MINISTERUL EDUCAȚIEI NAȚIONALE
UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Tema 2

Simulator de cozi

Student: Spinean Sebastian

Grupa: 302210

Cuprins:

1. Obiectivul temei

Obiectivul principal:

Obiectivul principal al acestei teme este proiectarea si implementarea unei aplicatii de simulare ce vizeaza analiza sistemelor bazate pe cozi pentru a determina si minimiza timpul de asteptare a clientilor.

Obiective secundare:

Obiective secundare:	Descriere	Capitol
Determinarea cazurilor de utilizare	Se identifica cazuile de utilizare si se descriu prin intermediul diagramelor use case	2
Alegerea structurilor de date	Se aleg structurile de date potrivite	3
Impartirea pe clase	Impartirea codului in pachete si clase	3
Dezvoltarea algoritmilor	Implementarea algoritmilor	4
Dezvoltarea unei aplicatii multithreading	Aplicatia trebuie sa ruleze pe mai multe fire de executie	3

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

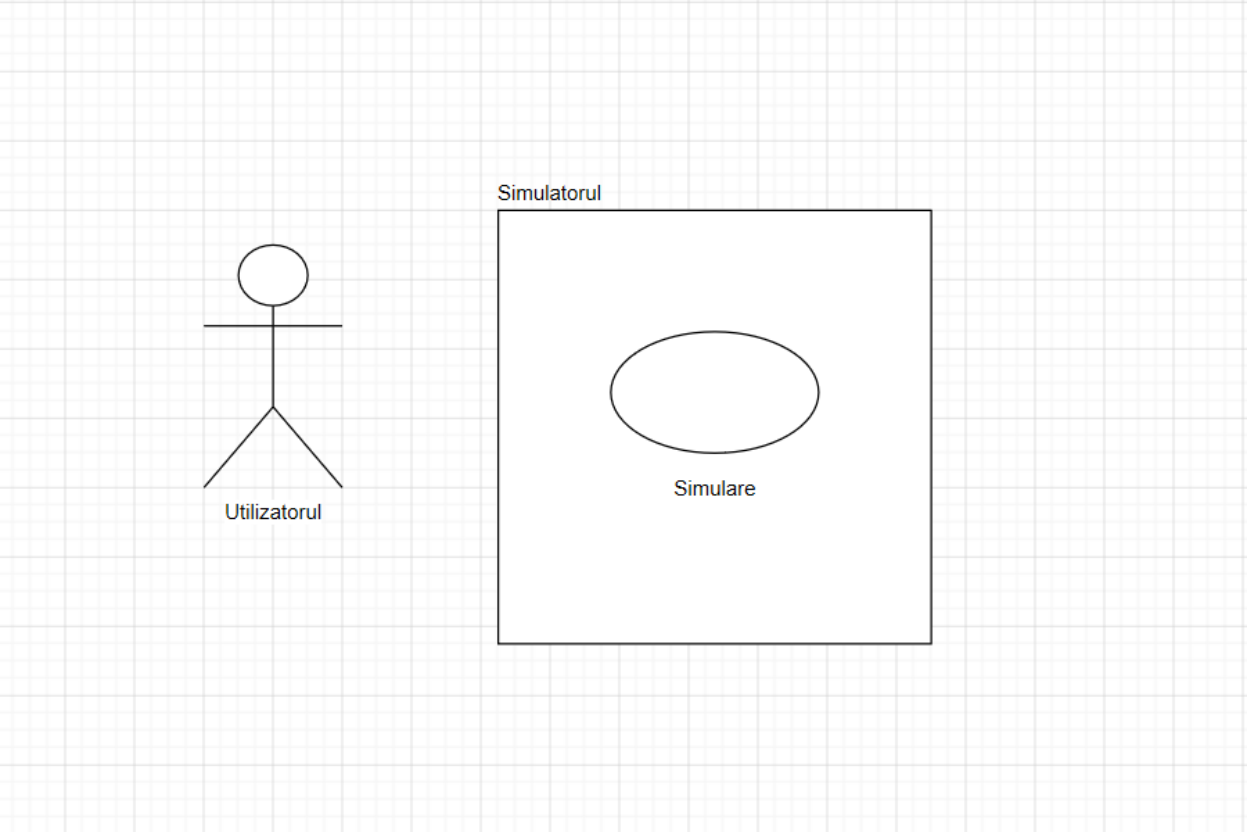
Cerintele functionale:

