MINISTERUL EDUCAȚIEI ȘI CERCETĂRII ȘTIINȚIFICE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE
CATEDRA CALCULATOARE

DOCUMENTATIE

TEMA *2*

Teglă Lidia-Sorina

CUPRINS

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

1. Obiectivul temei

Obiectivul principal al temei este de a proiecta și implementa o aplicație care vizează analiza sistemelor bazate pe cozi prin (1) simularea unei serii de N clienți care sosesc pentru a primi servicii, intră în Q cozi, așteaptă, sunt serviți și în cele din urmă părăsesc cozile, și (2) calcularea timpului mediu de așteptare, timpului mediu de servire și a orei de vârf.

Obiective secundare:

• Analiza problemei și identificarea cerinetlor
• Proiectarea aplicatiei de simulare
• Implementarea aplicatiei de simulare
• Testarea aplicatiei de simulare

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

Caz de utilizare: Configurarea simulării
Actor principal: Utilizator
Scenariu principal de succes:

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 validare a datelor de intrare.
3. Aplicația validează datele și afișează un mesaj informativ pentru utilizator pentru a începe simularea.

Secvență alternativă: valori invalide pentru parametrii de configurare

- Utilizatorul introduce valori invalide pentru parametrii de configurare ai aplicației
- Aplicația afișează un mesaj de eroare și solicită utilizatorului să introducă valori valide
- Scenariul revine la pasul 1.

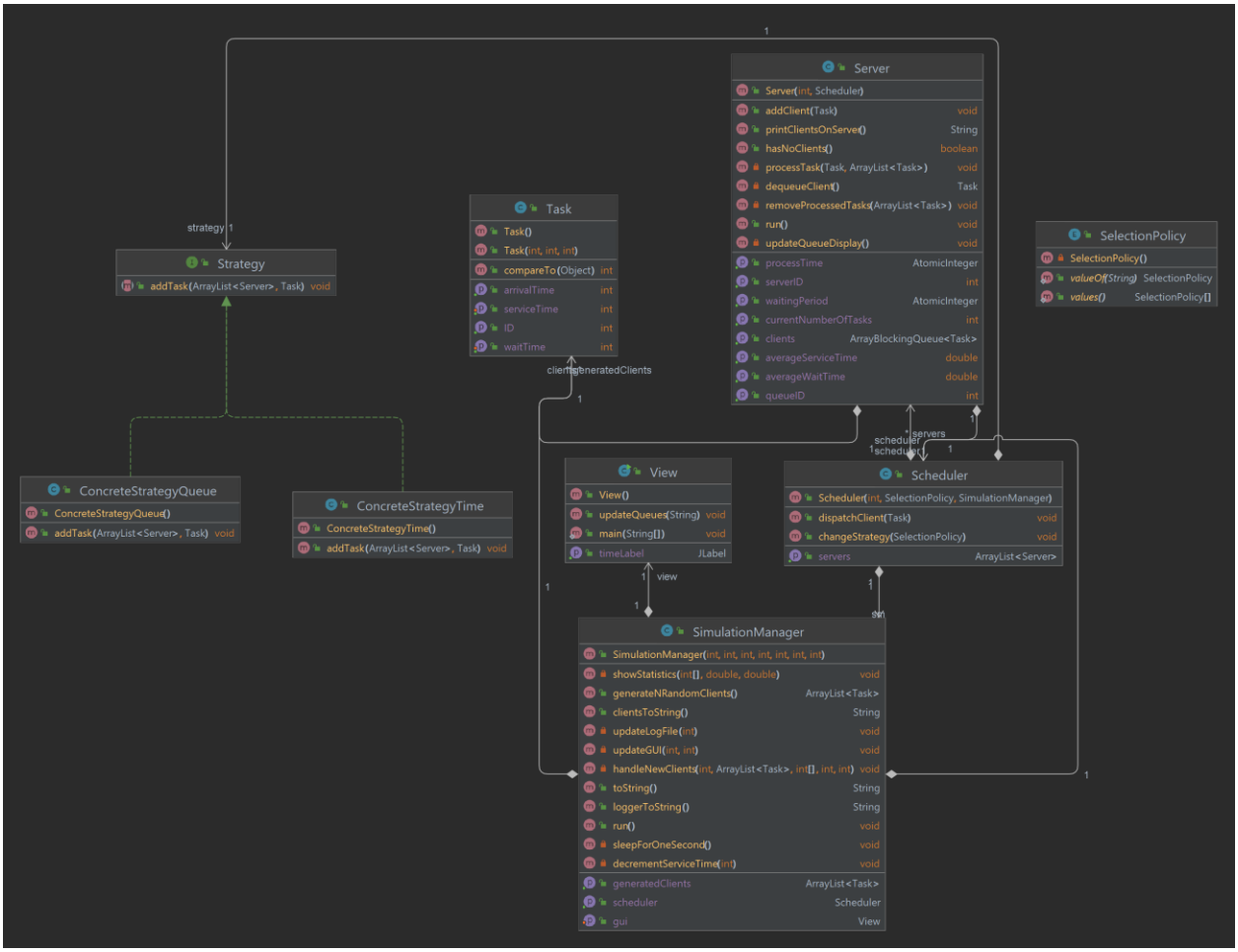
Cerințe funcționale:

- Aplicația de simulare ar trebui să permită utilizatorilor să configureze simularea.
- Aplicația de simulare ar trebui să permită utilizatorilor să înceapă simularea.
- Aplicația de simulare ar trebui să afișeze evoluția cozilor în timp real.

