DOCUMENTATIE

**Tema 2 – QUEUES MANAGEMENT**

**NUME STUDENT: SCUTURICI VLAD LUCIAN**

**GRUPA: 30225**

# CUPRINS

1. Obiectivul temei ………………………………………………………………………3

2. Analiza problemei, modelare, scenarii, cazuri de utilizare …………………………...3

3. Proiectare ……………………………………………………………………………...4

4. Implementare ………………………………………………………………………….5

5. Rezultate ………………………………………………………………………………8

6. Concluzii ………………………………………………………………………………9

7. Bibliografie ……………………………………………………………………………10

Obiectivul Temei

Obiectivul acestei teme este de a proiecta și implementa o aplicație de gestionare a cozilor care să permită atribuirea clienților la cozi astfel încât timpul de așteptare să fie minimizat. Cozile sunt utilizate în mod obișnuit pentru a modela domenii din lumea reală, iar obiectivul principal al unui sistem bazat pe cozi este de a oferi un loc pentru "client" să aștepte înainte de a primi un "serviciu".

Aplicația de gestionare a cozilor ar trebui să simuleze o serie de N clienți care sosesc pentru a primi un serviciu, intră în Q cozi, așteaptă, sunt serviți și în cele din urmă părăsesc cozile. Toți clienții sunt generați la începutul simulării și sunt caracterizați prin trei parametri: ID (un număr între 1 și N), timpul de sosire și timpul de serviciu. Aplicația urmărește timpul total petrecut de fiecare client în cozi și calculează timpul mediu de așteptare. Utilizatorul trebuie să introducă numărul de clienți, numărul de cozi, intervalul de simulare, timpul minim și maxim de sosire a clienților și timpul minim și maxim de serviciu.

Proiectare

Diagrama UML (Unified Modeling Language) este o modalitate standard de a reprezenta grafic elementele de proiectare ale unui sistem software. Această diagramă oferă o viziune de ansamblu asupra structurii și comportamentului unui sistem software, precum și a relațiilor dintre elementele acestuia. Diagrama UML poate fi utilizată pentru a defini modele de proiectare, pentru a planifica implementarea sistemelor software și pentru a documenta procesul de dezvoltare a acestora. Această diagramă este formată dintr-un set de simboluri și relații care descriu elementele componente ale sistemului și modul în care acestea comunică între ele. Diagrama UML este o unealtă importantă pentru proiectarea și dezvoltarea de sisteme software complexe.

A picture containing text

Description automatically generated

BlockingQueue este o interfata din Java care defineste o coada de elemente in care adaugarea sau eliminarea acestora poate fi blocata in cazul in care coada este deja plina sau goala. Aceasta interfata este folosita pentru a gestiona comunicarea intre mai multe thread-uri, oferind o metoda sigura si eficienta de sincronizare intre producatori si consumatori.

Pentru a asigura o sincronizare sigura intre thread-uri, interfata BlockingQueue include metode precum put() si take(), care blocheaza executia firului de executie pana cand exista spatiu pentru adaugarea unui element nou sau pana cand exista cel putin un element disponibil pentru eliminare. Aceasta abordare elimina riscul de supraincarcare sau blocare a cozii, asigurand o comunicare eficienta si fara probleme intre producatori si consumatori. In plus, exista mai multe implementari ale acestei interfete in Java, fiecare cu propriile caracteristici si utilizari specifice, precum ArrayBlockingQueue, LinkedBlockingQueue sau PriorityBlockingQueue.

Vom implementa acest simulator cu folosindu-ne de thread-uri, pentru a obtine un timp cat mai rapid si o rezolvare cat mai compacta. Thread-urile sunt o componenta esentiala a programarii concurente in Java. Un thread reprezinta o modalitate prin care un program poate executa mai multe activitati simultan, prin crearea unui nou fir de executie. In Java, thread-urile sunt implementate prin intermediul clasei Thread, care ofera metode pentru crearea, pornirea si sincronizarea acestora. Exista, de asemenea, si alte clase si interfete, cum ar fi Runnable si Callable, care ofera alternative sau extensii pentru utilizarea thread-urilor in Java.

Thread-urile pot fi utilizate intr-un spectru larg de aplicatii, de la jocuri si aplicatii multimedia, pana la aplicatii de retea si servicii web. Acestea permit programatorilor sa optimizeze si sa imbunatateasca performanta aplicatiilor prin impartirea sarcinilor in mai multe thread-uri de executie. De exemplu, un program poate utiliza un fir de executie pentru a citi datele de la o sursa de date si un alt fir de executie pentru a procesa aceste date in paralel. Prin utilizarea thread-urilor, aplicatiile pot fi mai responsive si pot raspunde mai rapid la evenimente si interactiuni ale utilizatorului.

Implementare

Proiectul este alcatuit din urmatoarele clase si pachete:

I) Pachetul Model

1. Task

Clasa Task descrie o sarcina ce poate fi procesata de un sistem. Aceasta contine informatii precum id-ul sarcinii, momentul de timp la care a ajuns sarcina, timpul necesar pentru procesarea acesteia si timpul de asteptare al sarcinii in coada. Metodele acestei clase permit decrementarea timpului de procesare, incrementarea timpului de asteptare si obtinerea informatiilor despre sarcina, cum ar fi timpul de asteptare, timpul necesar de procesare si momentul de timp la care a ajuns sarcina. Metoda toString() permite reprezentarea sarcinii sub forma de sir de caractere.

