DOCUMENTATIE

TEMA 2

NUME STUDENT: Rosca Andrei Bogdan

GRUPA: 30216

# CUPRINS

[1. Obiectivul temei 3](#_Toc95297885)

[2. Analiza problemei, modelare, scenarii, cazuri de utilizare 3](#_Toc95297886)

[3. Proiectare 3](#_Toc95297887)

[4. Implementare 3](#_Toc95297888)

[5. Rezultate 3](#_Toc95297889)

[6. Concluzii 3](#_Toc95297890)

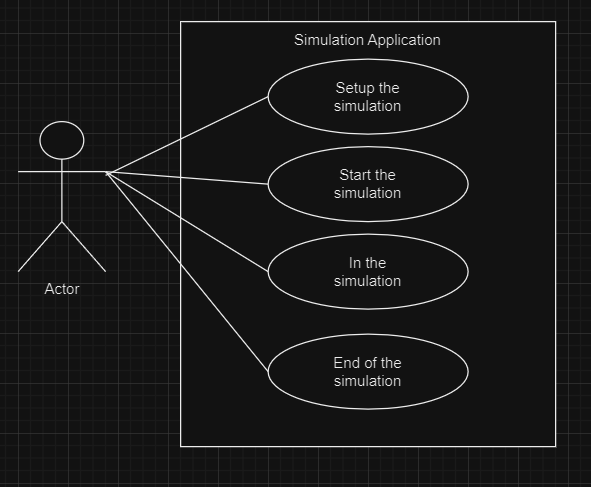
[7. Bibliografie 3](#_Toc95297891)

# Obiectivul temei

Obiectivul principal al proiectului constă în implementarea unei aplicații care să permită analiza unui sistem bazat pe cozi. Această analiză se realizează prin simularea sosirii unei serii de N clienți care doresc să beneficieze de un serviciu. Clienții vor intra într-un număr Q de cozi disponibile, unde vor aștepta să fie serviți. După ce sunt serviți, clienții vor părăsi cozile. În final, aplicația va calcula și va furniza timpul mediu de așteptare, timpul mediu de servire și ora de vârf în cadrul sistemului.Obiectivele secundare sunt analiza problemei si identificarea cerintelor,proiectarea aplicatiei , implementarea si testarea acesteia.

# Analiza problemei, modelare, scenarii, cazuri de utilizare

Scopul acestei teme este să ofere un sistem de gestionare a clientilor bazat pe cozi cu o interfață grafică intuitivă, care să permită utilizatorului să citească datele pentru simulare de la tastatura, să observe evolutia cozilor și să vizualizeze rezultatele. În același timp, se pune accent pe asigurarea unei experiențe de utilizare ușoare și plăcute pentru utilizator.



Caz de utilizare : configurarea simularii

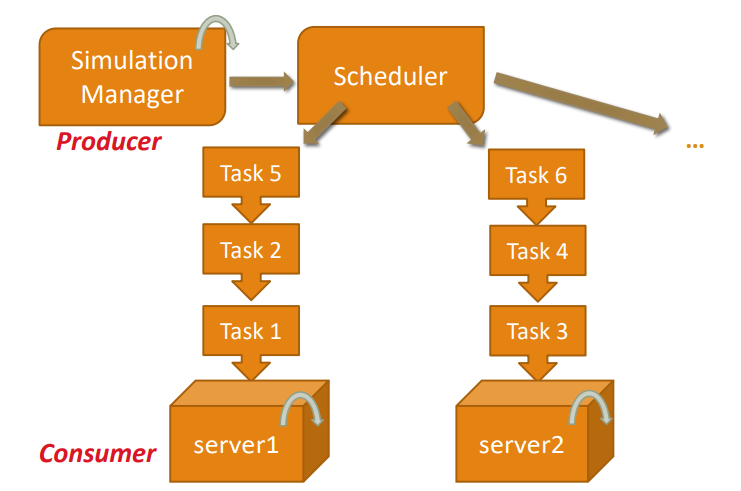
Actor principal: user

Scenariu de utilizare ideal:   
 1.Utilizatorul introduce valorile pentru: numărul de clienți, numărul de cozi, intervalul de simulare, timpul minim și maxim de sosire, și timpul minim și maxim de servire.  
 2.Utilizatorul apasă pe butonul de start pentru a incepe simularea .  
 3.Aplicația validează datele și afișează un mesaj informând utilizatorul să înceapă simularea.

