DOCUMENTATIE

TEMA *2*

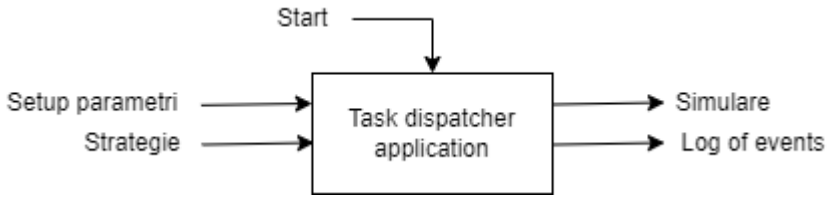
NUME STUDENT: Bartolomei Vlad-Alexandru
GRUPA: 30229

CUPRINS

1. Obiectivul temei 3
2. Analiza problemei, modelare, scenarii, cazuri de utilizare 3
3. Proiectare 5
4. Implementare 9
5. Rezultate 11
6. Concluzii 12
7. Bibliografie 12

1. Obiectivul temei

Obiectivul principal este de a implementa o aplicație care simulează distribuirea eficientă a unor task-uri cu diferiți timpi de ajungere și de procesare la diferite servere, în funcție de două strategii.

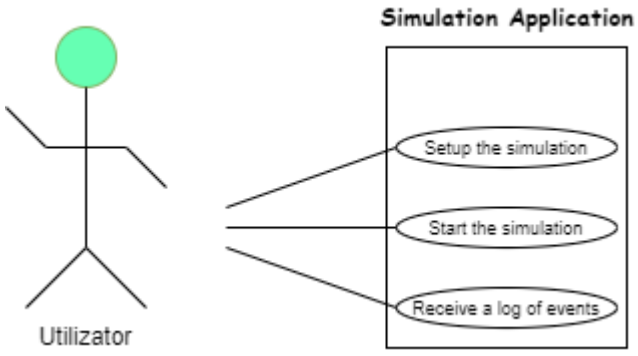


Obiective secundare:

- Analiza problemei, public țintă, modul în care un asemenea dispozitiv (descris ca un black box, mai sus) ar putea fi utilizat -> Capitolul 2
- Proiectarea interfeței grafice și a implementării funcționalităților. Cerințele (non-) funcționale se găsesc în Capitolul 2, iar împărțirea pe pachete și clase, precum și modul în care interacționează clasele, se găsesc în Capitolul 3.
- Descrierea într-un limbaj natural a celor mai importante metode și a implementării interfeței grafice, pe scurt. Se găsește în Capitolul 4.
- Observarea rezultatelor (Capitolul 5) și tragerea concluziilor în urma implementării, însoțite de posibile viitoare îmbunătățiri ale aplicației (Capitolul 6)
- Punerea în evidență a celor mai importante resurse bibliografice (Capitolul 7).

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

2.1. Diagrama Use Case



Use case: execuția sistemului (setup, observarea simulării, recepție log ov events)

Actor principal: utilizatorul

Scenariul principal de succes:

A) Uitilizatorul introduce valorile pentru: numărul de clienți, numărul de cozi, intervalul de simulare (doar valoarea maximă, valoarea minima fiind implicit 0), minimul și maximul pentru arrival Time și Service Time.

B) Utilizatorul apasă butonul „Lansează execuția”

C) Sistemul validează datele și lansează automat execuția simulării

D) Se așteaptă completarea bării de progress

E) Se închide fereastra de simulare prin butonul implicit de închidere al ferestrei

F) Se generează automat un jurnal al evenimentelor.

Scenarii Alternative: Date incorecte

- La apăsarea butonului „Lansează execuția”, validatorul va constata neregulile și va semnala userului prin căsuțe de dialog tipul erorii (câmpuri incomplete sau care nu sunt întregi pozitivi, minim mai mare decât maxim; unele câmpuri au o limită impusă – numărul de cozi poate fi maxim 20). => ne întoarcem la pasul A

- Se închide fereastra de simulare înainte ca bara de progres să fie completă. Ca outcome, nu se va genera nimic în jurnalul de evenimente. => soluție: reluarea execuției de la pasul A