In cadrul proiectului se disting urmatoarele cerinte functionale:

- Simularea unui numar N de clienti
- Simularea unui numar Q de cozi
- Distribuirea clientilor la cozi in functie de timpul minim de asteptare la o coada
- Calcularea timpului mediu de asteptare la coada

Diagrame use case:

Pentru analiza cazurilor de functionare se folosesc diagramele use case. Astfel, actorul este utilizatorul care foloseste acest proiect. Aceasta aplicatie simuleaza un numar N de client care intra in niste cozi si asteapat sa fie procesati.



Cazul de utilizare:

Actorul principal: Utilizatorul

Scenariul cu success:

1. Utilizatorul introduce datele intr-un fisier text
2. Utilizatorul salveaza fisierul in folderul in care se gaseste aplicatia
3. Se ruleaza aplicatia primind ca argumente numele fisierului ce contine datele de intrare si numele fisierului in care sa fie afisate rezultatele
4. Se afiseaza rezultatele in fisierul specificat, iar daca acest fisier nu exista se creaza automat

Scenariu alternativ:

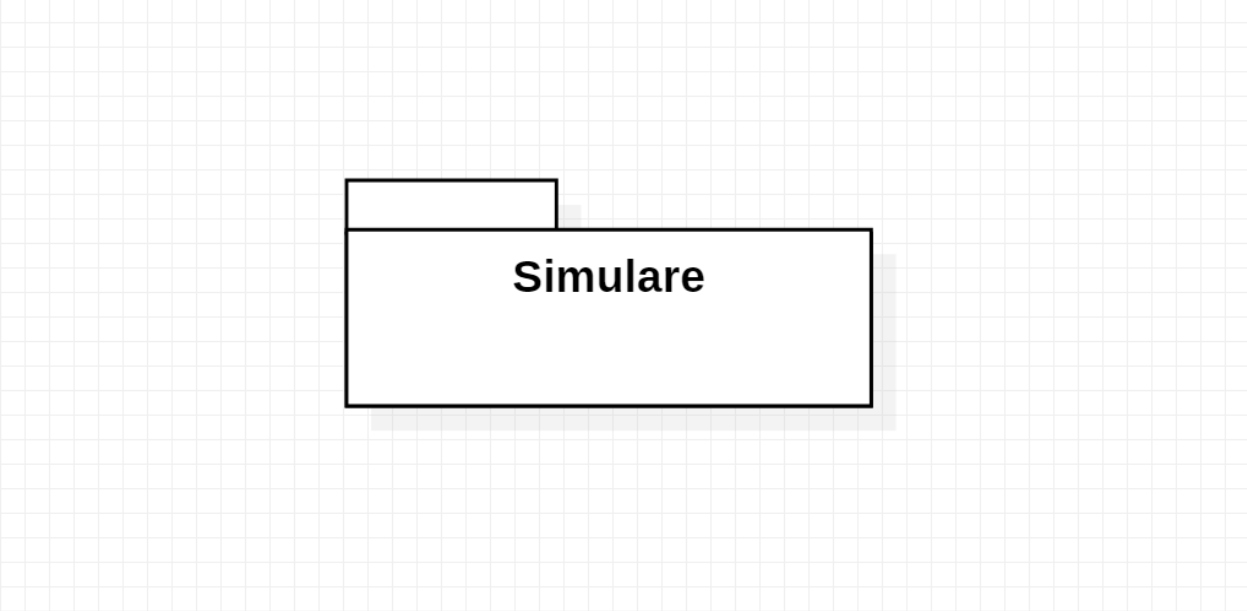
a. Datele de intrare sunt introduse in fisier in alta ordine decat cea corecta
    1. Nu se afiseaza rezultatele dorite sau se arunca o exceptie.
    2. Utilizatorul se reintoarce la pasul 1 si introduce datele in ordinea corecta
b. Fisierul cu datele de intrare nu este salvat in folderol current in care se gaseste si proiectul
    1. Aplicatia nu gaseste fisierul
    2. Se afiseaza un mesaj de eroare.
    3. Utilizatorul revine la pasul 2 si salveaza fisierul in folderul corespunzator

3 Proiectare

Pachet:

Codul aplicatiei este grupat intr-un singur pachet numit Simulare. Acest pachet cuprinde cele patru clase ale proiectului.

