**DOCUMENTATIE**

**TEMA 2**

LASZLO BOGDAN GHEORGHE

GRUPA: 30222

Cuprins

[1. Obiectivul temei 2](#_Toc133193520)

[2. Analiza problemei, modelare, scenarii, cazuri de utilizare 3](#_Toc133193521)

[3. Proiectare 3](#_Toc133193522)

[4. Implementare 5](#_Toc133193523)

[5. Rezultate 5](#_Toc133193524)

[6. Concluzii 5](#_Toc133193525)

[7. Bibliografie 5](#_Toc133193526)

# Obiectivul temei

Obiectivul principal al temei este de a simula un magazin.

Obiective secundare:

* Folosirea unui thread pentru fiecare coada pentru o casa de marcat
* Distribuirea clientilor pe cozi
* Flosirea metodelor safe pentru threaduri

# Analiza problemei, modelare, scenarii, cazuri de utilizare

Va trebui sa implementam un simulator pentru magazine. Coada de la fiecare casa de marcat va reprezenta un BlockingQueue.

Fiecare client are un id, un timp de sosire si un timp de procesare a comenzii. Ei se vor adauga in queue la timpul stability de sosire dupa un anumit criteriu(shortest time).

In momentul in care ei se afla in varful cozii timpul lor de procesare va incepe sa se decrementeze.

Pana la momentul momentul sosirii acestia vor astepta intr-un ArrayList.

In functia main se apeleaza SimulationManager in care se introduce datele necesare pentru inceperea simularii. Starea magazinului se scrie la fiecare secunda intr-un LogFile.

# Proiectare

Proiectul este impartit in clasele:

* Client
* Server
* Strategy
* SimulationManager
* Scheduler

Graphical user interface

Description automatically generated

In clasa Server am implementat un ArrayBlockingQueue pentru clientii de care se ocupa.

In clasa Scheduler am implementat un ArrayList de serveri care memoreaza serverii creati.

In clasa Strategy am implementat metoda prin care se aleg casele(servers).

In clasa SimulationManager am implementat un ArrayList de clienti genereati aleator in functie de timpul maxim si minim de sosire si timpul maxim si minim de procesare.

# Implementare

In clasa Strategy am implementat o metoda prin care se adauga client la o casa in functie de timpul de asteptare sa ajunga in varful cozii.

In clasa Client am implementat metodele decServiceTime care decrementeaza timpul de procesare, getServiceTime care returneaza timpul de procesare, getId care returneaza id-ul clientului si getArrivalTime care returneaza timpul de sosire.

In clasa Server am implementat metodele getId care reurneaza id-ul serverului, setRunFalse care opreste simularea pentru queue, getWaitingPeriod care returneaza timpul de asteptare pentru a termina toate comenzile din queue,setCurrClient care pune clientul de la inceputul queue-ului ca si clientul la care se lucreaza, getCurrClient care returneaza clientul. Current, addClient care adauga un client in queue, run care rescrie metoda run care se gaseste in clasa thread si getClients care returneaza queue-ul de client.

In clasa Scheduler am implementat metodele getServers care returneaza lista de server si dispatchClient care scoate un client din coada atunci csand se t=timpul de procesare devine 0.

In clasa SimulationManager am implementat metodele closeFile care inchide fisierul txt in care se scriu informatiile despre starea magazinului, run care incrementeaza un timer, in fiecare secunda incrementata se modifica simularea, main in care se porneste simularea, toString in care se face metoda de afisarea starii magazinului, generateRandomClients in care se genereaza un numar ales de clienti cu un id, timp de sosire aleatory din intervalul selectat si timpul de procesare din intervalul selctat si writeFile care afiseaza intr-un fisier starea magazinului din fiecare secunda a simularii.

# Rezultate

# Concluzii

Am invatat sa folosesc metodele din clasa Thread intr-un mod thread safe.

# Bibliografie

<https://dsrl.eu/courses/pt/>