2.2. Cerințe funcționale

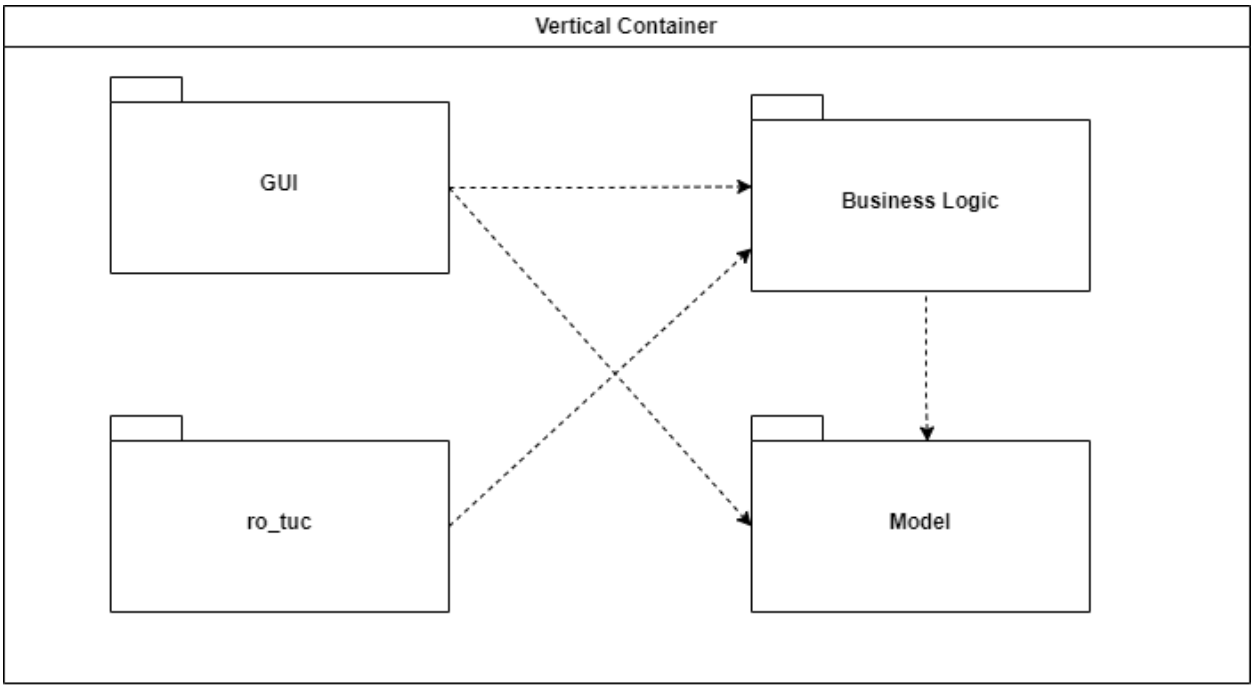
- Sistemul ar trebui să permit utilizatorului să seteze parametrii simulării: număr de clienți și de servere, timp de simulare (dat în secunde), intervalele minim-maxim de arrival time și processing time, numărul maxim de clienți care pot intra la o coadă, strategia de distribuire.
- Sistemul ar trebui să permită utilizatorului să termine execuția programului prin apăsarea butonului de închidere al ferestrei.
- Sistemul nu ar trebui să permită utilizatorului extinderea ferestrei.
- La lansarea execuției, fereastra de setup ar trebui să se închidă și să apară o nouă fereastră de simulare.
- La ajungerea timpului curent la valoarea maximă, utilizatorul ar trebui să acționeze butonul X (de închidere) al ferestrei de simulare pentru a genera Log of Events.