Cerințe non-funcționale:

- Aplicația de simulare ar trebui să fie intuitivă și ușor de utilizat de către utilizator.

3. Proiectare



4. Implementare

Clasa SimulationManager este o componenta esențiala a aplicației de gestionare a cozilor, responsabilă de generarea și gestionarea simulării sarcinilor primite. Clasa utilizează un generator de numere aleatoare pentru a simula generarea sarcinilor, atribuind timpuri de sosire și de serviciu aleatoare fiecărei sarcini generate.

Buclele de simulare ale clasei urmăresc timpul curent și apelează programatorul de sarcini pentru a trimite sarcinile generate. Acest lucru asigură că sarcinile sunt procesate în ordinea sosirii lor și li se alocă timpul corespunzător pentru servicii. În plus, clasa SimulationManager actualizează interfața utilizatorului pentru a afișa starea curentă a cozii, permițând utilizatorilor să monitorizeze simularea în timp real.

Clasa Scheduler joacă un rol critic în aplicația de gestionare a cozilor, prin transmiterea sarcinilor către servere conform strategiei stabilite. Această clasă este responsabilă pentru analizarea cozii și decizia care server este cel mai potrivit pentru a gestiona sarcina în funcție de factori precum disponibilitatea serverului și încărcarea de lucru.

Odată ce Scheduler-ul a determinat serverul potrivit, trimite sarcina către acel server, inițiind procesul de servisare. Aceasta asigură procesarea eficientă a sarcinilor, minimizând timpul de așteptare și maximizând utilizarea resurselor.

Clasa Task reprezintă o singură unitate de lucru în aplicația de gestionare a cozilor. Această clasă conține proprietăți esențiale, cum ar fi timpul de sosire și timpul de serviciu, care sunt folosite pentru a simula procesarea sarcinilor de către servere. În plus, clasa Task poate conține proprietăți suplimentare, cum ar fi prioritatea sarcinii, care poate fi utilizată pentru a influența strategia de distribuire a clasei Scheduler.

Clasa Server reprezintă un server fizic în aplicația de gestionare a cozilor. Această clasă conține proprietăți, cum ar fi capacitatea și încărcarea serverului, care sunt folosite pentru a determina disponibilitatea serverelor pentru procesarea sarcinilor. În plus, clasa Server poate conține proprietăți suplimentare, cum ar fi tipul de server, care poate fi utilizat pentru a informa strategia de distribuire a clasei Scheduler.

Clasele Strategy, ConcreteStrategyTime si ConcreteStrategyQueue oferă un cadru pentru implementarea diferitelor strategii de distribuire în aplicația de gestionare a cozilor. Clasa Strategy definește o interfață comună pentru toate strategiile de distribuire, în timp ce clasele ConcreteStrategy implementează strategii specifice bazate pe nevoile aplicației.

Clasa View este responsabilă de crearea interfeței grafice pentru aplicație. Aceasta extinde clasa JFrame și definește diverse componente UI, cum ar fi etichete, câmpuri de text, butoane și butoane radio, aranjate într-un GridBagLayout. De asemenea, include un JTextArea pentru a afișa starea cozilor din simulare.

Clasa are un constructor care configurează componentele UI și le inițializează cu valorile implicite. De asemenea, include un ActionListener pentru butonul Start care începe simularea și actualizează starea cozilor în JTextArea. Metoda updateQueues() este folosită pentru a actualiza JTextArea cu starea curentă a cozilor.

5. Rezultate.

Test 1	*Test 2*	*Test 3*
N = 4 Q = 2 $tsimulation\ MAX$ = 60 seconds [$tarrival\ MIN$,$tarrival\ MAX$] = [2, 30] [$tservice\ MIN$,$tservice\ MAX$]= [2, 4]	N = 50 Q = 5 $tsimulation\ MAX$ = 60 seconds [$tarrival\ MIN$,$tarrival\ MAX$] = [2, 40] [$tservice\ MIN$,$tservice\ MAX$]= [1, 7]	N = 1000 Q = 20 $tsimulation\ MAX$ = 200 seconds [$tarrival\ MIN$,$tarrival\ MAX$] = [10, 100] [$tservice\ MIN$,$tservice\ MAX$]= [3, 9]

S-au creat 3 teste pentru datele de intrare descrire in tabelul anterior, iar rezulatele lor au fost afisate in fisierele anexate Test1.txt, Test2.txt, Test3.txt.

6. Concluzii

În concluzie, aplicația de gestionare a cozilor dezvoltată în Java oferă o soluție eficientă pentru gestionarea sarcinilor într-un mediu cu mai multe servere. Aceasta oferă utilizatorilor o interfață intuitivă și ușor de utilizat pentru a monitoriza și gestiona sarcinile din coadă, în timp ce serverele procesează aceste sarcini în mod automat. Aplicația folosește clase precum Task, Server, Scheduler, Strategy și ConcreteStrategy pentru a simula și gestiona diferitele aspecte ale procesării sarcinilor, oferind astfel flexibilitate și scalabilitate în ceea ce privește modul în care se pot adapta nevoilor utilizatorilor. În

ansamblu, aplicația de gestionare a cozilor este o soluție puternică pentru orice mediu de lucru cu mai multe servere care necesită o abordare eficientă și automatizată pentru gestionarea sarcinilor.

7. Bibliografie

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. *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-and-threadpoolexecutor.html*