Queues Simulator

Profesor: Student:

Ana-Maria Nanes Cristiana Tintesan

Grupa:30227

Disciplina: Tehici de Programare

2019-2020

# Cuprins

[Cuprins 2](#_Toc37587042)

[1.Analiza problemei 3](#_Toc37587043)

[2.Obiectivul lucrarii 3](#_Toc37587044)

[2.1 Obiectivul principal 3](#_Toc37587045)

[2.2 Obiectivul secundar 4](#_Toc37587046)

[3.Proiectare 5](#_Toc37587047)

[3.1 Decizii de proiectare 5](#_Toc37587048)

[3.4 Implementarea claselor 6](#_Toc37587049)

[3.5 Diagrama de clase 6](#_Toc37587050)

[4.Implementare 7](#_Toc37587051)

[4.1 Clasa Client 7](#_Toc37587052)

[4.2 Clasa Queues 7](#_Toc37587053)

[4.3 Clasa StoreManager 8](#_Toc37587054)

[4.4 Clasa MainClass 10](#_Toc37587055)

[5.Rezultate 11](#_Toc37587056)

[6.Concluzii 12](#_Toc37587057)

[7.Bibliografie 13](#_Toc37587058)

# 1.Analiza problemei

Cerinta acestui assignment: Proiectati si implementati o aplicatie, ce are ca scop analiza sistemelor bazate pe cozi, pentru a determina si a minimiza timpul de asteptare al clientilor.

Cozile sunt, in mod frecvent, utilizate pentru a modela domenii din viata reala. Obiectivul principal al unei cozi este de a determina un loc pentru un client, unde acesta va astepta pana in momentul in care va primi serviciul dorit.Managementul cozii are ca scop scop principal minimizarea timpului de asteptare a unui client.Un mod de a minimiza acest timp de asteptare este a da aduga mai multe cozi, la care clientii pot primi servicile dorite, dar acest lucru are ca si consecinta cresterea costului total.Atunci cand este deschisa o noua coada, clientii pot fi distribuiti in mod cat mai optim la noua coada.Aplicatia ar trebui sa simuleze o serie de clienti care solicita obtinerea servicului dorit, sunt distribuiti in mod optim la o coada, sunt serviti, iar apoi parasesc incinta si sunt eliminati din coada.Pentru a calcula timpul de asteptare trebuie sa stim timpul de sosire, timpul de servire si timpul de plecare.

Pentru implementarea acetui proiect nu este suficient lucru cu medotele obisnuite de pana acuma.Este necesara folosirea conceptului de thread deoarece trebuie sa realizam mai multe lucruri in acelasi timp, in paralel.De exepmlu, nu este suficient sa parcurgem o coada, apoi sa trecem si urmatorea si asa mai departe, deoarece acest lucru ar fi fatal pentru programul nostru.

# 2.Obiectivul lucrarii

## 2.1 Obiectivul principal

Obiectivul principal al lucrarii este de a implementa o aplicatie pentru silumarea timpului de asteptare a clientilor la cozi. Ideal, timpul de asteptate a unui consumator trebuie sa fie minim.In cazul in care, nu este nicio coada goala, clientul va fi plasat la coada cu timpul de asteptare minim.Programul functioneaza dupa un anumit timp de simulare citit dintr-un fisier text de intrare.Un client este identificat prin: id-reprezinta un numar unic care apartine intervalului 1 -numarul total de clienti care ajung la magazin in perioada timpului de simulare; arrivalTime- indica momentul in care consumatorul ajunge in magazin si serviceTime- reprezinta timpul necesar pentru servirea clientului.

Datele de intrare necesare acestui program sunt:

* Numarul total de clienti care trebuie serviti;
* Numarul total de cozi de care dispune aplicatia;
* Timpul de simulare;
* Timpul minim de sosire al clientilor (clientii nu pot ajunge la magazin inainte de aceasta valoare);
* Timpul maxim de sosire al clientilor (clientii nu pot ajunge la magazine dupa aceasta valoare);
* Timpul minim de servire (durata de servire a unui client nu poate fi mai mica decat aceasta valoare);
* Timpul maxim de servire (durata de servire a unui client nu poate fi mai mare decat aceasta valoare);