2.3. Cerințe non-funcționale

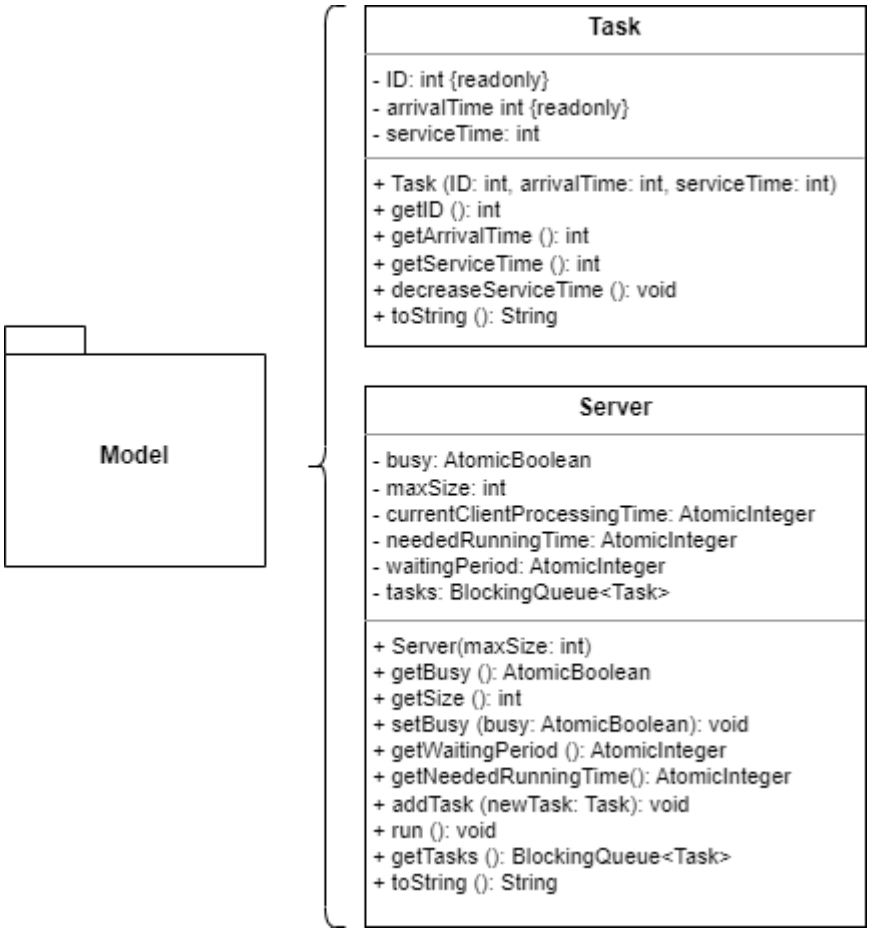
- Sistemul ar trebui să fie intuitiv și ușor de utilizat de utilizator.
- Sistemul ar trebui să aibă un aspect atractiv pentru utilizator.