Probleme ce pot fi intalnite: Utilizatorul introduce date gresite de configurare , in acest caz scenariul se intoarce la pasul 1

# Proiectare

Arhitectura programului



În cadrul diagramului de clase, sunt prezentate clasele utilizate în proiect, fiind organizate în trei pachete distincte în funcție de funcționalitatea lor. Pentru gestionarea cozilor, am optat să folosesc o clasă Task și o clasă Server. În Server, se păstrează toti clientii (reprezentate de obiecte de tip Task) ali unei cozi într-o colecție de tip BlockingQueue, iar prin intermediul metodei run, fiecare client este procesat.

Diagrama UML de pachete

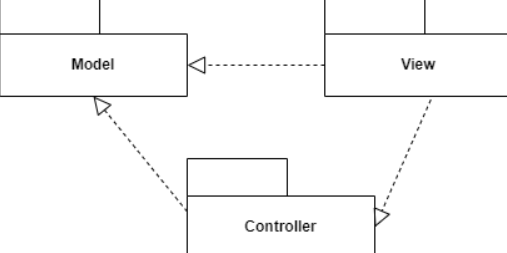
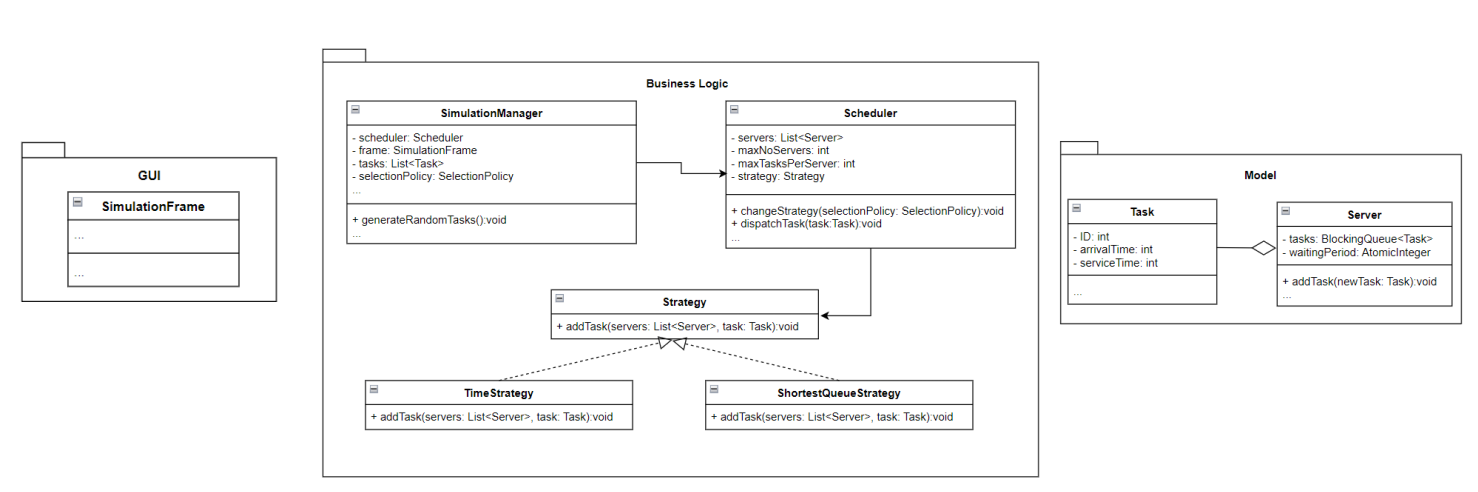


Diagrama UML de clase



# Implementare

Clasa Server reprezintă un server care gestionează clientii(tasks) într-o coadă. Aceasta implementează interfața Runnable, permițând utilizarea sa într-un fir de execuție separat, in acelasi timp cu alte fire de executie dechise.

**Atribute**:  
| -tasks: O colecție de tip BlockingQueue<Task> în care sunt stocati toti clientii de procesat.  
 -waitingPeriod: Un obiect de tip AtomicInteger care reprezintă perioada de așteptare totală a din coadă.