## 2.2 Obiectivul secundar

|  |  |  |
| --- | --- | --- |
| Obiectiv secundar | Descriere | Capitol |
| Alegerea structurilor de date | Structurile de date folosite pentru a duce la capat obiectivul principal | 3 |
| Impartirea pe clase | Sunt folosite patru clase pentru realizarea obiectivului principal:Clasa MainClass; Clasa Client; Clasa Queues si Clasa StoreManager care sa ocupa de gestinea clientilot in mod cat mai optim | 3 |
| Implementarea solutiei | Vor fi descriese, pentru fiecare clasa in parte, campurile si metodele importante | 4 |
| Testare | Vor fi descries cateva scenarii de testare.In functie de un anumit set de date de intrare, se vor prezenta rezultatele simularii. | 5 |
| Dezvoltarea de use case-uri si scenarii care pot sa apara | Intr-un sistem de software un “use-case” este o lista de actiuni sau pasi de evenimente care definesc in mod obisnuit interactiunile dintre un rol si un sistem in atingerea unui obiectiv. | 3 |

# 3.Proiectare

# 3.1 Decizii de proiectare

Pentru realizarea proiectului, am luat urmatoarele decizii de implementare:

* Am impartit proiectul in patru clase, care vor fi detaliate ulterior;
* Am incercat sa folosesc cat mai multe metode simple, care sa nu contina multe linii de cod cu scopul de a oferi o intelegere rapida si usoara a codului;
* Folosirea unor nume sugestive pentru variabile;
* Pastrarea codului cat mai lizibil si usor de urmarit.

3.2 Structuri de date folosite

* In clasa StoreManager este declarant un ArrayList de clienti pentru a putea pastra datele despre clientii din magazine si pentru a-i putea distribui la o coada.

Private List<Client> cList = new ArrayList<Client>();

* In clasa Queues este declarant un LinkedList de clienti pentru a putea pastra legatura dintre clienti si cozi.

Private LinkedList<Client> clientQueue = new LinkedList<Client>();

3.3 Use case-uri/Scenarii

Utilizarea programului presupune citirea dintr-un fisier de text a datelor de intrare.De pe prima linie a fisierului se citeste un numar natural care reprizinta numarul total de clienti pentru care trebuie sa se realizeze simularea.Pe a doua linie a fisierului se ragaseste numarul total de cozi de care dispune magazinul si care pot fi utilizate pe parcursul simularii, iar pe a treie linie se regaseste numarul care reprezinta durata in care va avea loc rularea proiuectului.Pe ultimele doua linii ale fisierului(liniile patru si cinci) sunt plasate doua numere separate prin caracterul virgula.Primul numar de pe linia patru indica timpul minim de sosire in magazin, iar primul numar de pe linia cinci a fisierului indica timpul minim de servire a unui client.Numerele de dupa virgula reprezinta timpul maxim de sosire, respevtiv durata maxima de servire a unui client.

## 3.4 Implementarea claselor

Proiectul este format din patru clase esentiale:

* Clasa Client – Contine metodele get pentru preluarea informatiilor despre un client: id, arrivalTime si serviceTime si metodele set pentru setarea datelor despre client atunci cand aceste se modifica;
* Clasa Queues – Contine LinkedList-ul de Clienti
* Clasa StoreManager – este cea mai importanta clasa a proiectului deoarece aici se ragasesc medotele esentiale pentru atingerea obiectivului principal: metoda de indentificare a cozii cu timpul minim de asteptare, metoda de generare random a clientilot si thread-ul principal al proiectului.
* Clasa MainClass – contine metoda statica main.In aceasta clasa are loc citirea si prelucrarea datelor din fisierul de intrare.

## 3.5 Diagrama de clase

A screenshot of text

Description automatically generated

# 4.Implementare

## 

## 4.1 Clasa Client

Aceasta clasa contine trei campuri esentiale: idClient, arrivalTime, serviceTime. Fiecare client are un id unic( nu pot exista doi client cu acelasi id), dar arrivalTime si serviceTime pot fi aceleasi pentru mai multi clienti.Clasa client este simpla deoarece contine doar constructorul si metodele getId, getArrivTime si getServTime pentru o putea obtine detaliile despre un client si metodele setIdClient, setArrivalTime, setServiceTime pentru a putea modifica aceste detalii atunci cand este nevoie.

public Client(int idClient, int arrivalTime, int serviceTime) {

setIdClient(idClient);

setArrivalTime(arrivalTime);

this.serviceTime=serviceTime;

}

public int getServTime() {

return serviceTime;

}

public void setServiceTime(int serviceTime) {

this.serviceTime=this.serviceTime-serviceTime;

}

Metoda setServiceTime decrementeaza timpul de asteptare a unui client cu numarul de secunde transmis ca si parametru.Aceasta va fi apelata in clasa StoreManager in threadul principal atunci cand se doreste decrementarea timpului de asteptare a clientului care se afla in fruntea cozii.

