

DOCUMENTATIE

TEMA NUMARUL 2

NUME STUDENT: SABĂU EMANUELA CRISTINA
GRUPA: 30224

CUPRINS

1. Obiectivul temei	3
1.1. Obiectivul principal	3
1.2. Obiectivele secundare	3
2. Analiza problemei, modelare, scenarii, cazuri de utilizare	4
2.1. Cerințe funcționale:.....	4
2.2. Cerințe non-funcționale:	4
2.3. Cazurile de utilizare	4
3. Proiectare	5
4. Implementare	7
4.1. Clase.....	7
4.2. Interfața utilizatorului (GUI):.....	9
5. Rezultate	9
6. Concluzii	10
7. Bibliografie	11

1. Obiectivul temei

1.1. Obiectivul principal

Simularea sosirii unui număr N de clienți pentru serviciu, aceștia intră în Q cozi, așteaptă, sunt serviți și în cele din urmă părăsesc cozile și Calcularea timpului mediu de așteptare, timpului mediu de serviciu și a orei de vârf.

1.2. Obiectivele secundare

- Analiza problemei și identificarea cerinței
- Proiectarea aplicației de simulare
- Implementarea aplicației de simulare
- Testarea aplicației de simulare

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

2.1. Cerințe funcționale:

- Simulare a sosirii și servirii clienților:
 - Să se poată simula sosirea unui număr specific de clienți în cozi.
 - Să se poată simula servirea clienților de către angajații din cozi.
 - Să se poată calcula timpul mediu de așteptare și timpul mediu de servire.
- Gestionarea coziilor:
 - Să se poată crea un număr specific de cozi pentru a servi clienții.
 - Să se poată adăuga clienți în cozi în mod dinamic.
 - Să se poată elimina clienți din cozi atunci când sunt serviți.

2.2. Cerințe non-funcționale:

- Performanță:
 - Aplicația trebuie să poată simula sisteme cu un număr mare de clienți și cozi într-un timp rezonabil.
 - Interfața utilizatorului trebuie să fie responsivă și să ofere feedback rapid.
- Fiabilitate:
 - Aplicația trebuie să fie robustă și să gestioneze corect toate situațiile, inclusiv cele de excepție.
 - Datele trebuie să fie stocate și manipulate în mod sigur pentru a evita pierderea informațiilor.

2.3. Cazurile de utilizare

1. Simulare sosire clienți:

Utilizatorul introduce numărul de clienți ce urmează să sosească.

Aplicația generează sosirile clienților în cozi conform distribuției specificate.

2. Simulare servire clienți:

Angajații din cozi servesc clienții în ordinea sosirii lor.

Timpul de servire al fiecărui client este generat aleator și conform distribuției specificate.

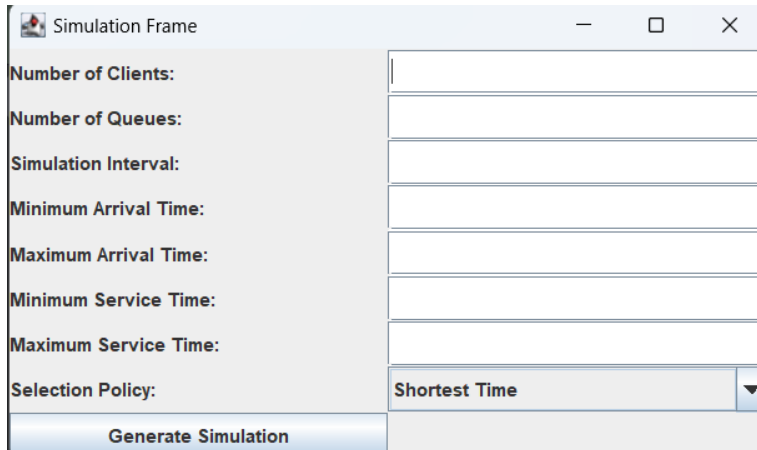
3. Calcul timp mediu de așteptare și timp mediu de servire:

La finalul simulării, aplicația calculează timpul mediu de așteptare și timpul mediu de servire pentru clienți.