Diagrama de pachete:

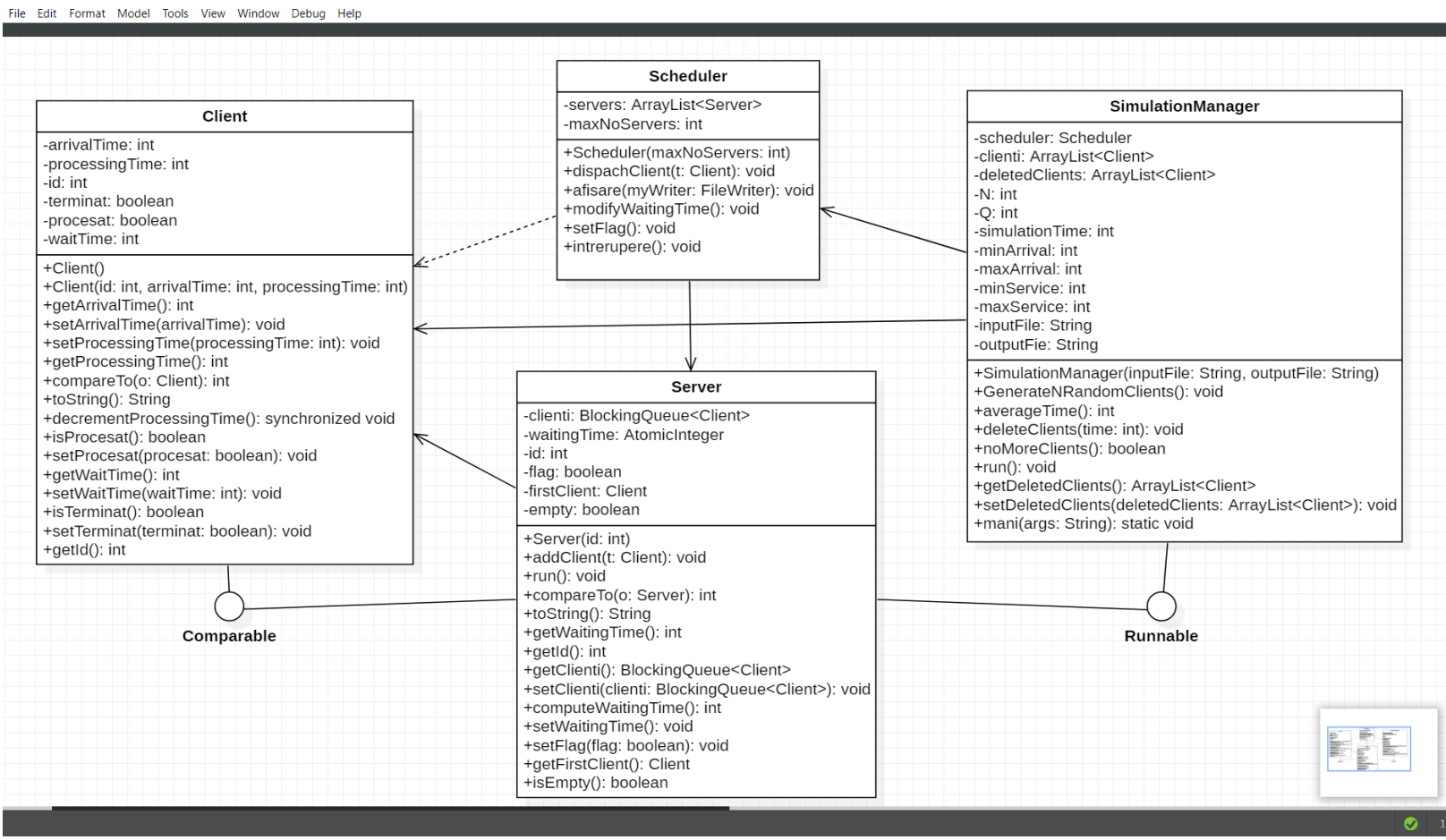


Clase:

