DOCUMENTATIE

TEMA 2

NUME STUDENT: Zubascu Maria
GRUPA: 30224

CUPRINS

1. Obiectivul temei

Obiectivul principal *al temei este analiza sistemelor bazate pe cozi de asteptare prin simularea unei serii de N clienti care sosesc la un timp dat, intra in cozi, sunt serviti, dupa care parasesc cozile.*

Obiectivele secundare *sunt:*

- *Analiza problemei si identificarea cerintelor*
- *Proiectarea aplicatiei*
- *Implementarea aplicatiei*
- *Testarea aplicatiei*

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

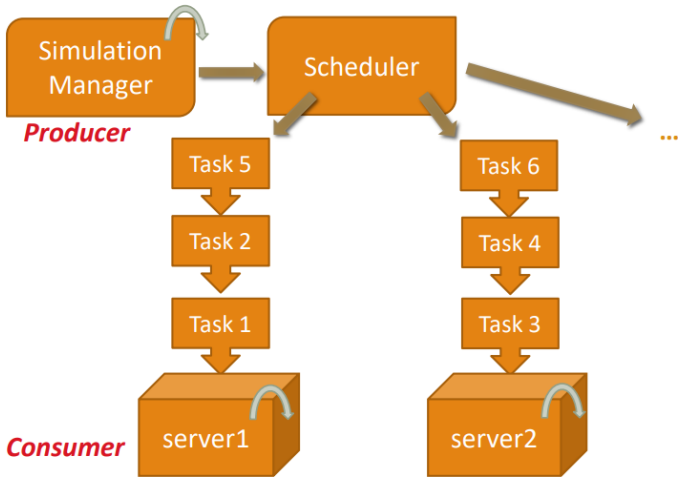
Cerinte functionale:

- *Aplicatia de simulare trebuie sa permita utilizatorului sa configureze simularea*
- *Aplicatia de simulare trebuie sa permita utilizatorului sa initieze o simulare*
- *Aplicatia de simulare ar trebui sa afiseze evolutia cozilor in timp real*

Cerinte non-functionale:

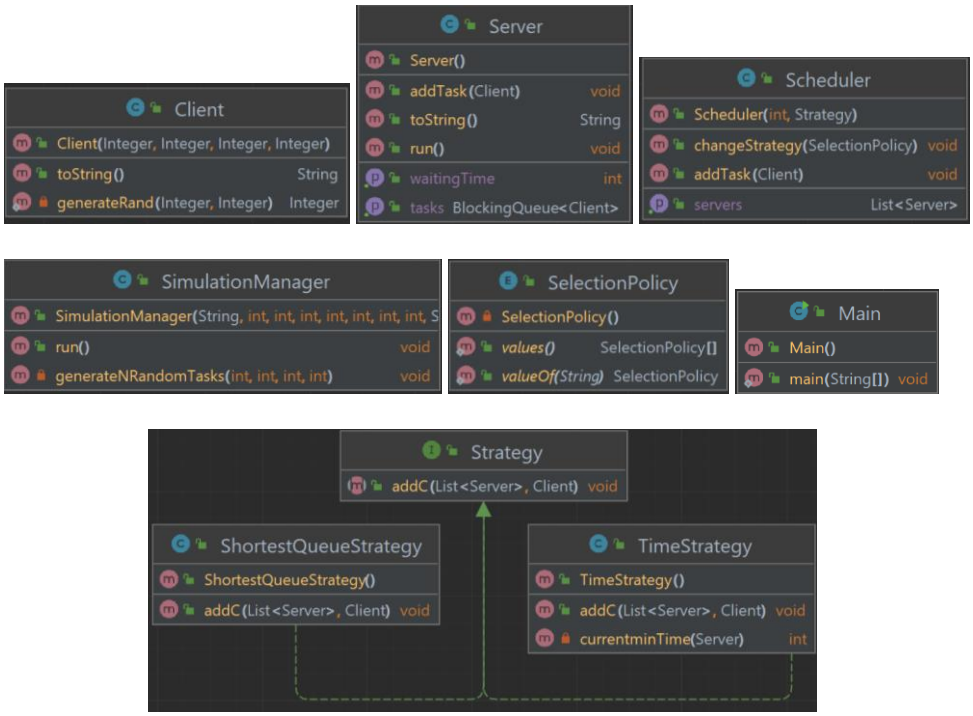
- *Aplicatie de simulare ar trebui sa fie intuitive si usor de utilizat pentru utilizator*

3. Proiectare



Pentru a retine cozile de clienti am folosit structura BlockingQueue<Client> in care am adaugat fiecare client in functie de strategia pe care o alege utilizatorul(TimeStrategy sau ShortestQueueStrategy).Am utilizat Thread-uri pentru fiecare coada, fiecare thread preia clientul, il proceseaza, il scoate din coada, repeta acesti pasi cat timp exista clienti in coada.

