemul de simulare.
* Face lagatura intre interfata si functionalitatea oferita de componentele de modelare.
* Initializeaza varabilele de instanta din clasa SimulationManager cu valorile introduse de utilizator in interfata.
* Permite lanseara in executie a simularii si oprirea acesteia.

**Clasa Client**

Aceasta clasa modeleaza informatiile despre fiecare client, precum timpul la care un client va intra in coada, timpul de procesare necesar si un ID de identificare.

* 1. **. Metode**

**Metodele clasei Client**

Aceasta clasa contine metode getters si setters pentru variabilele instanta care modeleaza clientii, metodele printClientToGUI() si represent() pentru a printa clientii in fisier sau in componenta de text din interfata, metoda compareTo() pentru a sorta clientii in functie de timpul de intrare in coada a fiecaruia, metoda setWaitingPeriodOnChosenServer(Integer waitingPeriodOnChosenServer) pentru a tine evidenta timpului de asteptare pe care il are un client cand intra intr-o coada si metoda decProcessingPeriod() pentru a tine evidenta procesului de procesare odata ajuns in fata unei cozi.

**Metodele clasei Server**

addClient(Client client) – pune un client in coada si imncrementeaza timpul de asteptare al acelei cozi

takeClient() – metoda care scoate cate un client din fata cozii si intrerupe executarea thread-ului curent pentru o perioada de timp egala cu timpul de procesare a clientului, timp in care acest timp de procesare se decrementeaza secvential si se afiseaza aceste update-uri in fisier si in interfata

run() – apeleaza metoda takeClient() intr-un loop. Aceasta metoda implementeaza rularea unui thread de tip coada si este apelata in momentul in care un astfel de thread este pornit

**Metodele clasei SimulatonManager**

run() - Metoda run() suprascrisa in clasa SimulationManager contorizeaza timpul de executie si foloseste metoda de strategie mplementata in clasa Scheduler pentru a adauga clienti in cozi pentru a fi procesati.

sendStatusToUI (int currentTime)– metoda care apeleaza la metodele din clasa Singleton pentru a scrie modificarile aparute asupra procesului de simulare in componenta text din interfata

sendStatusToFile(int currentTime) - metoda care apeleaza la metodele din clasa Singleton pentru a scrie modificarile aparute asupra procesului de simulare in fisierul de output

public double computeAvgServiceTimeInQueue() – metoda care calculeaza media timpului de procesare

public double computeAvgWaitingTimeInQueue - metoda care calculeaza media timpului de asteptare

private void generateNRandomClients() – metoda care genereaza random N client

public static void main(String[] args) – metoda care porneste thread-ul principal care executa simularea propriu-zisa

**Metodele clasei Singleton**

public void writeToFile(Object o) – aceasta metoda permite scrierea sincronizata in fisier a unui string

public void writeLineToFile(Object o) – aceasta metoda permite scrierea sincronizata in fisier a unui string cu caracter de new-line la sfarsit

public void writeToUI(Object o) - aceasta metoda permite scrierea sincronizata in fisier a unui string in componenta text area din interfata

**Metodele clasei Scheduler**

* dispatchClient(Client c)– metoda apeleaza addTask() care trimite un client catre un thread tip server
* public void addTask(List<Server> servers, Client c) - implementeaza strategia de transmitere a clientilor/ task-urilor catre cozi dupa o anumita logica (clientul va intra in coada cu timpul de asteptare cel mai scurt)
* public Scheduler (Integer maxNumOfServers, Integer maxNumOfClientsPerServer) – creaza instantele propriu-zise ale clasei Server, sub forma unei liste de cozi, fiecare element din aceasta lista fiind dat ca parametru unei instante a clasei Thread, iar fiecare din aceste instante este pornita cu metoda .start() in constructorul clasei Scheduler.

**Metodele clasei Controller:**

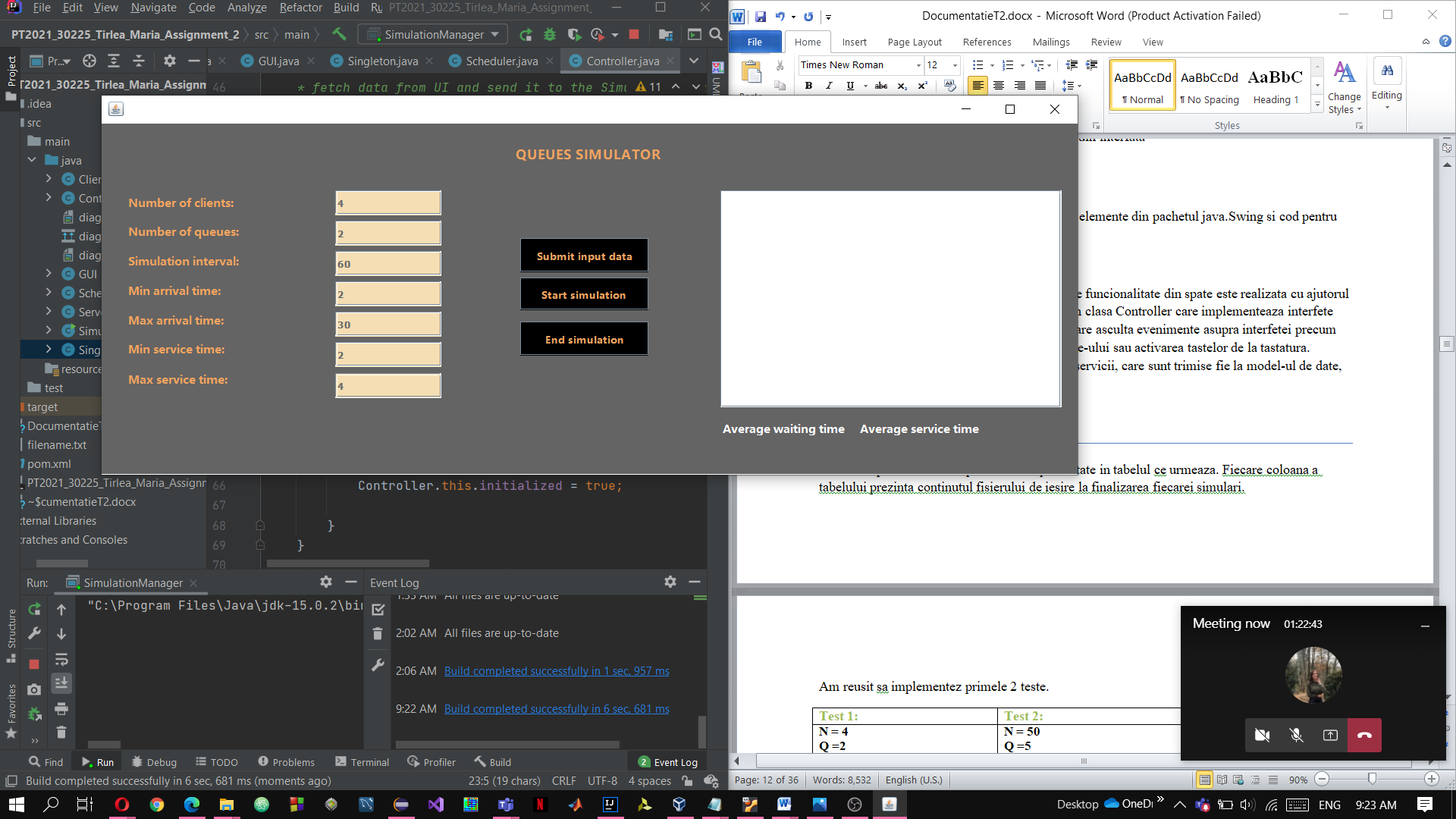
Clasa controller contine trei inner-classes care implementeaza actionListeners pentru butoanele din interfata.