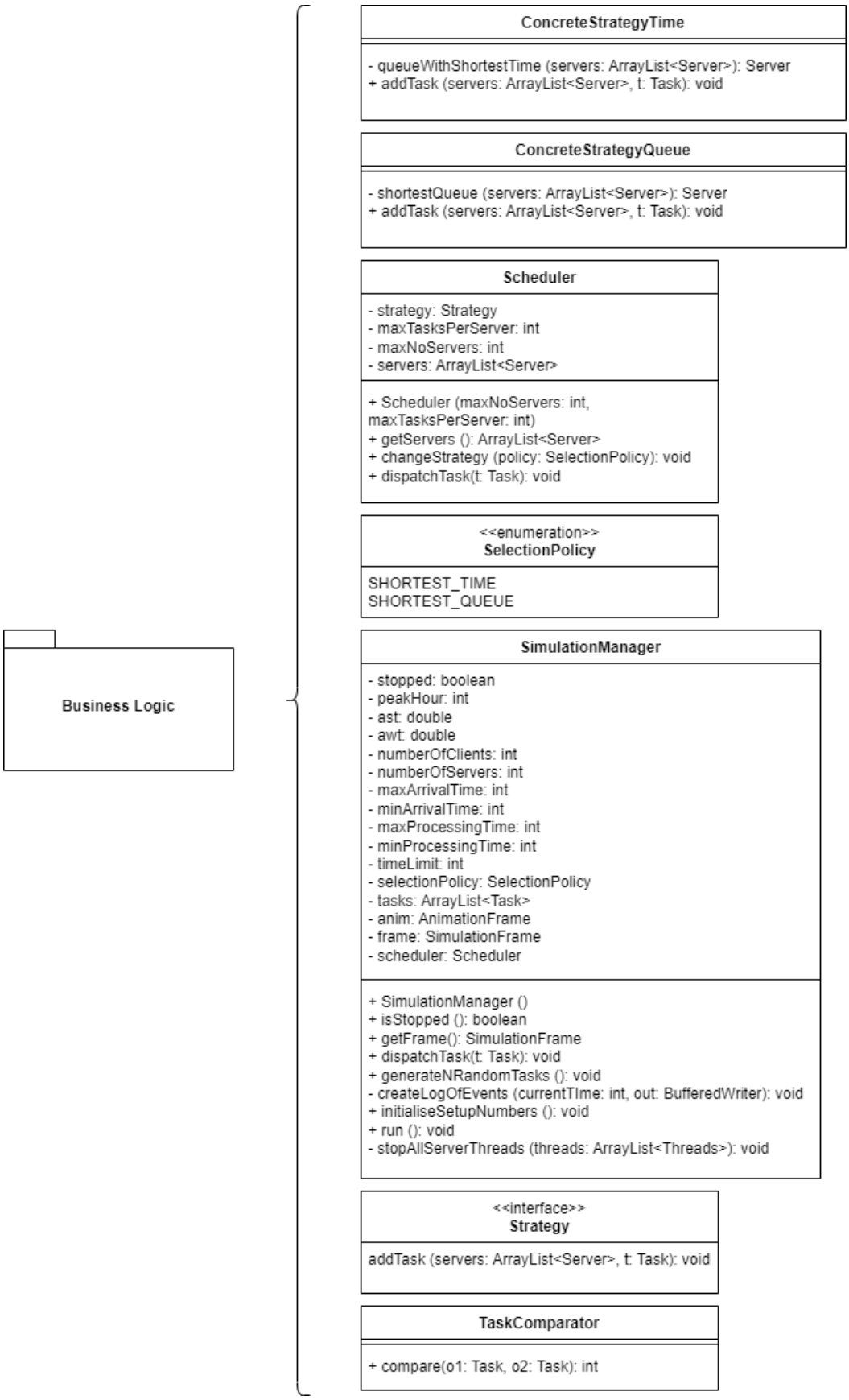
3. Proiectare

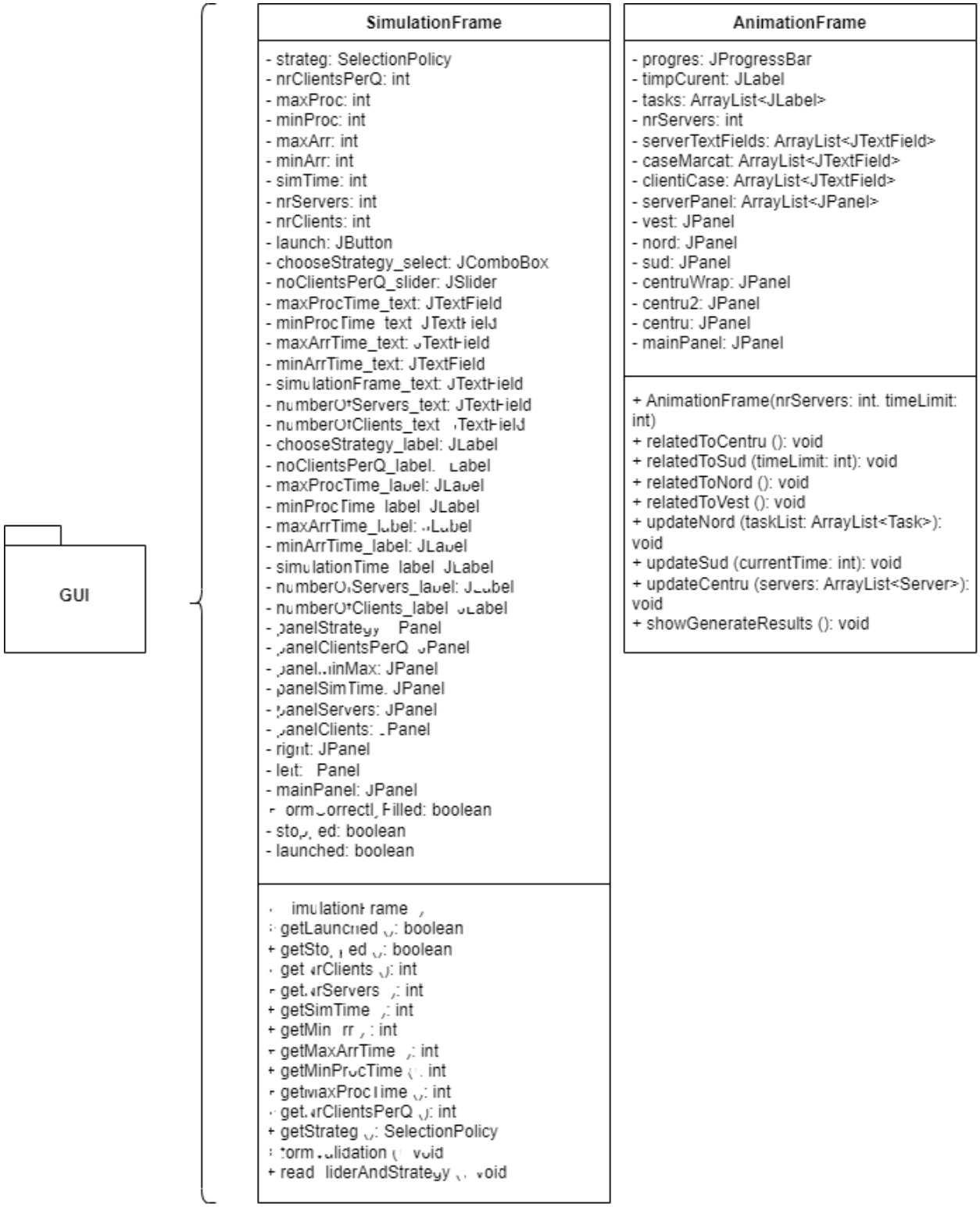
3.1. Diagrama de pachete

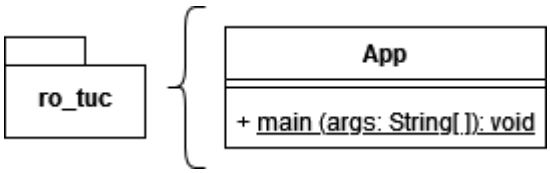


3.2. Conținutul fiecărui pachet

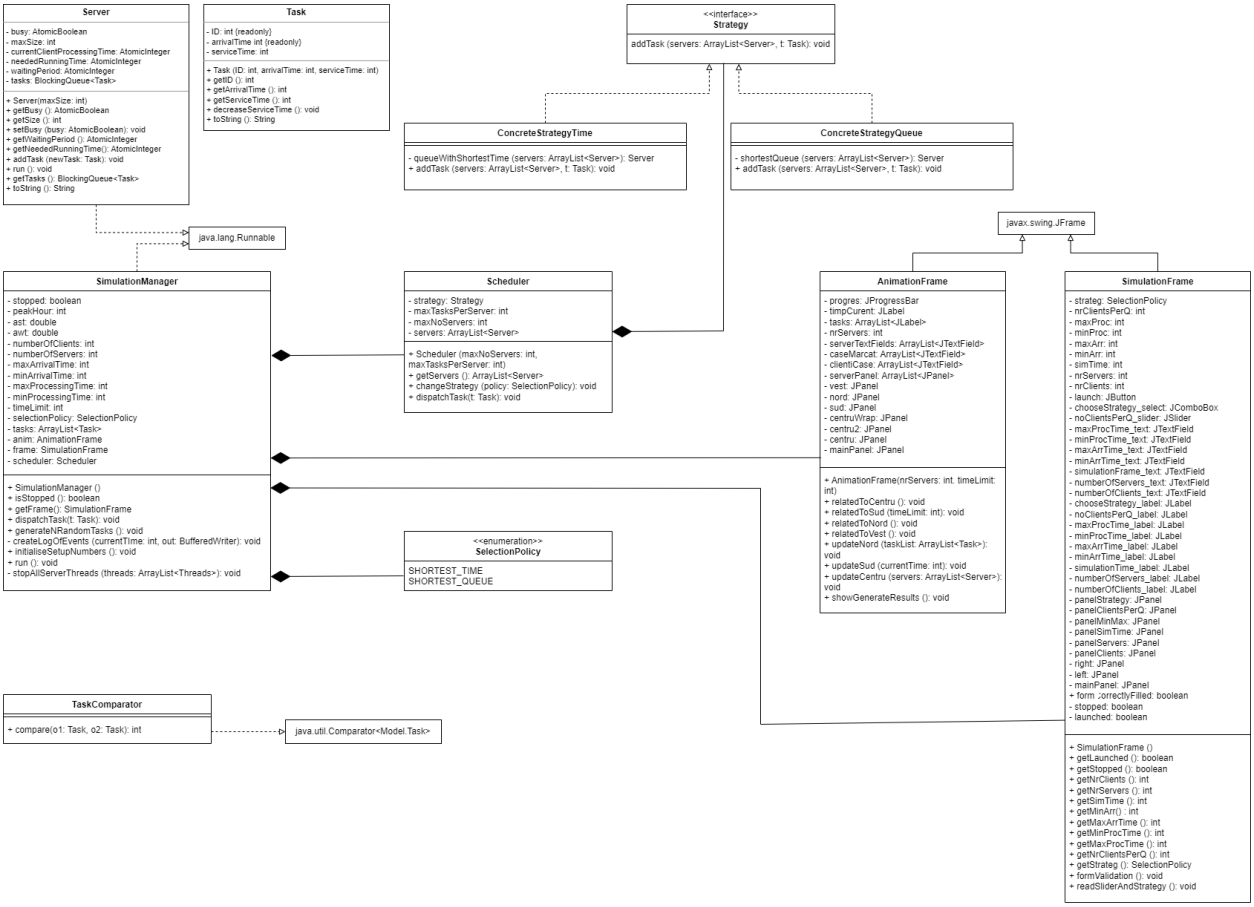



Business Logic
ConcreteStrategyTime
- queueWithShortestTime (servers: ArrayList<Server>): Server
+ addTask (servers: ArrayList<Server>, t: Task): void
ConcreteStrategyQueue
- shortestQueue (servers: ArrayList<Server>): Server
+ addTask (servers: ArrayList<Server>, t: Task): void
Scheduler
- strategy: Strategy
- maxTasksPerServer: int
- maxNoServers: int
- servers: ArrayList<Server>
+ Scheduler (maxNoServers: int, maxTasksPerServer: int)
+ getServers (): ArrayList<Server>
+ changeStrategy (policy: SelectionPolicy): void
+ dispatchTask(t: Task): void
<<enumeration>>
SelectionPolicy
SHORTEST_TIME
SHORTEST_QUEUE
SimulationManager
- stopped: boolean
- peakHour: int
- ast: double
- awt: double
- numberOfClients: int
- numberOfServers: int
- maxArrivalTime: int
- minArrivalTime: int
- maxProcessingTime: int
- minProcessingTime: int
- timeLimit: int
- selectionPolicy: SelectionPolicy
- tasks: ArrayList<Task>
- anim: AnimationFrame
- frame: SimulationFrame
- scheduler: Scheduler
+ SimulationManager ()