Codul este structurat in patru clase si anume:

- Client
- Server
- Scheduler
- SimulationManager

Diagrama de clase:



Multithreading:

Aplicatia trebuie sa fie una multithreading, adica sa ruleze pe mai multe fire de executie. In acest sens, se defineste cate un thread pentru fiecare coada. De asemenea, se mai defineste un thread care retine timpul de simulare si care se ocupa cu distribuirea clientilor in cozi dupa timpul minim de asteptare. Definirea thread-urilor se face prin implementarea interfetei Runnable de catre clasele respective: Server si SimulationManager.

4 Implementare

Clasa Client:

Aceasta clasa modeleaza clientii.

Clasa implementeaza interfata Comparable pentru a putea sorta clientii in functie de tipul de sosire la coada.

Variabile instanta:

Aceasta clasa are sase variabile instanta. Astfel, fiecare client este definit de un id, un timp de sosire la coada notat arrivalTime si de un timp necesar procesarii atunci cand clientul ajunge sa fie in fruntea cozii numit processingTime. Aceste variabile sunt de tip int. Pe langa aceste variabile mai exista inca una de tip int, waitTime, variabila care reprezinta timpul pe care un anumit client l-a petrecut asteptand la coada. Aceasta variabila va fi folosita la finalul simularii pentru a calcula timpul mediu de asteptare la coada. In clasa Client mai exista doua variabile de tip Boolean: variabila procesat – marcheaza momentul in care un anumit client ajunge sa fie in fruntea unei cozi si variabila terminat care marcheaza faptul ca un client a fost terminat de servit.

Constructori:

Clasa dispune de doi constructori. Primul constructor este unul fara parametrii iar cel de-al doilea are trei parametrii de tip int. Cei trei parametrii semnifica id-ul, timpul de sosire la coada si timpul de procesare. In interiorul acestui constructor sunt initializate si celalalte variabile instanta si anume: variabila waitTime este initializata cu 0 iar cele doua variabile de tip Boolean sunt setate ca fiind false.

Metode:

Clasa are metode de getters si setters pentru fiecare variabila instanta. Pe langa aceste metode mai exista si o metoda toString care returneaza un String folosit pentru a afisa un client prin id, timp de sosire la coada sit imp de procesare. Metoda compareTo returneaza un int si este folosita pentru a sorta clientii in ordine crescatoare in functie de timpul de sosire la cozi.

Clasa Server:

Aceasta clasa reprezinta cozile.

Clasa Server implementeaza doua interfete: interfata Runnable si interfata Comparable. Obiectele acestei clase for reprezenta cate un thread asa ca se face implementarea interfetei Runnable. Interfata Comparable este folosita pentru sortarea cozilor in functie de timpul de asteptare.

Variabile instanta:

Aceasta clasa are sase variabile instanta. Prima variabila este de tip int si reprezinta id-ul cozii. O coada detine o lista de clienti, aceasta lista fiind de tip BlockingQueue. O alta variabila importanta este variabila waitingTime care este de tip AtomicInteger. Aceasta variabila reprezinta timpul de asteptare pentru o anumita coada. Variabila de tip Boolean flag este folosita pentru rularea threadului fiind initializata la valoarea true. In variabila firstClient de tip Client se retine primul client din coada si este initializat ca un client special avand id,arrivalTime si processingTime egale cu 0. Variabila booleana empty este true daca in coada nu mai este niciun client si firstClient este clientul cu toate campurile 0. In cazul in care coada este goala dar firstClient este diferit de clientul special atunci variabila este false.

Constructor:

Aceasta metoda are un singur constructor cu un parametru de tip int. Acest parametru reprezinta id-ul cozii respective. In interiorul constructorului se face initializarea variabilei waitingTime la 0.