- matoda actionPerformed() din clasa StartListener seteaza un flag care permite inceperea thread-ului principal.

- matoda actionPerformed() din clasa StopListener seteaza un flag care permite oprirea thread-ului principal si incide fisierul de output

- matoda actionPerformed() din clasa SubmitListener initializeaza variabilele instanta din clasa SimulationManager cu valorile luate din interfata

**4.3. GUI**

 - Interfata implementata in clasa View contine elemente din pachetul java.Swing si cod pentru layout-ul acestor elemente :

Comunicarea intre interfata/ utiliator si codul de funcionalitate din spate este realizata cu ajutorul clasei Controller si al claselor interne definite in clasa Controller care implementeaza interfete precum ActionListener, care asculta evenimente asupra interfetei precum mișcarea mouse-ului, activarea butoanele mouse-ului sau activarea tastelor de la tastatura. Evenimentele sunt ulterior traduse în cereri de servicii, care sunt trimise fie la model-ul de date, fie la vedere.

1. Testare

Scenariile pentru testarea aplicatiei sunt prezentate in tabelul ce urmeaza. Fiecare coloana a tabelului prezinta continutul fisierului de iesire la finalizarea fiecarei simulari.

Am reusit sa implementez primele 2 teste.

|  |  |
| --- | --- |
| **Test 1:** | **Test 2:** |
| **N = 4**  **Q =2**  **𝑡𝑠𝑖𝑚𝑢𝑙𝑎𝑡𝑖𝑜𝑛𝑀𝐴𝑋=60sec**  **[𝑡𝑎𝑟𝑟𝑖𝑣𝑎𝑙𝑀𝐼𝑁,𝑡𝑎𝑟𝑟𝑖𝑣𝑎𝑙𝑀𝐴𝑋] = [2,30]**  **[𝑡𝑠𝑒𝑟𝑣𝑖𝑐𝑒𝑀𝐼𝑁,𝑡𝑠𝑒𝑟𝑣𝑖𝑐𝑒𝑀𝐴𝑋] = [2,4]** | **N = 50**  **Q =5**  **𝑡𝑠𝑖𝑚𝑢𝑙𝑎𝑡𝑖𝑜𝑛𝑀𝐴𝑋=60sec**  **[𝑡𝑎𝑟𝑟𝑖𝑣𝑎𝑙𝑀𝐼𝑁,𝑡𝑎𝑟𝑟𝑖𝑣𝑎𝑙𝑀𝐴𝑋] = [2,40]**  **[𝑡𝑠𝑒𝑟𝑣𝑖𝑐𝑒𝑀𝐼𝑁,𝑡𝑠𝑒𝑟𝑣𝑖𝑐𝑒𝑀𝐴𝑋]=**  **[1,7]** |
| Time 0:  Waiting clients: (1, 3, 3) (0, 4, 3) (3, 12, 2) (2, 13, 2)  Queue 0: closed  Queue 1: closed  Time 1:  Waiting clients: (1, 3, 3) (0, 4, 3) (3, 12, 2) (2, 13, 2)  Queue 0: closed  Queue 1: closed  Time 2:  Waiting clients: (1, 3, 3) (0, 4, 3) (3, 12, 2) (2, 13, 2)  Queue 0: closed  Queue 1: closed  Time 3:  Waiting clients: (0, 4, 3) (3, 12, 2) (2, 13, 2)  Queue 0: (1, 3, 3)  Queue 1: closed  Time 4:  Waiting clients: (3, 12, 2) (2, 13, 2)  Queue 0: (1, 3, 2)  Queue 1: (0, 4, 3)  Time 5:  Waiting clients: (3, 12, 2) (2, 13, 2)  Queue 0: (1, 3, 1)  Queue 1: (0, 4, 2)  Time 6:  Waiting clients: (3, 12, 2) (2, 13, 2)  Queue 0: closed  Queue 1: (0, 4, 1)  Time 7:  Waiting clients: (3, 12, 2) (2, 13, 2)  Queue 0: closed  Queue 1: closed  Time 8:  Waiting clients: (3, 12, 2) (2, 13, 2)  Queue 0: closed  Queue 1: closed  Time 9:  Waiting clients: (3, 12, 2) (2, 13, 2)  Queue 0: closed  Queue 1: closed  Time 10:  Waiting clients: (3, 12, 2) (2, 13, 2)  Queue 0: closed  Queue 1: closed  Time 11:  Waiting clients: (3, 12, 2) (2, 13, 2)  Queue 0: closed  Queue 1: closed  Time 12:  Waiting clients: (2, 13, 2)  Queue 0: (3, 12, 2)  Queue 1: closed  Time 13:  Waiting clients:  Queue 0: (3, 12, 1)  Queue 1: (2, 13, 2)  Time 14:  Waiting clients:  Queue 0: closed  Queue 1: (2, 13, 1)  Time 15:  Waiting clients:  Queue 0: closed  Queue 1: closed  Time 16:  Waiting clients:  Queue 0: closed  Queue 1: closed  Time 17:  Waiting clients:  Queue 0: closed  Queue 1: closed  Time 18:  Waiting clients:  Queue 0: closed  Queue 1: closed  Time 19:  Waiting clients:  Queue 0: closed  Queue 1: closed  Time 20:  Waiting clients:  Queue 0: closed  Queue 1: closed  Time 21:  Waiting clients:  Queue 0: closed  Queue 1: closed  Time 22:  Waiting clients:  Queue 0: closed  Queue 1: closed  Time 23:  Waiting clients:  Queue 0: closed  Queue 1: closed  Time 24:  Waiting clients:  Queue 0: closed  Queue 1: closed  Time 25:  Waiting clients:  Queue 0: closed  Queue 1: closed  Time 26:  Waiting clients:  Queue 0: closed  Queue 1: closed  Time 27:  Waiting clients:  Queue 0: closed  Queue 1: closed  Time 28:  Waiting clients:  Queue 0: closed  Queue 1: closed  Time 29:  Waiting clients:  Queue 0: closed  Queue 1: closed  Time 30:  Waiting clients:  