+ isStopped (): boolean
+ getFrame(): SimulationFrame
+ dispatchTask(t: Task): void
+ generateNRandomTasks (): void
- createLogOfEvents (currentTIme: int, out: BufferedWriter): void
+ initialiseSetupNumbers (): void
+ run (): void
- stopAllServerThreads (threads: ArrayList<Threads>): void
<<interface>>
Strategy
addTask (servers: ArrayList<Server>, t: Task): void
TaskComparator
+ compare(o1: Task, o2: Task): int


GUI
SimulationFrame
- strateg: SelectionPolicy
- nrClientsPerQ: int
- maxProc: int
- minProc: int
- maxArr: int
- minArr: int
- simTime: int
- nrServers: int
- nrClients: int
- launch: JButton
- chooseStrategy_select: JComboBox
- noClientsPerQ_slider: JSlider
- maxProcTime_text: JTextField
- minProcTime_text: JTextField
- maxArrTime_text: JTextField
- minArrTime_text: JTextField
- simulationFrame_text: JTextField
- numberOfServers_text: JTextField
- numberOfClients_text: JTextField
- chooseStrategy_label: JLabel
- noClientsPerQ_label: JLabel
- maxProcTime_label: JLabel
- minProcTime_label: JLabel
- maxArrTime_label: JLabel
- minArrTime_label: JLabel
- simulationTime_label: JLabel
- numberOfServers_label: JLabel
- numberOfClients_label: JLabel
- panelStrategy: JPanel
- panelClientsPerQ: JPanel
- panelMinMax: JPanel