Aceste cerințe funcționale și non-funcționale, împreună cu cazurile de utilizare, vor servi drept ghid pentru proiectarea și implementarea aplicației de simulare a sistemelor bazate pe cozi.

3. Proiectare

Când dăm run la program apare o fereastră cu interfața utilizatorului. Pentru acestea avem câteva câmpuri text care vor fi completate de utilizator. Cu datele introduse se va rula programul.



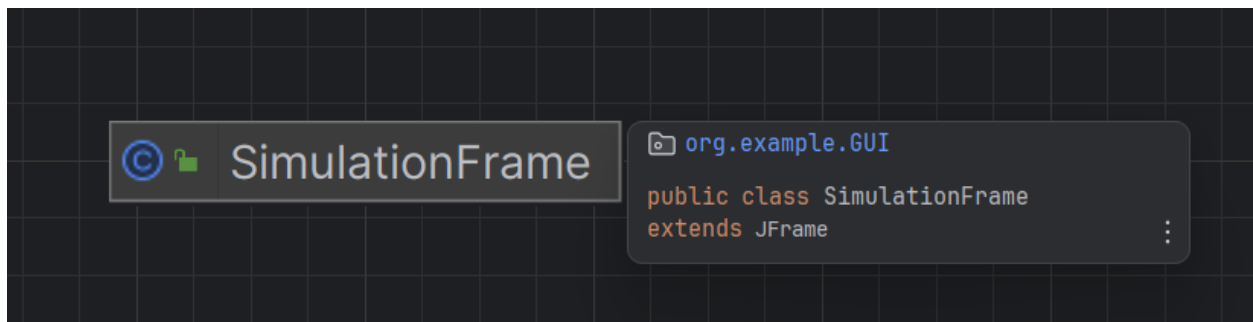
The screenshot shows a window titled "Simulation Frame" with standard Windows window controls (minimize, maximize, close). The window contains several input fields for simulation parameters:

- Number of Clients:
- Number of Queues:
- Simulation Interval:
- Minimum Arrival Time:
- Maximum Arrival Time:
- Minimum Service Time:
- Maximum Service Time:
- Selection Policy: (with a dropdown menu showing "Shortest Time")

At the bottom of the window is a button labeled "Generate Simulation".

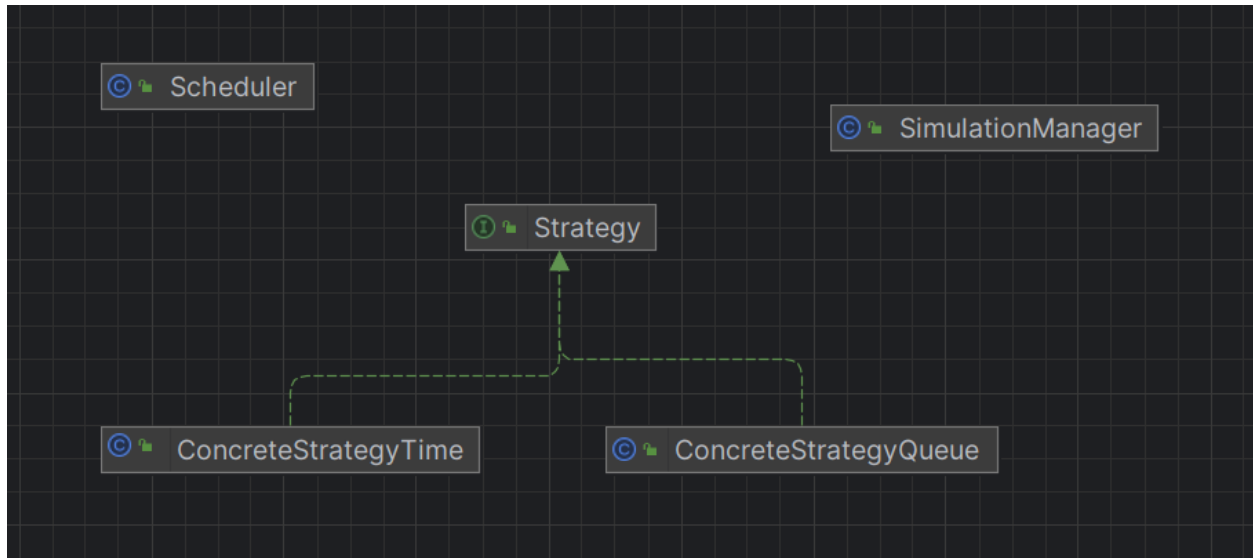
Avem și o repartizare a claselor în 3 pachete mari și diagramele UML pentru acestea.

