DOCUMENTATIE

TEMA 2

CURCANU SAMUEL-ADRIAN

GRUPA:30227

# 

# CUPRINS

[1. Obiectivul temei 3](file:///C:\Users\Samuel-Adrian\Downloads\PT2023_Template_Documentatie_RO%20(1).doc#_Toc95297885)

[2. Analiza problemei, modelare, scenarii, cazuri de utilizare 3](file:///C:\Users\Samuel-Adrian\Downloads\PT2023_Template_Documentatie_RO%20(1).doc#_Toc95297886)

[3. Proiectare 3](file:///C:\Users\Samuel-Adrian\Downloads\PT2023_Template_Documentatie_RO%20(1).doc#_Toc95297887)

4. Implementare…………………...……………………………………………………………………………..4

5. Rezultate………………………………………………………………………………………………………5

6. Concluzii……………………………………………………………………………………………………...5

7. Bibliografie………………………………………………………………………………………………...…5

**1.Obiectivul temei**

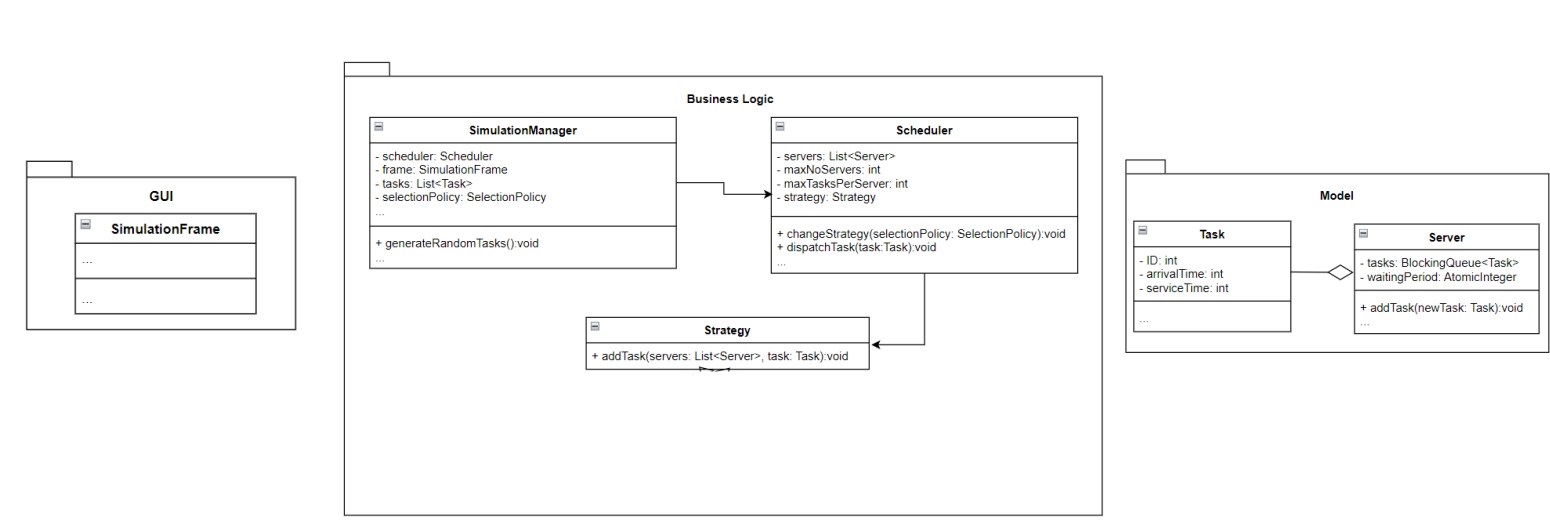
Proiectează și implementează o aplicație de gestionare a cozilor care atribuie clienților cozi astfel încât timpul de așteptare să fie minimizat. Cozile sunt folosite în mod obișnuit pentru a modela domenii din lumea reală. Obiectivul principal al unei cozi este de a furniza un loc pentru un "client" să aștepte înainte de a primi un "serviciu". Managementul sistemelor bazate pe cozi are interesul de a minimiza timpul de așteptare al "cliștilor" în cozi înainte de a fi serviți. O modalitate de a minimiza timpul de așteptare este de a adăuga mai multe servere, adică mai multe cozi în sistem (fiecare coadă este considerată ca având un procesor asociat), dar această abordare crește costurile furnizorului de servicii. Aplicația de gestionare a cozilor ar trebui să simuleze (prin definirea unei durate de simulare 𝑡𝑠𝑖𝑚𝑢𝑙𝑎𝑡𝑖𝑜𝑛) o serie de N clienți care sosesc pentru servicii, intră în Q cozi, așteaptă, sunt serviți și părăsesc în cele din urmă cozile. Toți clienții sunt generați când simularea începe și sunt caracterizați de trei parametri: ID (un număr între 1 și N), 𝑡𝑎𝑟𝑟𝑖𝑣𝑎𝑙 (timpul de simulare când sunt pregătiți să intre în coadă) și 𝑡𝑠𝑒𝑟𝑣𝑖𝑐𝑒 (intervalul de timp sau durata necesară pentru a servi clientul; adică timpul de așteptare atunci când clientul se află în fața cozii). Aplicația urmărește timpul total petrecut de fiecare client în cozi și calculează timpul mediu de așteptare. Fiecare client este adăugat la coadă cu timpul minim de așteptare atunci când timpul său 𝑡𝑎𝑟𝑟𝑖𝑣𝑎𝑙 este mai mare sau egal decât timpul de simulare (𝑡𝑎𝑟𝑟𝑖𝑣𝑎𝑙 ≥ 𝑡𝑠𝑖𝑚𝑢𝑙𝑎𝑡𝑖𝑜𝑛).