- panelSimTime: JPanel
- panelServers: JPanel
- panelClients: JPanel
- right: JPanel
- left: JPanel
- mainPanel: JPanel
- formCorrectlyFilled: boolean
- stopped: boolean
- launched: boolean
+ SimulationFrame()
+ getLaunched(): boolean
+ getStopped(): boolean
+ getNrClients(): int
+ getNrServers(): int
+ getSimTime(): int
+ getMinArr(): int
+ getMaxArrTime(): int
+ getMinProcTime(): int
+ getMaxProcTime(): int
+ getNrClientsPerQ(): int
+ getStrategy(): SelectionPolicy
+ formValidation(): void
+ readSliderAndStrategy(): void
AnimationFrame
- progres: JProgressBar
- timpCurent: JLabel
- tasks: ArrayList<JLabel>
- nrServers: int
- serverTextFields: ArrayList<JTextField>
- caseMarcat: ArrayList<JTextField>
- clientiCase: ArrayList<JTextField>
- serverPanel: ArrayList<JPanel>
- vest: JPanel
- nord: JPanel
- sud: JPanel
- centruWrap: JPanel
- centru2: JPanel
- centru: JPanel
- mainPanel: JPanel
+ AnimationFrame(nrServers: int. timeLimit: int)
+ relatedToCentru (): void
+ relatedToSud (timeLimit: int): void
+ relatedToNord (): void
+ relatedToVest (): void
+ updateNord (taskList: ArrayList<Task>): void
+ updateSud (currentTime: int): void
+ updateCentru (servers: ArrayList<Server>): void
+ showGenerateResults (): void