## 4.2 Clasa Queues

Clasa Queues contine structura de date prezentata aneterior: LinkedList-ul de clienti.In aceasta clasa sunt prezente cateva medote simple, dar esentiale pentru rezolvarea cerintelor.: getSize()-returneaza un int care indica cati client sunt in coada respectiva; addClient(Client client)- medota folosita pentru adaugarea unui client in una din cozi; removeClient(int i) – sterge clientul din coada care se afla pe pozitia transmisa ca si parametru; getClient() – returneaza primul client din coada; getID(int i); getArriv(int i); getServ(int i) – sunt metode care returneaza un numar ca tipul int care indica id-ul, timpul de sosire, respective timpul de servire al clientului care se afla pe pozitia transmisa ca si parametru in coada.

public void addClient(Client client) {

clientQueue.add(client);

}

public void removeClient(int i) {

clientQueue.remove();

}

public Client getClient() {

Client a=clientQueue.getFirst();

return a;

}

public int getID(int i) {

return clientQueue.get(i).getId();

}

public int getArriv(int i) {

return clientQueue.get(i).getArrivTime();

}

public int getServ(int i) {

return clientQueue.get(i).getServTime();

}

## 4.3 Clasa StoreManager

Clasa StoreManager este clasa de baza a programului nostru. Aici sunt declarate sapte variabile de tipul private int : nrQueues- numarul de cozi disponibile; simTime- timpul de simulare al programului; nrClients- indica numarul total de clienti care necesita servicii pe durata timpului de simulare; minArrivTime- timpul minim de sosire al fiecarui client( un client nu poate ajunge la magazin mai repede decat indica timpul minim de sosire); maxArrivTime- timpul maxim de sosire( dupa aceasta valoare clientii nu mai pot solicita servicii); minServTime- timpul minim de servire al clientului; maxServTime- timpul maxim de servire.Variabila private static int realTime indica timpul real in care ne aflam( secunda de simulare).De asemenea, aici este declarant ArrayListu-ul de clienti: cList si un sir de tipul Queues pentru a stoca toate cozile din program.

Constructorul clasei StoreManager:

public StoreManager(int minArrivTime, int maxArrivTime, int minServTime, int maxServTime, int nrQueues, int simTime,int nrofClients) {

this.minArrivTime = minArrivTime;

this.maxArrivTime = maxArrivTime;

this.minServTime = minServTime;

this.maxServTime = maxServTime;

this.nrQueues = nrQueues;

this.simTime = simTime;

this.nrClients = nrofClients;

qList = new Queues[this.nrQueues];

for (int i = 0; i < nrQueues; i++) {

qList[i] = new Queues(nrQueues);

}

}

Metodele din aceasta clasa sunt: getRealTime()- intoarce o valoare de tipul int care ne indica secunda de simulare in care ne aflam; findMin()- returneaza indexul cozii cu timpul de asteptare minim(coada cea mai optima la care putem aseza un nou client); generateClients()- metoda folosita pentru generarea random a clientilor.Dupa generarea unui client, acesta este adaugat in ArrayList-ul clientilor cList. Generarea clientilor se realizeaza in asa fel incat, se respecta conditiile de validare a unui client: arrivalMinTime<= arrivalTime <= arrivalMaxTime si serviceMinTime<= serviceTime <= serviceMaxTime.

In clasa StoreManager, este regasit thread-ul principal al programului:

A screenshot of a social media post

Description automatically generated

## In prima parte a thread-lui, este apelata metoda generateClients() pentru generearea random a clientilor si “pornirea” cozilor.Rezultatele simularii se regaseste in fisierul de iesire Out-Test1.txt.Daca crearea acestui fiser este esuata programul va arunca o exceptie, iar executia va fi oprita. Prima instuctiune de tip for parcurge timpul de simulare citit din fiser.Toate instructiunile care urmeaza for face parte din aceasta bucla.Parcurgerea prin for each a clientilor are rolul de a asfisa clientii care sunt trecuti pe lista de “Waiting clients” si de a adauga clientii in coada atunci cand sunt indeplinite conditiile.Prin apelarea metodei findMin() se gaseste cea mai potrivita coada la care sa fie plasat un client.