Metode:

O prima metoda din aceasta clasa este metoda addClient care adauga un nou client primit ca parametru in coada. In aceasta metoda se modifica waitingTime-ul cozii adunandu-se la valoarea acestuia valoarea timpului de procesare a clientului adaugat. De asemenea, se seteaza timpul de asteptare a clientului la coada cu noua valoare a timpului de asteptare a cozii.

Metoda compareTo returneaza un int si este folosita pentru a sorta cozile in ordine crescatoare a timpilor de asteptare. Metoda toString returneaza un String care contine toti clientii din coada.

O metoda foarte importanta din aceasta clasa este metoda run. Aceasta metoda ruleaza cat timp variabila instanta flag este setata la valoarea true. La inceput se seteaza firstClient cu valoarea clientului special si variabila emty la valoarea true. Apoi se extrage primul client din coada iar daca coada nu contine niciun client threadul este pus la somn pana cand in coada o sa apara cel putin un client. Dupa se marcheaza pentru clientul respective faptul ca a inceput sa fie procesat si se retine adresa acestuia in variabila firstClient. La pasul urmator se verifica daca coada este goala caz in care variabila empty devine false deoarece chiar daca nu mai sunt clienti in coada primul element este retinut in variabila firstClient. Se pune threadul la somn pentru o perioada egala cu perioada de procesare a clientului extras. Dupa trezirea threadului se marcheaza clientul ca fiind terminat.

Metoda computeWaitingTime calculeaza suma timpilor de procesare pentru clientii aflati in coada. Metoda setWaitingTime modifica valoarea variabilei waitingTime cu valoarea returnata de metoda computeWaitingTime.

Clasa mai contine metode de getters si setters pentru celalalte variabile instanta.

Clasa Scheduler:

Variabile instanta:

Aceasta clasa are doar doua variabile instanta. Prima variabila este o lista de cozi numita servers, aceasta lista fiind de tipul ArrayList. Cealalta variabila instanta este de timp int, maxNoServers, si reprezinta numarul de cozi.

Constructori:

Clasa Scheduler are un singur constructor care primeste un parametru de tipul int. Cu acest parametru se seteaza numarul de cozi. In cazul constructorului se instantiaza un numar de maxNoServers de cozi, fiecare coada primind un id unic. De asemenea, se pornesc threadurile asociate fiecarei cozi.

Metode:

O prima metoda este metoda de dispachClient care primeste ca argument un obiect de tipul Client. Aceasta metoda se ocupa cu distribuirea clientiilor la cozi. Impartirea clientilor se face in functie de timpul minim de asteptare. In acest fel, in interiorul metodei se face prima data o sortare a listei de cozi, sortare care aranjeaza cozile in ordine crescatoare dupa timpul de asteptare.Dupa aceea, se adauga clientul in coada de pe prima pozitie din lista, coada care are timpul de asteptare cel mai mic la momentul respective.

O alta metoda importanta din aceasta clasa este metoda afisare. Aceasta metoda parcurge toate cozile din lista de cozi si pentru fiecare afiseaza mesajul closed daca nu exista niciun client in ea sau daca nu este goala afiseaza toti clientii din ea.

Metoda modifyWaitingTime modifica timpul de steptare pentru fiecare coada din lista. Metoda setFlag schimba pe false valoarea variabilei instanta flag din fiecare coada. O alta finctie este functia intrerupere care apeleaza metoda setFlag pentru fiecare coada dupa care mai introduce un client special. Astfel, aceasta metoda inchide toate threadurile corespunzatoare cozilor.

Clasa SimulationManager:

Aceasta clasa implementeaza interfata Runnable. Astfel, aceasta clasa va reprezenta un thread care va semnifica durata de simulare.