2. Server

Clasa Server reprezinta un server ce poate procesa sarcini. Fiecare server are o coada de sarcini si un timp de asteptare. Metodele acestei clase permit adaugarea unei sarcini in coada serverului, preluarea si procesarea sarcinilor, actualizarea starii sarcinilor, obtinerea informatiilor despre sarcinile din coada serverului si afisarea acestora intr-un fisier si pe interfata grafica. Metoda run() este metoda principala a clasei si preia sarcinile din coada, opreste firul de executie pentru durata de procesare a sarcinii si actualizeaza timpul de asteptare al serverului. Metoda update() actualizeaza starea sarcinilor din coada. Metoda displayTasks() afiseaza sarcinile din coada intr-un fisier si pe interfata grafica, in timp ce metodele getWaitingPeriod(), getTasks() si isEmpty() returneaza informatii despre starea serverului.

II) Pachetul BusinessLogic

1. Scheduler

Clasa Scheduler gestioneaza serverele si procesarea sarcinilor. Aceasta contine o lista de servere, un numar maxim de servere si un numar maxim de sarcini per server, o strategie de distribuire a sarcinilor intre servere, un numar de clienti si un manager de simulare. Metodele acestei clase permit initializarea serverelor, schimbarea strategiei de distribuire a sarcinilor, adaugarea sarcinilor in coada serverelor, obtinerea informatiilor despre starea serverelor, afisarea informatiilor despre starea serverelor si numarul total de sarcini. Metoda dispatchTask() foloseste strategia selectata pentru a adauga sarcina in coada serverului ales. Metoda changeStrategy() aplica sablonul de proiectare Strategy pentru a instantia strategia cu strategia concreta corespunzatoare politicii de selectie.

2. SimulationManager

Clasa SimulationManager este clasa principala a aplicatiei. Aceasta gestioneaza simularea si interactiunea cu utilizatorul. Clasa contine un numar de servere, un numar de clienti, o politica de selectie, un manager de simulare si o lista de sarcini generate aleator. Metodele acestei clase permit initializarea serverelor si generarea sarcinilor, adaugarea timpului de asteptare la timpul total de asteptare, calcularea timpului mediu de asteptare si a timpului mediu de procesare, afisarea sarcinilor in asteptare, desfasurarea simularii si afisarea rezultatelor. Metoda run() implementeaza simularea si se executa intr-un fir de executie separat. Metoda startSimulation() initializeaza si porneste simularea cu parametrii specificati. Metoda main() porneste aplicatia prin crearea unei instante a clasei SimulationFrame.

3, 4, 5. SelectionPolicy, ConcreteStrategyTime, ConcreteStrategyQueue

Aceste trei clase sunt utilizate in alegerea criteriului de distribuire. Clasa SelectionPolicy enumereaza cele doua politici de selectie disponibile - SHORTEST\_TIME si SHORTEST\_QUEUE. Clasa ConcreteStrategyTime implementeaza strategia SHORTEST\_TIME, selectand serverul cu cel mai mic timp de asteptare pentru a procesa urmatoarea sarcina. Clasa ConcreteStrategyQueue implementeaza strategia SHORTEST\_QUEUE, selectand serverul cu cea mai scurta coada de asteptare pentru a procesa urmatoarea sarcina. Aceste doua strategii sunt folosite pentru a decide ce server va procesa urmatoarea sarcina si sunt folosite de clasa Scheduler pentru a distribui sarcinile primite de la clienti catre servere.

III) Pachetul GUI

1. SimulationFrame

Aceasta clasa reprezinta fereastra principala a aplicatiei de simulare a unei cozi. Contine un set de componente grafice, cum ar fi campuri text, butoane si o zona de text pentru log-uri, care permit utilizatorului sa introduca parametrii pentru simulare si sa afiseze rezultatele. De asemenea, clasa contine si logica necesara pentru a initializa componentele si a gestiona evenimentele utilizatorului, cum ar fi apasarea butonului de start simulare sau selectarea unui test predefinit.

Rezultate

Graphical user interface, application, table

Description automatically generatedGraphical user interface, application, table

Description automatically generatedGraphical user interface, application, table

Description automatically generatedText

Description automatically generated with medium confidence

Concluzii

Realizarea acestui proiect de simulare a unui sistem de cozi a fost o experiență valoroasă și educativă, care a implicat o serie de competențe precum programare orientată obiect, algoritmi și structuri de date, dezvoltarea de interfețe grafice cu utilizatorul și abilități de gestionare a proiectelor. Proiectul a oferit o oportunitate excelentă de a aplica cunoștințele teoretice într-un proiect practic și de a dezvolta abilități și experiență practică în dezvoltarea de software. În general, acest proiect a fost o modalitate excelentă de a exersa și consolida abilitățile de programare, precum și de a crea un produs final care poate fi utilizat într-un mediu real.

Bibliografie

1. https://dsrl.eu/courses/pt/ - cerinte, prezentari, exemple

2. https://www.javatpoint.com/java-swing - tutorial javaswing