## A screenshot of a cell phone Description automatically generated

In a doua parte a thread-ului sunt parcurse cozile din program si sunt afisati clientii care asteapta la o coada sau mesajul “closed ”, in cazul in care coada este goala.Este apelata metoda setServiceTime(1) cu parametrul 1 pentru a decrementa timpul de servire a clientului care sa afla pe prima pozitie in coada.

## 4.4 Clasa MainClass

MainClass este clasa in care sunt deschise cele trei fisiere de intrare: in-test-1.txt; in-test-2.txt; in-test-3.txt.In functie de ce fisier este transmis ca argument pentru functia Scanner, datele de intrare sunt citite din unul dintre cele 3 fisiere.Fiecare fisier de intrare trebuie sa contine exact 5 linii, iar datele de intrare trebuie sa respecte ordinea specificata.Indiferent din ce fiser se va face citirea datelor, rezultatele vor fi trecue in acelasi fisier de iesire.

# 

# 

# 5.Rezultate

In urma testarii programumlui, s-a constatat ca operatiile de generare random a clientilor sunt corecte.De semenea, se determina coada cu timpul de asteptare cel mai redus pentru adaugarea unui nou client la momentul potrivit.In momentul in care o coada este goala, nu este niciun client, aceasta devine inchisa si urmeaza sa se redeschida doar atunci cand este necesara adaugarea unui nou client.

Exemplu de date de intrare:

* Numarul total de client= 4;
* Numarul total de cozi=2;
* Timpul de simulare=60;
* Timpul minim de sosire=2
* Timpul maxim de sosire=30;
* Timpul minim de servire=2
* Timpul maxim de servire=4;

Rezultatele simularii:

Time: 0

Wainting clients: (1,13,2); (2,3,2); (3,27,3); (4,6,3);

Queue 0: closed

Queue 1: closed

Time: 3

Wainting clients: (1,13,2); (3,27,3); (4,6,3);

Queue 0: (2,3,2)

Queue 1: closed

Time: 4

Wainting clients: (1,13,2); (3,27,3); (4,6,3);

Queue 0: (2,3,1)

Queue 1: closed

Wainting clients: (1,13,2); (3,27,3);

Queue 0: (4,6,3)

Queue 1: closed

Time: 7

Wainting clients: (1,13,2); (3,27,3);

Queue 0: (4,6,2)

Queue 1: closed

Time: 8

Wainting clients: (1,13,2); (3,27,3);

Queue 0: (4,6,1)

Queue 1: closed

Time: 9

Wainting clients: (1,13,2); (3,27,3);

Queue 0: closed

Queue 1: closed

Wainting clients: (3,27,3);

Queue 0: (1,13,2)

Queue 1: closed

Time: 14

Wainting clients: (3,27,3);

Queue 0: (1,13,1)

Queue 1: closed

Wainting clients:

Queue 0: (3,27,3)

Queue 1: closed

Time: 28

Wainting clients:

Queue 0: (3,27,2)

Queue 1: closed

Time: 29

Wainting clients:

Queue 0: (3,27,1)

Queue 1: closed

# 6.Concluzii

In urma realizarii acestui assignment, am reusit sa ma familiarizez cu conceptul de thread.

Proiectul realizat este destul de restrans, dar cu toate aceste, prezinta unele posibilati de dezoltare: indeplinirea corecta si efcienta a tuturor cerintelor; oprirea simularii atunci nu mai sunt lienti trecuti in lista de asteptare,determinarea timpului mediu de asteptare.

Proiectul nu a fost unul usor, datorita lucrului thread-uri si a necesitat multe ore de munca, dar in cele din urma am reusit sa il finalizez cu succes.

# 7.Bibliografie

<https://www.tutorialspoint.com/java/java_multithreading.htm>

<https://www.geeksforgeeks.org/multithreading-in-java/>

<https://www.geeksforgeeks.org/jar-files-java/>