Queue 0: closed  Queue 1: closed  Time 31:  Waiting clients:  Queue 0: closed  Queue 1: closed  Time 32:  Waiting clients:  Queue 0: closed  Queue 1: closed  Time 33:  Waiting clients:  Queue 0: closed  Queue 1: closed  Time 34:  Waiting clients:  Queue 0: closed  Queue 1: closed  Time 35:  Waiting clients:  Queue 0: closed  Queue 1: closed  Time 36:  Waiting clients:  Queue 0: closed  Queue 1: closed  Time 37:  Waiting clients:  Queue 0: closed  Queue 1: closed  Time 38:  Waiting clients:  Queue 0: closed  Queue 1: closed  Time 39:  Waiting clients:  Queue 0: closed  Queue 1: closed  Time 40:  Waiting clients:  Queue 0: closed  Queue 1: closed  Time 41:  Waiting clients:  Queue 0: closed  Queue 1: closed  Time 42:  Waiting clients:  Queue 0: closed  Queue 1: closed  Time 43:  Waiting clients:  Queue 0: closed  Queue 1: closed  Time 44:  Waiting clients:  Queue 0: closed  Queue 1: closed  Time 45:  Waiting clients:  Queue 0: closed  Queue 1: closed  Time 46:  Waiting clients:  Queue 0: closed  Queue 1: closed  Time 47:  Waiting clients:  Queue 0: closed  Queue 1: closed  Time 48:  Waiting clients:  Queue 0: closed  Queue 1: closed  Time 49:  Waiting clients:  Queue 0: closed  Queue 1: closed  Time 50:  Waiting clients:  Queue 0: closed  Queue 1: closed  Time 51:  Waiting clients:  Queue 0: closed  Queue 1: closed  Time 52:  Waiting clients:  Queue 0: closed  Queue 1: closed  Time 53:  Waiting clients:  Queue 0: closed  Queue 1: closed  Time 54:  Waiting clients:  Queue 0: closed  Queue 1: closed  Time 55:  Waiting clients:  Queue 0: closed  Queue 1: closed  Time 56:  Waiting clients:  Queue 0: closed  Queue 1: closed  Time 57:  Waiting clients:  Queue 0: closed  Queue 1: closed  Time 58:  Waiting clients:  Queue 0: closed  Queue 1: closed  Time 59:  Waiting clients:  Queue 0: closed  Queue 1: closed  Average waiting time: 2.5  Average service time: 2.5 | Time 0:  Waiting clients: (23, 3, 3) (10, 4, 2) (18, 4, 2) (44, 4, 4) (22, 5, 2) (2, 6, 6) (35, 6, 3) (9, 7, 1) (31, 7, 4) (33, 7, 3) (5, 10, 3) (42, 11, 5) (38, 12, 6) (13, 13, 2) (1, 14, 5) (8, 14, 2) (37, 14, 4) (49, 15, 4) (32, 16, 5) (48, 16, 4) (3, 17, 4) (30, 17, 5) (20, 18, 6) (27, 18, 4) (0, 20, 2) (26, 20, 6) (36, 20, 4) (47, 20, 4) (21, 21, 5) (24, 21, 4) (40, 21, 2) (11, 22, 6) (34, 22, 2) (45, 22, 4) (14, 25, 5) (28, 25, 3) (12, 26, 2) (6, 29, 5) (41, 31, 1) (29, 32, 6) (7, 33, 6) (17, 34, 4) (46, 34, 3) (16, 35, 4) (19, 35, 4) (39, 35, 3) (43, 35, 3) (4, 38, 3) (15, 38, 4) (25, 39, 3)  Queue 0: closed  Queue 1: closed  Queue 2: closed  Queue 3: closed  Queue 4: closed  Time 1:  Waiting clients: (23, 3, 3) (10, 4, 2) (18, 4, 2) (44, 4, 4) (22, 5, 2) (2, 6, 6) (35, 6, 3) (9, 7, 1) (31, 7, 4) (33, 7, 3) (5, 10, 3) (42, 11, 5) (38, 12, 6) (13, 13, 2) (1, 14, 5) (8, 14, 2) (37, 14, 4) (49, 15, 4) (32, 16, 5) (48, 16, 4) (3, 17, 4) (30, 17, 5) (20, 18, 6) (27, 18, 4) (0, 20, 2) (26, 20, 6) (36, 20, 4) (47, 20, 4) (21, 21, 5) (24, 21, 4) (40, 21, 2) (11, 22, 6) (34, 22, 2) (45, 22, 4) (14, 25, 5) (28, 25, 3) (12, 26, 2) (6, 29, 5) (41, 31, 1) (29, 32, 6) (7, 33, 6) (17, 34, 4) (46, 34, 3) (16, 35, 4) (19, 35, 4) (39, 35, 3) (43, 35, 3) (4, 38, 3) (15, 38, 4) (25, 39, 3)  Queue 0: closed  Queue 1: closed  Queue 2: closed  Queue 3: closed  Queue 4: closed  Time 2:  Waiting clients: (23, 3, 3) (10, 4, 2) (18, 4, 2) (44, 4, 4) (22, 5, 2) (2, 6, 6) (35, 6, 3) (9, 7, 1) (31, 7, 4) (33, 7, 3) (5, 10, 3) (42, 11, 5) (38, 12, 6) (13, 13, 2) (1, 14, 5) (8, 14, 2) (37, 14, 4) (49, 15, 4) (32, 16, 5) (48, 16, 4) (3, 17, 4) (30, 17, 5) (20, 18, 6) (27, 18, 4) (0, 20, 2) (26, 20, 6) (36, 20, 4) (47, 20, 4) (21, 21, 5) (24, 21, 4) (40, 21, 2) (11, 22, 6) (34, 22, 2) (45, 22, 4) (14, 25, 5) (28, 25, 3) (12, 26, 2) (6, 29, 5) (41, 31, 1) (29, 32, 6) (7, 33, 6) (17, 34, 4) (46, 34, 3) (16, 35, 4) (19, 35, 4) (39, 