3.3. Diagrama de clase



4. Implementare

În toată implementarea proiectului, de interes ar fi selectarea serverului la care task-ul este distribuit de către Programator (Scheduler) în funcție de strategia selectată din setup.

```
public class ConcreteStrategyQueue implements Strategy{
    1 usage
    private Server shortestQueue(ArrayList<Server> servers) {
        Iterator<Server> s = servers.iterator();
        Server shortest = s.next();
        while(s.hasNext()) {
            Server curr = s.next();
            if(shortest.getSize() > curr.getSize())
                shortest = curr;
        }
        return shortest;
    }
    1 usage
    @Override
    public void addTask(ArrayList<Server> servers, Task t) {
        if(servers.size() > 0) {
            Server theChosenOne = shortestQueue(servers);
            theChosenOne.addTask(t);
        }
    }
}
```

Metoda shortestQueue parcurge toate serverele și returnează serverul care are cei mai puțini clienți. Dacă sunt mai multe de servere de acest fel, se va returna primul găsit. Lui addTask îi rămâne doar să adauge taskul t, transmis ca parametru, la serverul determinat de metoda deja descrisă.

```
public class ConcreteStrategyTime implements Strategy{
    1 usage
    private Server queueWithShortestTime(ArrayList<Server> servers) {
        Iterator<Server> s = servers.iterator();
        Server shortest = s.next();
        while(s.hasNext()) {
            Server curr = s.next();
            if(shortest.getWaitingPeriod().get() > curr.getWaitingPeriod().get())
                shortest = curr;
        }
        return shortest;
    }
    1 usage
    @Override
    public void addTask(ArrayList<Server> servers, Task t) {
        if(servers.size() > 0) {
            Server theChosenOne = queueWithShortestTime(servers);
            theChosenOne.addTask(t);
        }
    }
}
```

În cazul metodei queueWithShortestTime, criteriul de selecție al serverului este shortest waiting period. Dacă sunt mai multe de servere de acest fel, se va returna primul găsit.

```
@Override
public void run() {
    while(true) {
        try {
            if(null != tasks.peek()) {
                //va trebui sa memoram timpul initial de procesare al clientului
                if(0 == currentClientProcessingTime.get())
                    currentClientProcessingTime.set(tasks.peek().getServiceTime());
                Thread.sleep( millis: 1000); //orice task va fi procesat o secunda
                tasks.peek().decreaseServiceTime();
                //scadem acea secunda in care sta
                // *1000 e ca sa transform acest numar in milisecs
                //daca am ajuns cu timpul de servire la 0 => clientul a fost procesat
                if(-1 == tasks.peek().getServiceTime()) {//daca clientul pleaca de la coada, scadem timpul de asteptare
                    waitingPeriod.addAndGet(-currentClientProcessingTime.get());
                    tasks.poll();
                    //resetam pe 0 currentClientProcessingTime
                    currentClientProcessingTime.set(0);
                }
            }
        } catch (InterruptedException e) { }
    }
}
```

Odată ce taskul a ajuns în server și se află pe prima poziție, urmează să fie procesat. O dată la fiecare secundă, pentru a vedea evoluția, se scade timpul de procesare inițial cu 1. Ajunge Processing Time la 0, îl lăsăm și atunci în coadă (server) ca să vedem că taskul a fost terminat de procesat, mai decrementăm o dată și chemăm metoda tasks.poll(), care să îl scoată din server.