**Metode principale**:  
 -addTask(Task t): Adaugă o nouă sarcină în coadă. Actualizează și perioada de așteptare totală.  
 -toString(): Returnează o reprezentare sub formă de șir a sarcinilor din coadă.  
 -isEmpty(): Verifică dacă coada este goală.  
 -crestereWaitingTime(): Incrementarea timpului de așteptare pentru fiecare sarcină din coadă.  
 -run(): Metoda care se execută în mod continuu, procesând clientii din coadă. Pentru fiecare client, se reduce timpul de servire și se actualizează perioada de așteptare totală , se adauga la sumele care realizeaza timpul de asteptare mediu si timpul de procesare mediu



Clasa SimulationManager reprezintă managerul de simulare, care gestionează întregul proces de simulare a sistemului bazat pe cozi. Aceasta implementează interfața Runnable, permițând utilizarea sa într-un fir de execuție separat.

Atribute:  
 -timeLimit: Durata totală a simulării.  
 -maxProcessingTime: Timpul maxim de procesare al unei sarcini.  
 -maxArrivalTime: Timpul maxim de sosire al unei sarcini.  
 -minProcessingTime: Timpul minim de procesare al unei sarcini.  
 -numberOfServers: Numărul de servere disponibile.  
 -numberOfClients: Numărul total de clienți.  
 -currentTime: Momentul de timp curent în cadrul simulării.  
 -maxTime: Momentul de timp în care s-a înregistrat cel mai mare trafic în cozi.  
 -maxSize: Dimensiunea maximă a cozilor înregistrată în timpul simulării.  
 -writer: Obiect de tip FileWriter utilizat pentru scrierea într-un fișier.  
 -selectionPolicy: Politica de selecție a serverelor (ex: SHORTEST\_TIME, SHORTEST\_QUEUE, etc).  
 -scheduler: Obiect de tip Scheduler care se ocupă de gestionarea serverelor și a sarcinilor.  
 -generatedTasks: Listă de sarcini generate aleatoriu pentru simulare.  
 -form: Obiect de tip Interface care reprezintă interfața grafică a aplicației.

Metode principale:  
 -buton(): Metodă privată care adaugă un ascultător pentru butonul de pornire a simulării.  
 -generateNRandomTasks(): Generează aleatoriu sarcini pentru simulare, bazat pe parametrii specificați.  
 -afisareFisiere(String s): Scrie un șir de caractere într-un fișier.  
 -main(String args[]): Metodă principală care inițializează un obiect SimulationManager și pornește simularea.

-run(): Metodă care se execută în mod continuu și gestionează procesul de simulare.



Clasa Scheduler  
 - reprezintă un planificator responsabil pentru distribuirea sarcinilor către servere în funcție de o anumită strategie de selecție. Aceasta gestionează o listă de servere și implementează logica de planificare a sarcinilor.

Atribute:  
 -servers: Listă de servere disponibile pentru procesare.  
 -maxNoServers: Numărul maxim de servere.  
 -maxTasksPerServer: Numărul maxim de sarcini pe server.  
 -strategy: Strategia de selecție a serverului.

Metode principale:  
 -Scheduler(int maxNoServers, int maxTasksPerServer,int strateg): Constructorul clasei, în care se inițializează serverele și se pornește câte un fir de execuție pentru fiecare server.  
 -dispatchTask(Task t): Metodă pentru distribuirea unei sarcini către un server utilizând strategia curentă.

  
 -terminare(): Verifică dacă toate serverele sunt goale și toate sarcinile au fost procesate.

Clasa Interface este o componentă a interfeței utilizator pentru sistemul de gestionare pt cozi. Aceasta utilizează framework-ul Swing din Java pentru a crea elementele interfeței utilizator și a le lega de acțiunile utilizatorului prin intermediul unor ascultători de evenimente.

# Rezultate

*Se vor prezenta scenariile pentru testare. In cazul in care in cerinta temei se specifica sa se faca testare unitara cu utilitarul JUnit, se vor integra in acesta sectiune rezultate testari cu JUnit.*

# Concluzii

*Se vor prezenta concluziile, ce s-a invatat din tema, posibile de dezvoltari ulterioare.*

# Bibliografie

*Se vor adauga referintele care au fost consultate de student pe parcursul implementarii temei .*

*Exemplu:*

1. *Bruce Eckel, Thinking in Java (4th Edition), Publisher: Prentice Hall PTRUpper Saddle River, NJUnited States, ISBN:978-0-13-187248-6 Published:01 December 2005.*
2. *What are Java classes? -* [*www.tutorialspoint.com*](http://www.tutorialspoint.com)