35, 3) (43, 35, 3) (4, 38, 3) (15, 38, 4) (25, 39, 3)  Queue 0: closed  Queue 1: closed  Queue 2: closed  Queue 3: closed  Queue 4: closed  Time 3:  Waiting clients: (10, 4, 2) (18, 4, 2) (44, 4, 4) (22, 5, 2) (2, 6, 6) (35, 6, 3) (9, 7, 1) (31, 7, 4) (33, 7, 3) (5, 10, 3) (42, 11, 5) (38, 12, 6) (13, 13, 2) (1, 14, 5) (8, 14, 2) (37, 14, 4) (49, 15, 4) (32, 16, 5) (48, 16, 4) (3, 17, 4) (30, 17, 5) (20, 18, 6) (27, 18, 4) (0, 20, 2) (26, 20, 6) (36, 20, 4) (47, 20, 4) (21, 21, 5) (24, 21, 4) (40, 21, 2) (11, 22, 6) (34, 22, 2) (45, 22, 4) (14, 25, 5) (28, 25, 3) (12, 26, 2) (6, 29, 5) (41, 31, 1) (29, 32, 6) (7, 33, 6) (17, 34, 4) (46, 34, 3) (16, 35, 4) (19, 35, 4) (39, 35, 3) (43, 35, 3) (4, 38, 3) (15, 38, 4) (25, 39, 3)  Queue 0: (23, 3, 3)  Queue 1: closed  Queue 2: closed  Queue 3: closed  Queue 4: closed  Time 4:  Waiting clients: (22, 5, 2) (2, 6, 6) (35, 6, 3) (9, 7, 1) (31, 7, 4) (33, 7, 3) (5, 10, 3) (42, 11, 5) (38, 12, 6) (13, 13, 2) (1, 14, 5) (8, 14, 2) (37, 14, 4) (49, 15, 4) (32, 16, 5) (48, 16, 4) (3, 17, 4) (30, 17, 5) (20, 18, 6) (27, 18, 4) (0, 20, 2) (26, 20, 6) (36, 20, 4) (47, 20, 4) (21, 21, 5) (24, 21, 4) (40, 21, 2) (11, 22, 6) (34, 22, 2) (45, 22, 4) (14, 25, 5) (28, 25, 3) (12, 26, 2) (6, 29, 5) (41, 31, 1) (29, 32, 6) (7, 33, 6) (17, 34, 4) (46, 34, 3) (16, 35, 4) (19, 35, 4) (39, 35, 3) (43, 35, 3) (4, 38, 3) (15, 38, 4) (25, 39, 3)  Queue 0: (23, 3, 2)  Queue 1: (10, 4, 2)  Queue 2: (18, 4, 2)  Queue 3: (44, 4, 4)  Queue 4: closed  Time 5:  Waiting clients: (2, 6, 6) (35, 6, 3) (9, 7, 1) (31, 7, 4) (33, 7, 3) (5, 10, 3) (42, 11, 5) (38, 12, 6) (13, 13, 2) (1, 14, 5) (8, 14, 2) (37, 14, 4) (49, 15, 4) (32, 16, 5) (48, 16, 4) (3, 17, 4) (30, 17, 5) (20, 18, 6) (27, 18, 4) (0, 20, 2) (26, 20, 6) (36, 20, 4) (47, 20, 4) (21, 21, 5) (24, 21, 4) (40, 21, 2) (11, 22, 6) (34, 22, 2) (45, 22, 4) (14, 25, 5) (28, 25, 3) (12, 26, 2) (6, 29, 5) (41, 31, 1) (29, 32, 6) (7, 33, 6) (17, 34, 4) (46, 34, 3) (16, 35, 4) (19, 35, 4) (39, 35, 3) (43, 35, 3) (4, 38, 3) (15, 38, 4) (25, 39, 3)  Queue 0: (23, 3, 1)  Queue 1: (10, 4, 1)  Queue 2: (18, 4, 1)  Queue 3: (44, 4, 3)  Queue 4: (22, 5, 2)  Time 6:  Waiting clients: (9, 7, 1) (31, 7, 4) (33, 7, 3) (5, 10, 3) (42, 11, 5) (38, 12, 6) (13, 13, 2) (1, 14, 5) (8, 14, 2) (37, 14, 4) (49, 15, 4) (32, 16, 5) (48, 16, 4) (3, 17, 4) (30, 17, 5) (20, 18, 6) (27, 18, 4) (0, 20, 2) (26, 20, 6) (36, 20, 4) (47, 20, 4) (21, 21, 5) (24, 21, 4) (40, 21, 2) (11, 22, 6) (34, 22, 2) (45, 22, 4) (14, 25, 5) (28, 25, 3) (12, 26, 2) (6, 29, 5) (41, 31, 1) (29, 32, 6) (7, 33, 6) (17, 34, 4) (46, 34, 3) (16, 35, 4) (19, 35, 4) (39, 35, 3) (43, 35, 3) (4, 38, 3) (15, 38, 4) (25, 39, 3)  Queue 0: (2, 6, 6)  Queue 1: (35, 6, 3)  Queue 2: closed  Queue 3: (44, 4, 2)  Queue 4: (22, 5, 1)  Time 7:  Waiting clients: (5, 10, 3) (42, 11, 5) (38, 12, 6) (13, 13, 2) (1, 14, 5) (8, 14, 2) (37, 14, 4) (49, 15, 4) (32, 16, 5) (48, 16, 4) (3, 17, 4) (30, 17, 5) (20, 18, 6) (27, 18, 4) (0, 20, 2) (26, 20, 6) (36, 20, 4) (47, 20, 4) (21, 21, 5) (24, 21, 4) (40, 21, 2) (11, 22, 6) (34, 22, 2) (45, 22, 4) (14, 25, 5) (28, 25, 3) (12, 26, 2) (6, 29, 5) (41, 31, 1) (29, 32, 6) (7, 33, 6) (17, 34, 4) (46, 34, 3) (16, 35, 4) (19, 35, 4) (39, 35, 3) (43, 35, 3) (4, 38, 3) (15, 38, 4) (25, 39, 3)  Queue 0: (2, 6, 5) (33, 7, 3)  Queue 1: (35, 6, 2)  Queue 2: (9, 7, 1)  Queue 3: (44, 4, 1)  Queue 4: (31, 7, 4)  Time 8:  Waiting clients: (5, 10, 3) (42, 11, 5) (38, 12, 6) (13, 13, 2) (1, 14, 5) (8, 14, 2) (37, 14, 4) (49, 15, 4) (32, 16, 5) (48, 16, 4) (3, 17, 4) (30, 17, 5) (20, 18, 6) (27, 18, 4) (0, 20, 2) (26, 20, 6) (36, 20, 4) (47, 20, 4) (21, 21, 5) (24, 21, 4) (40, 21, 2) (11, 22, 6) (34, 22, 2) (45, 22, 4) (14, 25, 5) (28, 25, 3) (12, 26, 2) (6, 29, 5) (41, 31, 1) (29, 32, 6) (7, 33, 6) (17, 34, 4) (46, 34, 3) (16, 35, 4) (19, 35, 