Despre interfața grafică

Există două frame-uri în acest proiect, deoarece s-a dorit ca setup-ul și simularea să aibă loc în doi pași (două ferestre diferite): SimulationFrame (setup) și AnimationFrame.

În cazul lui SimulationFrame, există o metodă de validare — formValidation(), care setează un atribut boolean formCorrectlyFilled pe false în cazul în care se constată că există nereguli, precum: unele câmpuri nu au fost completate sau au fost, dar cu altceva decât numere pozitive, câmpul de max este mai mic decât câmpul de min, ș.a.m.d. Dacă formCorrectlyFilled este pe true, fereastra de setup se închide și se lansează automat fereastra de simulare. Acest ”switch” este realizat în clasa SimulationManager (sic!)

În cazul lui AnimationFrame, există metode de update pe panels (nord, centru, sud) care, în funcție de ce se primește de la SimulationManager (de unde se și face apelul, in while (currentTime < timeLimit)), golește conținutul containerelor, reumple cu noile date transmise ca parametru și redesenează.

5. Rezultate

În cadrul proiectului au fost făcute o serie de teste în care am variat mai mulți parametri, pentru a verifica funcționalitatea sistemului în diferite condiții. Deoarece documentul ar deveni mult prea greu de urmărit dacă am adăuga aici, vom expune în repository-ul proiectului conținutul Log of Events pentru trei setup-uri cerute de către client, și care vor fi afișate în Log.

Deoarece rezulatele simulării corespund cu cele așteptate, sistemul a trecut toate testele și este gata de utilizare.

6. Concluzii

În urma realizării acestei teme, au fost atinse concepte de POO legate de rularea în paralel a mai multor thread-uri. O adevărată provocare a constituit-o inserarea de task-uri în server-ul potrivit, în funcție de strategie, precum și excluderea acestui task din server, în urma procesării. Noțiuni despre implementarea graficii a unei aplicații au fost reamintite cu succes.

Există suficient loc de îmbunătățiri. Aplicația se poate îmbunătăți în primul rând grafic, autorul proiectului propunând o animație cursivă și cu multe obiecte din lumea reală. De asemenea, siguranța firelor de execuție (thread safety) și eficența aplicației pot fi îmbunătățite.

7. Bibliografie

1. *https://users.utcluj.ro/~igiosan/Resources/POO/Curs/POO11.pdf - Conf.univ.dr. Ion Augustin Giosan, cursul de Programare Orientată pe Obiect: cursul 11 – Fire de execuție (Threads)*
2. *http://users.utcluj.ro/~igiosan/Resources/POO/Curs/POO09.pdf - Conf.univ.dr. Ion Augustin Giosan, cursul de Programare Orientată pe Obiect: cursul 9 – Interfețe utilizator grafice (GUIs)*
3. *http://users.utcluj.ro/~igiosan/Resources/POO/Curs/POO07.pdf - Conf.univ.dr. Ion Augustin Giosan, cursul de Programare Orientată pe Obiect: cursul 7 – Erori și excepții în Java*
4. *https://www.baeldung.com/java-atomic-variables - Java Atomic Variables*
5. *https://www.digitalocean.com/community/tutorials/java-blockingqueue-example - Java BlockingQueue*
6. *https://www.youtube.com/watch?v=px4W-HXRWKk – Thread Safety in Java*
7. *https://www.youtube.com/watch?v=d3xb1Nj88pw – Java BlockingQueue*
8. *https://www.geeksforgeeks.org/how-to-add-custom-class-objects-to-the-treeset-in-java/ - Adding a custom class object in a TreeSet*
9. *https://www.educative.io/answers/how-to-generate-random-numbers-in-java - How to generate random numbers in Java*
10. *https://www.youtube.com/watch?v=ScUJx4aWRi0 - Java File Input/Output*
11. *https://mkyong.com/java/how-to-round-double-float-value-to-2-decimal-points-in-java/ Floating number with desired format in Java*
12. *https://docs.oracle.com/javase/tutorial/uiswing/components/slider.html - Sliders in Java*
13. *https://stackoverflow.com/questions/1234912/how-to-programmatically-close-a-jframe - How to programatically close a JFrame in Java*
14. *https://www.geeksforgeeks.org/java-swing-jprogressbar/ - About Progress Bars in Java*
15. *https://stackoverflow.com/questions/38349445/how-to-delete-all-components-in-a-jpanel-dynamically - How to delete all components in a JPanel dynamically*