Diagrame UML de clase si pachete



4. Implementare

Clasa Client:

- *Se afla in pachetul Model*
- *In interiorul acestei clase se genereaza random timpul de sosire si timpul de servire dintr-un interval dat cu ajutorul metodei ,,generateRand,,*
- *Clasa contine un constructor*
- *Metoda toString este folosita pentru a afisa ID-ul clientului si timpii generati random*

Clasa server:

- *Se afla in pachetul model*
- *Contine o lista de client de tipul BlockingQueue si o variabila de tipul AtomicInteger in care se stocheaza timpul total de procesare*
- *Metoda addTask adauga un client in coada si actualizeaza timpul de procesare*
- *In metoda run este procesat fiecare client de catre thread la fiecare 1000ms*

Clasa Scheduler

- *Aceasta clasa se afla in pachetul BusinessLogic*
- *Contine o lista de servere si o strategie*
- *In interiorul acestei clase sunt create serverele si thred-urile*
- *Clasa contine o metoda changeStrategy care schimba strategia in functie de preferintele utilizatorului*
- *Metoda addTask adauga un nou client in locul stabilit de strategie*

Clasa TimeStrategy

- *In interiorul acestei clase este implementata metoda addC, care primeste un client si o lista de servere. Acesta metoda stabileste serverul la care este adaugat noul client dupa strategia de timp, adica se calculeaza timeService total pe fiecare coada din server, iar clientul este adaugat la server-ul cu timp minim*
- *Aceasta clasa contine un constructor*
- *Se afla in pachetul BusinessLogic*

Clasa ShortestQueueStrategy

- *Acesta clasa se afla in pachetul BusinessLogic*
- *Contine o metoda si un constructor*

- *Metoda addC primeste un client si o lista de servere. Aceasta metoda stabileste si adauga noul client unui server dupa strategia: se calculeaza lungimea fiecarei cozi din lista de servere, se alege serverul cu coada de lungime minima si se adauga noul client*

Clasa SimulationManager

- *Acesta clasa se afla in pachetul BusinessLogic*
- *Contine o lista de clienti generati random, un constructor si doua metode*
- *Metoda generateNRandomTasks genereaza n clienti pe care ii adauga intr-o structura de tipul BlockingQueue<Client>, dupa care sorteaza aceasta lista in functie de timpul de servire*
- *Metoda run calculeaza valoarea medie a timpului de servire, pentru fiecare server afiseaza in timp real clientii, daca un client este adaugat in coada, atunci acesta este scos din lista de asteptare*

Clasa Main

- *Aceasta clasa contine o metoda care creeaza un obiect de tipul SimulationMaganer in care sunt trimise datele de intrare*

5. Rezultate

Testarea simularii a avut loc in metoda main in care au fost rulate 3 seturi de teste, rezultatul acestora fiind stocat in trei fisiere: "test1.txt", "test2.txt", "test3.txt"

6. Concluzii

In concluzie, Tema2 m-a ajutat sa-mi imbunatatesc modul de a aborda o problema, am invatat cum sa organizez codul Java astfel incat sa fie lizibil si simplu, usor de inteles.

De asemenea, in aceasta tema am reusit sa inteleg conceptele de Thread si sincronizare si cum functioneaza acestea.

Posibile dezvoltari ulterioare ale acestui proiect ar fi: realizarea unei interfete grafice prietenoase cu utilizatorul, afisarea unor grafice care sa ilustreze evolutia in timp real, animarea unei simulari cu cozi de client si case de marcat.

7. Bibliografie

1. *What are Java classes? - www.tutorialspoint.com*
2. *https://www.w3schools.com/java/java_threads.asp*
3. *https://www.geeksforgeeks.org/filewriter-class-in-java/*
4. *https://docs.oracle.com/javase/7/docs/api/java/util/concurrent/BlockingQueue.html*
5. *https://linuxhint.com/generate-random-number-within-specified-range-in-java/*
6. *.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.*