4) (39, 35, 3) (43, 35, 3) (4, 38, 3) (15, 38, 4) (25, 39, 3)  Queue 0: (2, 6, 4) (33, 7, 3)  Queue 1: (35, 6, 1)  Queue 2: closed  Queue 3: closed  Queue 4: (31, 7, 3)  Time 9:  Waiting clients: (5, 10, 3) (42, 11, 5) (38, 12, 6) (13, 13, 2) (1, 14, 5) (8, 14, 2) (37, 14, 4) (49, 15, 4) (32, 16, 5) (48, 16, 4) (3, 17, 4) (30, 17, 5) (20, 18, 6) (27, 18, 4) (0, 20, 2) (26, 20, 6) (36, 20, 4) (47, 20, 4) (21, 21, 5) (24, 21, 4) (40, 21, 2) (11, 22, 6) (34, 22, 2) (45, 22, 4) (14, 25, 5) (28, 25, 3) (12, 26, 2) (6, 29, 5) (41, 31, 1) (29, 32, 6) (7, 33, 6) (17, 34, 4) (46, 34, 3) (16, 35, 4) (19, 35, 4) (39, 35, 3) (43, 35, 3) (4, 38, 3) (15, 38, 4) (25, 39, 3)  Queue 0: (2, 6, 3) (33, 7, 3)  Queue 1: closed  Queue 2: closed  Queue 3: closed  Queue 4: (31, 7, 2)  Time 10:  Waiting clients: (42, 11, 5) (38, 12, 6) (13, 13, 2) (1, 14, 5) (8, 14, 2) (37, 14, 4) (49, 15, 4) (32, 16, 5) (48, 16, 4) (3, 17, 4) (30, 17, 5) (20, 18, 6) (27, 18, 4) (0, 20, 2) (26, 20, 6) (36, 20, 4) (47, 20, 4) (21, 21, 5) (24, 21, 4) (40, 21, 2) (11, 22, 6) (34, 22, 2) (45, 22, 4) (14, 25, 5) (28, 25, 3) (12, 26, 2) (6, 29, 5) (41, 31, 1) (29, 32, 6) (7, 33, 6) (17, 34, 4) (46, 34, 3) (16, 35, 4) (19, 35, 4) (39, 35, 3) (43, 35, 3) (4, 38, 3) (15, 38, 4) (25, 39, 3)  Queue 0: (2, 6, 2) (33, 7, 3)  Queue 1: (5, 10, 3)  Queue 2: closed  Queue 3: closed  Queue 4: (31, 7, 1)  Time 11:  Waiting clients: (38, 12, 6) (13, 13, 2) (1, 14, 5) (8, 14, 2) (37, 14, 4) (49, 15, 4) (32, 16, 5) (48, 16, 4) (3, 17, 4) (30, 17, 5) (20, 18, 6) (27, 18, 4) (0, 20, 2) (26, 20, 6) (36, 20, 4) (47, 20, 4) (21, 21, 5) (24, 21, 4) (40, 21, 2) (11, 22, 6) (34, 22, 2) (45, 22, 4) (14, 25, 5) (28, 25, 3) (12, 26, 2) (6, 29, 5) (41, 31, 1) (29, 32, 6) (7, 33, 6) (17, 34, 4) (46, 34, 3) (16, 35, 4) (19, 35, 4) (39, 35, 3) (43, 35, 3) (4, 38, 3) (15, 38, 4) (25, 39, 3)  Queue 0: (2, 6, 1) (33, 7, 3)  Queue 1: (5, 10, 2)  Queue 2: (42, 11, 5)  Queue 3: closed  Queue 4: closed  Time 12:  Waiting clients: (13, 13, 2) (1, 14, 5) (8, 14, 2) (37, 14, 4) (49, 15, 4) (32, 16, 5) (48, 16, 4) (3, 17, 4) (30, 17, 5) (20, 18, 6) (27, 18, 4) (0, 20, 2) (26, 20, 6) (36, 20, 4) (47, 20, 4) (21, 21, 5) (24, 21, 4) (40, 21, 2) (11, 22, 6) (34, 22, 2) (45, 22, 4) (14, 25, 5) (28, 25, 3) (12, 26, 2) (6, 29, 5) (41, 31, 1) (29, 32, 6) (7, 33, 6) (17, 34, 4) (46, 34, 3) (16, 35, 4) (19, 35, 4) (39, 35, 3) (43, 35, 3) (4, 38, 3) (15, 38, 4) (25, 39, 3)  Queue 0: (33, 7, 2)  Queue 1: (5, 10, 1)  Queue 2: (42, 11, 4)  Queue 3: (38, 12, 6)  Queue 4: closed  Time 13:  Waiting clients: (1, 14, 5) (8, 14, 2) (37, 14, 4) (49, 15, 4) (32, 16, 5) (48, 16, 4) (3, 17, 4) (30, 17, 5) (20, 18, 6) (27, 18, 4) (0, 20, 2) (26, 20, 6) (36, 20, 4) (47, 20, 4) (21, 21, 5) (24, 21, 4) (40, 21, 2) (11, 22, 6) (34, 22, 2) (45, 22, 4) (14, 25, 5) (28, 25, 3) (12, 26, 2) (6, 29, 5) (41, 31, 1) (29, 32, 6) (7, 33, 6) (17, 34, 4) (46, 34, 3) (16, 35, 4) (19, 35, 4) (39, 35, 3) (43, 35, 3) (4, 38, 3) (15, 38, 4) (25, 39, 3)  Queue 0: (33, 7, 1)  Queue 1: (13, 13, 2)  Queue 2: (42, 11, 3)  Queue 3: (38, 12, 5)  Queue 4: closed  Time 14:  Waiting clients: (49, 15, 4) (32, 16, 5) (48, 16, 4) (3, 17, 4) (30, 17, 5) (20, 18, 6) (27, 18, 4) (0, 20, 2) (26, 20, 6) (36, 20, 4) (47, 20, 4) (21, 21, 5) (24, 21, 4) (40, 21, 2) (11, 22, 6) (34, 22, 2) (45, 22, 4) (14, 25, 5) (28, 25, 3) (12, 26, 2) (6, 29, 5) (41, 31, 1) (29, 32, 6) (7, 33, 6) (17, 34, 4) (46, 34, 3) (16, 35, 4) (19, 35, 4) (39, 35, 3) (43, 35, 3) (4, 38, 3) (15, 38, 4) (25, 39, 3)  Queue 0: (1, 14, 5) (37, 14, 4)  Queue 1: (13, 13, 1)  Queue 2: (42, 11, 2)  Queue 3: (38, 12, 4)  Queue 4: (8, 14, 2)  Time 15:  Waiting clients: (32, 16, 5) (48, 16, 4) (3, 17, 4) (30, 17, 5) (20, 18, 6) (27, 18, 4) (0, 20, 2) (26, 20, 6) (36, 20, 4) (47, 20, 4) (21, 21, 5) (24, 21, 4) (40, 21, 2) (11, 22, 6) (34, 22, 2) (45, 22, 