Variabile instanta:

Aceasta clasa are 12 variabile instanta. O prima variabila este variabila scheduler care este de tip Scheduler. Mai exista doua liste de clienti: clienti reprezinta clientii care asteapta sa intre in cozi si lista deletedClients care cuprinde clientii care au intrat deja in niste cozi. Ambele liste de clienti sunt de tipul ArrayLisy. Urmeaza sapte variabile de tip int care reprezinat : N - numarul de clienti, Q - numarul de cozi, simulationTime – timpul de simulare, minArrival – timpul minim de sosire la coada pentru un client, maxArrival – timpul maxim de sosire la coada pentru un client, minService – timpul minim de procesare al unui client, maxService – timpul maxim de procesare al unui client. Clasa mai contine inca doau variabile de tip String. Aceste variabile reprezinta numele fisierului ce contine datele de intrare si numele fisierului in care vor fi afisate rezultatele.

Constructori:

Clasa SimulationManager are definit un singur constructor cu doi parametrii de tip String. Cei doi parametrii corespund numelor celor doua fisiere, de intrare si de iesire. In interiorul constructorului are loc citirea datelor din fisierul de intrare si atribuirea valorilor acestora variabilelor instanta corespunzatoare. Tot in constructor se mai instantiaza obiectul scheduler si se apeleaza metoda GenerateNRandomClients

Metode:

Metoda GenerateNRandomClients genereaza N clienti. Fiecare client primeste un id unic de la 1 pana la N. De asemenea, pentru fiecare client se genereaza random un timp de sosire la coada cu valoarea cuprinsa intre minArrival si maxArrival. La fel se procedeaza si cu timpul de procesare, valoarea acestuia fiind cuprinsa intre minService si maxService. Dupa ce este generat un client este adaugat in lista clienti. Metoda deleteClients primeste ca argument un parametru de tip int si sterge din lista clienti toti clientii care au timpul de sosire egal cu valoarea parametrului primit. In acelasi timp clientii stersi din aceasta lista sunt adaugati in lista deletedClients.

O alta metoda este metoda averageTime care returneaza un int. Aceasta metoda calculeaza suma timpilor de asteptare la coada a fiecarui client din lista deletedClients.

O alta metoda extrem de importanta este metoda run. Aceasta metoda detine timpul de simulare si face impartirea clientilor la cozi in functie de timpul de asteptare minim. Metoda ruleaza cat timp dureaza simularea sau se opreste daca toti clientii au fost deja procesati. La inceput se parcurge lista clienti si acei clienti care au timpul de sosire egal cu valoarea timpului la care a ajuns simularea sunt distribuiti la cozi. Mai apoi se apeleaza metoda deleteCients si se afiseaza clientii care inca nu au ajuns la cozi. La urmatorul pas se modifica timpul de asteptare pentru fiecare coada

si se decrementeaza timpul de procesare pentru clientii care sunt in acel moment in fruntea cozilor. La fnial se verifica daca mai sunt clienti care asteapta sa fie trimisi la cozi iar daca nu mai sunt se iasa din bucla de while, altfel se incrementeaza timpul simularii si se pune threadul la somn pentru o secunda. Dupa iesirea din bucla while se calculeaza timpul mediu de asteptare la coada si se intrerup threadurile corespunzatoare cozilor. Toate datele sunt afisate intr-un fisier.

De asemenea, clasa mai cintine metode de getters si setters.

Ultima metoda este metoda main. Aceasta metoda primeste ca ergumente numele celor doua fisiere. Se creaza un fisier cu numele fisierului de iesire in caz ca acesta nu exista. Dupa se porneste threadul corespunzator simularii.

5 Rezultate

Pentru testare s-au folosit trei fisiere de intrare numite: in-test-1.txt, in-test-2.txt, in-test-3.txt. Rezultatele lor au fost afisate in fisierele out-test-1.txt, out-test-2.txt, out-test-3.txt.

6 Concluzii

Aceasta aplicatie simuleaza un sistem bazat pe cozi folosit pentru a determina timpul mediu de asteptare si a-l minimiza.

Ca posibilitati de dezvoltare ulterioara s-ar putea adauga mai multe strategii de minimizare a timpului de asteptare precum cresterea numarului de cozi sau introducerea unei strategii bazata pe numarul cel mai mic de clienti ce asteapta la o coada.

7 Bibliografie

http://docs.oracle.com/javase/tutorial/essential/concurrency/index.html