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

Pentru functionarea corecta a problemei trebuie sa avem in vedere faptul ca timpul minim de sosire trebuie sa fie mai mic decat timpul maxim de sosire, la fel ca si cel pentru timpul necesar fiecarui serviciu. Trebuie sa avem un interval correct in ambele cazuri.

**3.Proiectare**

Tema este creeata in stil OOP, variabilele sunt toate private putand fi accesate doar prin metode de get si set. Pentru o vizualizare cat mai buna clasele sunt in pachete diferite avand un nume corespunzator, pentru a ajuta utilizatorul sa gaseasca portiuni de cod dorite cat mai usor.



**4.Implementare**

Clasa "Client" și este parte din pachetul "Model". Aceasta reprezintă un obiect de tip client, care are trei proprietăți: id-ul clientului, timpul de sosire și timpul necesar pentru a fi servit. Clasa are un constructor care primește ca parametri valorile pentru cele trei proprietăți și metode get și set pentru fiecare dintre ele. De asemenea, aceasta include o metodă "toString()" care returnează o reprezentare sub forma unui șir de caractere a obiectului Client, care conține id-ul, timpul de sosire și timpul de servire al clientului.

Clasa “QueueClass” reprezintă o coadă de clienți, implementată folosind clasa `BlockingQueue` din Java. Clasa are un atribut `queue` de tip `BlockingQueue<Client>` în care se stochează clienții, un atribut `waitingPeriod` de tip `AtomicInteger` care reprezintă perioada totală de așteptare a clienților din coadă și un atribut `number` care reprezintă numărul acestei cozi. Clasa `QueueClass` implementează interfața `Runnable` și are o metodă `run()` care este folosită în cadrul unui fir de execuție pentru a trata clienții din coadă. Metoda `addClient()` adaugă un client în coadă, iar `removeClient()` îl elimină pe cel specificat din coadă.

Clasa "Scheduler" este responsabilă de gestionarea listei de cozi și de atribuirea clienților către coada potrivită. Are un constructor care primește ca parametri numărul maxim de cozi și numărul maxim de clienți pe coadă. Are o listă de obiecte QueueClass care reprezintă cozi și o strategie de alocare a clienților la cozi. Metoda "dispatchClient" este folosită pentru a alege coada potrivită în care să fie plasat noul client.

Clasa “Strategy” implementează strategia de asignare a clienților la cozi. Metoda `addClient` primește lista de cozi și un obiect Client pe care îl va asigna uneia dintre cozi în funcție de strategia implementată. Mai întâi, metoda verifică fiecare coadă și încearcă să adauge clientul în prima coadă goală pe care o găsește. Dacă toate cozile sunt pline, metoda va căuta coada cu cel mai mic timp de așteptare și va adăuga clientul în acea coadă. Aceasta va continua până când clientul va fi asignat la o coadă.

Clasa “SimulationManager” este alcătuita dintr-un manager de simulare și un planificator. Managerul de simulare este responsabil pentru generarea de clienți aleatori și inițializarea planificatorului cu cozi și clienți. Planificatorul este responsabil pentru gestionarea cozi și procesarea clienților, în funcție de regulile definite de utilizator.

Mai exact, managerul de simulare primește parametrii pentru configurarea simulării, cum ar fi limita de timp, timpul maxim și minim de procesare, numărul de cozi și numărul de clienți. Apoi generează clienți aleatorii cu timp de sosire și timp de procesare aleatorii, sortându-i în ordine crescătoare a timpului de sosire. Acești clienți sunt trimiși apoi către planificator, care îi procesează în funcție de regulile stabilite. În timpul simulării, managerul actualizează starea planificatorului la fiecare unitate de timp și afișează starea într-un fișier text și pe interfața grafică a utilizatorului. Managerul și planificatorul calculează, de asemenea, timpul mediu de așteptare și timpul mediu de servire și determină ora de vârf, care reprezintă momentul în care numărul maxim de clienți așteaptă în coadă. Aceste valori sunt afișate și pe interfața grafică a utilizatorului.

**5.Rezultate**

Se va afisa la fiecare moment de timp lista de cozi cu fiecare numar de client in parte, codul se va executa pana la finalul perioadei de timp pe care am dat-o chiar daca nu mai avem clienti de adaugat.

**6.Concluzii**

Tema a fost una destul de complexa punand la testare cunostintele de java, de asemenea aceasta tema se mai poate dezvolta ulterior pentru a o face mult mai complexa, cum ar fi un management la o scara mult mai mare, mai multe cozi cu taxe si multe alte lucruri complexe.

**7.Bibliografie**

1. FUNDAMENTAL PROGRAMMING TECHNIQUES LABORATORY GUIDE - <https://dsrl.eu/courses/pt/>

2. http://docs.oracle.com/javase/tutorial/essential/concurrency/index.html - 3.http://www.tutorialspoint.com/java/util/timer\_schedule\_period.htm – 4.http://www.javacodegeeks.com/2013/01/java-thread-pool-example-using-executors-andthreadpoolexecutor.html