4) (14, 25, 5) (28, 25, 3) (12, 26, 2) (6, 29, 5) (41, 31, 1) (29, 32, 6) (7, 33, 6) (17, 34, 4) (46, 34, 3) (16, 35, 4) (19, 35, 4) (39, 35, 3) (43, 35, 3) (4, 38, 3) (15, 38, 4) (25, 39, 3)  Queue 0: (1, 14, 4) (37, 14, 4)  Queue 1: (49, 15, 4)  Queue 2: (42, 11, 1)  Queue 3: (38, 12, 3)  Queue 4: (8, 14, 1)  Time 16:  Waiting clients: (3, 17, 4) (30, 17, 5) (20, 18, 6) (27, 18, 4) (0, 20, 2) (26, 20, 6) (36, 20, 4) (47, 20, 4) (21, 21, 5) (24, 21, 4) (40, 21, 2) (11, 22, 6) (34, 22, 2) (45, 22, 4) (14, 25, 5) (28, 25, 3) (12, 26, 2) (6, 29, 5) (41, 31, 1) (29, 32, 6) (7, 33, 6) (17, 34, 4) (46, 34, 3) (16, 35, 4) (19, 35, 4) (39, 35, 3) (43, 35, 3) (4, 38, 3) (15, 38, 4) (25, 39, 3)  Queue 0: (1, 14, 3) (37, 14, 4)  Queue 1: (49, 15, 3)  Queue 2: (32, 16, 5)  Queue 3: (38, 12, 2)  Queue 4: (48, 16, 4)  Time 17:  Waiting clients: (20, 18, 6) (27, 18, 4) (0, 20, 2) (26, 20, 6) (36, 20, 4) (47, 20, 4) (21, 21, 5) (24, 21, 4) (40, 21, 2) (11, 22, 6) (34, 22, 2) (45, 22, 4) (14, 25, 5) (28, 25, 3) (12, 26, 2) (6, 29, 5) (41, 31, 1) (29, 32, 6) (7, 33, 6) (17, 34, 4) (46, 34, 3) (16, 35, 4) (19, 35, 4) (39, 35, 3) (43, 35, 3) (4, 38, 3) (15, 38, 4) (25, 39, 3)  Queue 0: (1, 14, 2) (37, 14, 4)  Queue 1: (49, 15, 2) (3, 17, 4)  Queue 2: (32, 16, 4) (30, 17, 5)  Queue 3: (38, 12, 1)  Queue 4: (48, 16, 3)  Time 18:  Waiting clients: (0, 20, 2) (26, 20, 6) (36, 20, 4) (47, 20, 4) (21, 21, 5) (24, 21, 4) (40, 21, 2) (11, 22, 6) (34, 22, 2) (45, 22, 4) (14, 25, 5) (28, 25, 3) (12, 26, 2) (6, 29, 5) (41, 31, 1) (29, 32, 6) (7, 33, 6) (17, 34, 4) (46, 34, 3) (16, 35, 4) (19, 35, 4) (39, 35, 3) (43, 35, 3) (4, 38, 3) (15, 38, 4) (25, 39, 3)  Queue 0: (1, 14, 1) (37, 14, 4)  Queue 1: (49, 15, 1) (3, 17, 4)  Queue 2: (32, 16, 3) (30, 17, 5)  Queue 3: (20, 18, 6) (27, 18, 4)  Queue 4: (48, 16, 2)  Time 19:  Waiting clients: (0, 20, 2) (26, 20, 6) (36, 20, 4) (47, 20, 4) (21, 21, 5) (24, 21, 4) (40, 21, 2) (11, 22, 6) (34, 22, 2) (45, 22, 4) (14, 25, 5) (28, 25, 3) (12, 26, 2) (6, 29, 5) (41, 31, 1) (29, 32, 6) (7, 33, 6) (17, 34, 4) (46, 34, 3) (16, 35, 4) (19, 35, 4) (39, 35, 3) (43, 35, 3) (4, 38, 3) (15, 38, 4) (25, 39, 3)  Queue 0: (37, 14, 3)  Queue 1: (3, 17, 3)  Queue 2: (32, 16, 2) (30, 17, 5)  Queue 3: (20, 18, 5) (27, 18, 4)  Queue 4: (48, 16, 1)  Time 20:  Waiting clients: (21, 21, 5) (24, 21, 4) (40, 21, 2) (11, 22, 6) (34, 22, 2) (45, 22, 4) (14, 25, 5) (28, 25, 3) (12, 26, 2) (6, 29, 5) (41, 31, 1) (29, 32, 6) (7, 33, 6) (17, 34, 4) (46, 34, 3) (16, 35, 4) (19, 35, 4) (39, 35, 3) (43, 35, 3) (4, 38, 3) (15, 38, 4) (25, 39, 3)  Queue 0: (37, 14, 2) (26, 20, 6)  Queue 1: (3, 17, 2) (36, 20, 4)  Queue 2: (32, 16, 1) (30, 17, 5)  Queue 3: (20, 18, 4) (27, 18, 4)  Queue 4: (0, 20, 2) (47, 20, 4)  Time 21:  Waiting clients: (11, 22, 6) (34, 22, 2) (45, 22, 4) (14, 25, 5) (28, 25, 3) (12, 26, 2) (6, 29, 5) (41, 31, 1) (29, 32, 6) (7, 33, 6) (17, 34, 4) (46, 34, 3) (16, 35, 4) (19, 35, 4) (39, 35, 3) (43, 35, 3) (4, 38, 3) (15, 38, 4) (25, 39, 3)  Queue 0: (37, 14, 1) (26, 20, 6) (24, 21, 4)  Queue 1: (3, 17, 1) (36, 20, 4) (40, 21, 2)  Queue 2: (30, 17, 4) (21, 21, 5)  Queue 3: (20, 18, 3) (27, 18, 4)  Queue 4: (0, 20, 1) (47, 20, 4)  Time 22:  Waiting clients: (14, 25, 5) (28, 25, 3) (12, 26, 2) (6, 29, 5) (41, 31, 1) (29, 32, 6) (7, 33, 6) (17, 34, 4) (46, 34, 3) (16, 35, 4) (19, 35, 4) (39, 35, 3) (43, 35, 3) (4, 38, 3) (15, 38, 4) (25, 39, 3)  Queue 0: (26, 20, 6) (24, 21, 4) (34, 22, 2)  Queue 1: (36, 20, 4) (40, 21, 2) (45, 22, 4)  Queue 2: (30, 17, 3) (21, 21, 5)  Queue 3: (20, 18, 2) (27, 18, 4)  Queue 4: (47, 20, 3) (11, 22, 6)  Time 23:  Waiting clients: (14, 25, 5) (28, 25, 3) (12, 26, 2) (6, 29, 5) (41, 31, 1) (29, 32, 6) (7, 33, 6) (17, 34, 4) (46, 34, 3) (16, 35, 4) (19, 35, 4) (39, 35, 3) (43, 35, 3) (4, 38, 3) (15, 38, 4) (25, 39, 3)  Queue 0: (26, 20, 