- GUI



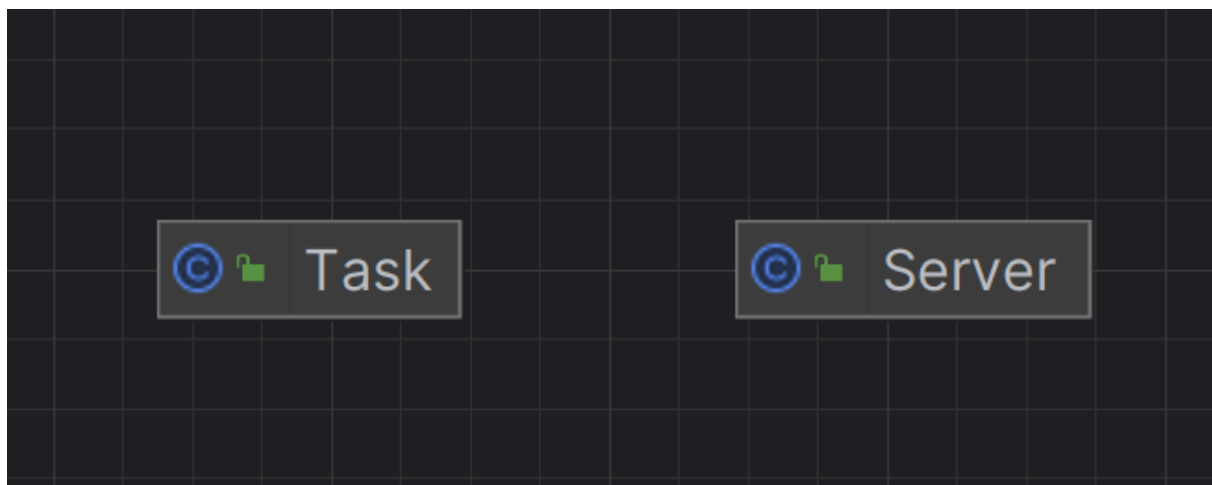
Pentru acest pachet avem clasa în care este scris programul pentru interfața utilizatorului.

- LOGIC



Aici putem observa folosirea unei interfațe numite Strategy care definește o strategie. În cazul nostru sunt două strategii pe care le folosim și anume ConcreteStrategyTime și ConcreteStrategyQueue.

- MODEL



În acest pachet avem definite structura unui task și a unui server pentru a putea fi folosite cu ușurință în logica programului.

4. Implementare

4.1. Clasele implementate

- **Clasa SimulationManager:**

Atribute:

- timeLimit: Intervalul de timp al simulării.
- maxProcessingTime: Timpul maxim de servire.
- minProcessingTime: Timpul minim de servire.
- minArrivalTime: Timpul minim de sosire al clienților.
- maxArrivalTime: Timpul maxim de sosire al clienților.
- numberOfServers: Numărul de servere disponibile (cozi).
- numberOfClients: Numărul total de clienți.
- generatedTasks: Lista de sarcini generate pentru simulare.
- averageWaitingTime: Timpul mediu de așteptare.
- averageServiceTime: Timpul mediu de servire.
- peakHour: Ora de vârf în intervalul de simulare.

Metode:

- calculateTimesFromSimulation(int serviceTime): Calculează timpul de servire mediu din simulare.
- generateNRandomTasks(int numberOfClients, int minTimeArrival, int maxTimeArrival, int minTimeServ, int maxTimeServ): Generează sarcini aleatorii pentru simulare.
- calculateAverageWaitingTime(): Calculează timpul mediu de așteptare.
- getPeakHourFromSimulation(int peakHourTime, int currentTime): Obține ora de vârf din simulare.
- dispatchTasks(int currentTime): Distribuie sarcinile către servere în funcție de timpul de sosire.
- run(): Implementează metoda run() din interfața Runnable, care inițiază și gestionează simularea.