5) (24, 21, 4) (34, 22, 2)  Queue 1: (36, 20, 3) (40, 21, 2) (45, 22, 4)  Queue 2: (30, 17, 2) (21, 21, 5)  Queue 3: (20, 18, 1) (27, 18, 4)  Queue 4: (47, 20, 2) (11, 22, 6)  Time 24:  Waiting clients: (14, 25, 5) (28, 25, 3) (12, 26, 2) (6, 29, 5) (41, 31, 1) (29, 32, 6) (7, 33, 6) (17, 34, 4) (46, 34, 3) (16, 35, 4) (19, 35, 4) (39, 35, 3) (43, 35, 3) (4, 38, 3) (15, 38, 4) (25, 39, 3)  Queue 0: (26, 20, 4) (24, 21, 4) (34, 22, 2)  Queue 1: (36, 20, 2) (40, 21, 2) (45, 22, 4)  Queue 2: (30, 17, 1) (21, 21, 5)  Queue 3: (27, 18, 3)  Queue 4: (47, 20, 1) (11, 22, 6)  Time 25:  Waiting clients: (12, 26, 2) (6, 29, 5) (41, 31, 1) (29, 32, 6) (7, 33, 6) (17, 34, 4) (46, 34, 3) (16, 35, 4) (19, 35, 4) (39, 35, 3) (43, 35, 3) (4, 38, 3) (15, 38, 4) (25, 39, 3)  Queue 0: (26, 20, 3) (24, 21, 4) (34, 22, 2)  Queue 1: (36, 20, 1) (40, 21, 2) (45, 22, 4)  Queue 2: (21, 21, 5) (14, 25, 5)  Queue 3: (27, 18, 2) (28, 25, 3)  Queue 4: (11, 22, 6)  Time 26:  Waiting clients: (6, 29, 5) (41, 31, 1) (29, 32, 6) (7, 33, 6) (17, 34, 4) (46, 34, 3) (16, 35, 4) (19, 35, 4) (39, 35, 3) (43, 35, 3) (4, 38, 3) (15, 38, 4) (25, 39, 3)  Queue 0: (26, 20, 2) (24, 21, 4) (34, 22, 2)  Queue 1: (40, 21, 1) (45, 22, 4)  Queue 2: (21, 21, 4) (14, 25, 5)  Queue 3: (27, 18, 1) (28, 25, 3)  Queue 4: (11, 22, 5) (12, 26, 2)  Time 27:  Waiting clients: (6, 29, 5) (41, 31, 1) (29, 32, 6) (7, 33, 6) (17, 34, 4) (46, 34, 3) (16, 35, 4) (19, 35, 4) (39, 35, 3) (43, 35, 3) (4, 38, 3) (15, 38, 4) (25, 39, 3)  Queue 0: (26, 20, 1) (24, 21, 4) (34, 22, 2)  Queue 1: (45, 22, 4)  Queue 2: (21, 21, 3) (14, 25, 5)  Queue 3: (28, 25, 3)  Queue 4: (11, 22, 4) (12, 26, 2)  Time 28:  Waiting clients: (6, 29, 5) (41, 31, 1) (29, 32, 6) (7, 33, 6) (17, 34, 4) (46, 34, 3) (16, 35, 4) (19, 35, 4) (39, 35, 3) (43, 35, 3) (4, 38, 3) (15, 38, 4) (25, 39, 3)  Queue 0: (24, 21, 3) (34, 22, 2)  Queue 1: (45, 22, 3)  Queue 2: (21, 21, 2) (14, 25, 5)  Queue 3: (28, 25, 2)  Queue 4: (11, 22, 3) (12, 26, 2)  Time 29:  Waiting clients: (41, 31, 1) (29, 32, 6) (7, 33, 6) (17, 34, 4) (46, 34, 3) (16, 35, 4) (19, 35, 4) (39, 35, 3) (43, 35, 3) (4, 38, 3) (15, 38, 4) (25, 39, 3)  Queue 0: (24, 21, 2) (34, 22, 2)  Queue 1: (45, 22, 2) (6, 29, 5)  Queue 2: (21, 21, 1) (14, 25, 5)  Queue 3: (28, 25, 1)  Queue 4: (11, 22, 2) (12, 26, 2)  Time 30:  Waiting clients: (41, 31, 1) (29, 32, 6) (7, 33, 6) (17, 34, 4) (46, 34, 3) (16, 35, 4) (19, 35, 4) (39, 35, 3) (43, 35, 3) (4, 38, 3) (15, 38, 4) (25, 39, 3)  Queue 0: (24, 21, 1) (34, 22, 2)  Queue 1: (45, 22, 1) (6, 29, 5)  Queue 2: (14, 25, 4)  Queue 3: closed  Queue 4: (11, 22, 1) (12, 26, 2)  Time 31:  Waiting clients: (29, 32, 6) (7, 33, 6) (17, 34, 4) (46, 34, 3) (16, 35, 4) (19, 35, 4) (39, 35, 3) (43, 35, 3) (4, 38, 3) (15, 38, 4) (25, 39, 3)  Queue 0: (34, 22, 2)  Queue 1: (6, 29, 4)  Queue 2: (14, 25, 3)  Queue 3: (41, 31, 1)  Queue 4: (12, 26, 1)  Time 32:  Waiting clients: (7, 33, 6) (17, 34, 4) (46, 34, 3) (16, 35, 4) (19, 35, 4) (39, 35, 3) (43, 35, 3) (4, 38, 3) (15, 38, 4) (25, 39, 3)  Queue 0: (34, 22, 1)  Queue 1: (6, 29, 3)  Queue 2: (14, 25, 2)  Queue 3: (29, 32, 6)  Queue 4: closed  Time 33:  Waiting clients: (17, 34, 4) (46, 34, 3) (16, 35, 4) (19, 35, 4) (39, 35, 3) (43, 35, 3) (4, 38, 3) (15, 38, 4) (25, 39, 3)  Queue 0: (7, 33, 6)  Queue 1: (6, 29, 2)  Queue 2: (14, 25, 1)  Queue 3: (29, 32, 5)  Queue 4: closed  Time 34:  Waiting clients: (16, 35, 4) (19, 35, 4) (39, 35, 3) (43, 35, 3) (4, 38, 3) (15, 38, 4) (25, 39, 3)  Queue 0: (7, 33, 5)  Queue 1: (6, 29, 1)  Queue 2: (17, 34, 4)  Queue 3: (29, 32, 4)  Queue 4: (46, 34, 3)  Time 35:  Waiting clients: (4, 38, 3) (15, 38, 4) (25, 39, 3)  Queue 0: (7, 33, 4) (19, 35, 4)  Queue 1: (16, 35, 4) (39, 35, 3)  Queue 2: (17, 34, 3) (43, 35, 3)  Queue 3: (29, 32, 3)  Queue 4: (46, 34, 2)  Time 36:  Waiting clients: (4, 38, 3) (15, 38, 4) (25, 39, 3)  Queue 0: (7, 33, 3) (19, 35, 4)  Queue 1: (16, 35, 3) (39, 35, 3)  Queue 2: (17, 34, 2) (43, 35, 3)  Queue 3: (29, 32, 2)  Queue 4: (46, 34, 1)  Time 37:  Waiting clients: (4, 38, 3) (15, 38, 4) (25, 39, 3)  Queue 0: (7, 33, 2) (19, 35, 4)  Queue 1: (16, 35, 2) (39, 35, 3)  Queue 2: (17, 34, 1) (43, 35, 3)  Queue 3: (29, 32, 1)  Queue 4: closed  Time 38:  Waiting clients: (25, 39, 3)  Queue 0: (7, 33, 1) (19, 35, 4)  Queue 1: (16, 35, 1) (39, 35, 3)  Queue 2: (43, 35, 2)  Queue 3: (4, 38, 3)  Queue 4: (15, 38, 4)  Time 39:  Waiting clients:  Queue 0: (19, 35, 3) (25, 39, 3)  Queue 1: (39, 35, 2)  Queue 2: (43, 35, 1)  Queue 3: (4, 38, 2)  Queue 4: (15, 38, 3)  Time 40:  Waiting clients:  Queue 0: (19, 35, 2) (25, 39, 3)  Queue 1: (39, 35, 1)  Queue 2: closed  Queue 3: (4, 38, 1)  Queue 4: (15, 38, 2)  Time 41:  Waiting clients:  Queue 0: (19, 35, 1) (25, 39, 3)  Queue 1: closed  Queue 2: closed  Queue 3: closed  Queue 4: (15, 38, 1)  Time 42:  Waiting clients:  Queue 0: (25, 39, 3)  Queue 1: closed  Queue 2: closed  Queue 3: closed  Queue 4: closed  Time 43:  Waiting clients:  Queue 0: (25, 39, 2)  Queue 1: closed  Queue 2: closed  Queue 3: closed  Queue 4: closed  Time 44:  Waiting clients:  Queue 0: (25, 39, 1)  Queue 1: closed  Queue 2: closed  Queue 3: closed  Queue 4: closed  Time 45:  Waiting clients:  Queue 0: closed  Queue 1: closed  Queue 2: closed  Queue 3: closed  Queue 4: closed  Time 46:  Waiting clients:  Queue 0: closed  Queue 1: closed  Queue 2: closed  Queue 3: closed  Queue 4: closed  Time 47:  Waiting clients:  Queue 0: closed  Queue 1: closed  Queue 2: closed  Queue 3: closed  Queue 4: closed  Time 48:  Waiting clients:  Queue 0: closed  Queue 1: closed  Queue 2: closed  Queue 3: closed  Queue 4: closed  Time 49:  Waiting clients:  Queue 0: closed  Queue 1: closed  Queue 2: closed  Queue 3: closed  Queue 4: closed  Time 50:  Waiting clients:  Queue 0: closed  Queue 1: closed  Queue 2: closed  Queue 3: closed  Queue 4: closed  Time 51:  Waiting clients:  Queue 0: closed  Queue 1: closed  Queue 2: closed  Queue 3: closed  Queue 4: closed  Time 52:  Waiting clients:  Queue 0: closed  Queue 1: closed  Queue 2: closed  Queue 3: closed  Queue 4: closed  Time 53:  Waiting clients:  Queue 0: closed  Queue 1: closed  Queue 2: closed  Queue 3: closed  Queue 4: closed  Time 54:  Waiting clients:  Queue 0: closed  Queue 1: closed  Queue 2: closed  Queue 3: closed  Queue 4: closed  Time 55:  Waiting clients:  Queue 0: closed  Queue 1: closed  Queue 2: closed  Queue 3: closed  Queue 4: closed  Time 56:  Waiting clients:  Queue 0: closed  Queue 1: closed  Queue 2: closed  Queue 3: closed  Queue 4: closed  Time 57:  Waiting clients:  Queue 0: closed  Queue 1: closed  Queue 2: closed  Queue 3: closed  Queue 4: closed  Time 58:  Waiting clients:  Queue 0: closed  Queue 1: closed  Queue 2: closed  Queue 3: closed  Queue 4: closed  Time 59:  Waiting clients:  Queue 0: closed  Queue 1: closed  Queue 2: closed  Queue 3: closed  Queue 4: closed  Average waiting time: 4.26  Average service time: 3.74 |

1. Concluzii

In concluzie, consider ca acest proiect mi-a aprofundat cunostiintele de programare in limbajul Java si de implementare a paradigmelor OOP. Problema a cuprins toate etapele importante ale dezvoltarii unei aplicatii cu interfata grafica respectand standardul OOP, fapt care m-a ajutat sa imi reamintesc aspectele predate la cursul de Programare Orientata pe Obiecte de semestrul trecut si mi-a oferit totodata oportunitatea de a crea de la 0 o aplicatie functionala si intuitiva intr-un timp relativ scurt.

1. Bibliografie

* FUNDAMENTAL PROGRAMMING TECHNIQUES ASSIGNMENT 2 – SUPPORT PRESENTATION
* Cursuri POO
* <https://dzone.com/articles/multi-threading-java-swing>
* <http://elf.cs.pub.ro/poo/laboratoare/design-patterns>