- **Clasa Scheduler**

Attribute:

- servers: Lista de servere disponibile în sistem.
- maxNoServers: Numărul maxim de servere permise.
- maxTaskPerServer: Numărul maxim de sarcini pe care poate să le aibă fiecare server.
- strategy: Strategia de distribuire a sarcinilor către servere.

Metode:

- changeStrategy(SelectionPolicy policy): Metodă pentru schimbarea strategiei de distribuire a sarcinilor.
- dispatchTask(Task t): Metodă pentru distribuirea unei sarcini către servere folosind strategia specificată.

- **Clasa Server**

Attribute:

- tasks: Coada blocantă de sarcini pentru acest server.
- waitingPeriod: Perioada totală de așteptare a sarcinilor la acest server.

Metode:

- Server(): Constructor care inițializează coada de sarcini și perioada de așteptare.
- addTask(Task newTask): Metodă pentru adăugarea unei sarcini la coada de sarcini a serverului.
- run(): Metodă implementată din interfața Runnable, care gestionează procesul de servire a sarcinilor.
- getQueueSize(): Metodă pentru obținerea dimensiunii cozii de sarcini a serverului.

- **Clasa Task**

Attribute:

- id: Identificatorul unic al sarcinii.
- arrivalTime: Timpul de sosire al sarcinii.
- processingTime: Timpul necesar pentru procesarea sarcinii.

4.2. Interfața utilizatorului (GUI):

- **Componente:**

- Câmpuri text pentru introducerea parametrilor simulării (numărul de clienți, numărul de cozi, intervalele de timp etc.).
- Meniu derulant pentru selectarea politicii de selecție a coziilor.
- Buton pentru generarea și pornirea simulării.
- Posibil un panou pentru afișarea evoluției cozilor în timp real.

- **Interacțiune:**

- Utilizatorul introduce parametrii și selectează opțiunile dorite.
- Apasă butonul de generare a simulării.
- Aplicația inițiază simularea și afișează rezultatele sau evoluția cozilor în interfața utilizatorului.

5. Rezultate

Pentru testarea problemei am avut la dispoziție 3 exemple, mai specific cele din figura de mai jos.

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

Pentru aceste exemple s-au salvat câte un fișier text cu denumirea “Test 1.txt”, “Test 2.txt”, respectiv “Test 3.txt” în directorul proiectului.

6. Concluzii

Pentru realizarea acestei temă am învățat cum să lucrez cu thread-urile. Funcționalitatea programului a depins foarte mult de acest aspect. Aceasta ne-a permis să lucrăm cu mai multe cozi în paralel. Fără utilizarea firelor de execuție acest lucru ar fi probabil imposibil.

În ceea ce privește posibilele dezvoltări ale aplicației există câteva aspect care ar putea fi luate în considerare cum ar fi implementarea unei interfețe de utilizator mai interactive și intuitive, care să ofere mai multe opțiuni de vizualizare a rezultatelor simulării și să permită modificarea parametrilor în timp real.

O altă posibilă dezvoltare ar fi optimizarea performanței aplicației pentru a gestiona eficient un număr mare de clienți și cozi, eventual prin utilizarea de tehnici de paralelizare sau de optimizare a algoritmilor.

7. Bibliografie

1. Concurrency - docs.oracle.com/javase/tutorial/essential/concurrency/
2. Brian Goetz, Tim Peierls, Joshua Bloch, Joseph Bowbeer, David Holmes, and Doug Lea, Java Concurrency in Practice, Addison Wesley, Pearson Education
3. K. Sharan, Beginning Java 8 Language Features: Lambda Expressions, Inner Classes, Threads, I/O, Collections, and Streams 1st Edition, APRESS, 2014.
4. Java Thread Pool Example - www.javacodegeeks.com