

Scutaru Lucian

Gr. 30225

1. Obiectivul temei

Se va prezenta obiectivul principal al temei printr-o fraza si un tabel sau o lista cu obiectivele secundare. Obiectivele secundare reprezinta pasii care trebuie urmati pentru indeplinirea obiectivului principal. Fiecare obiectiv secundar va fi descris si se va indica in care capitol al documentatiei va fi detaliat:

Design and implement a simulation application aiming to analyse queuing based systems for determining and minimizing clients’ waiting time.

The application should simulate (by defining a simulation time 𝑡𝑠𝑖𝑚𝑢𝑙𝑎𝑡𝑖𝑜𝑛) a series of N clients arriving for service, entering Q queues, waiting, being served and finally leaving the queues. All clients are generated when the simulation is started, and are characterized by three parameters: ID (a number between 1 and N), 𝑡𝑎𝑟𝑟𝑖𝑣𝑎𝑙 (simulation time when they are ready to go to the queue; i.e. time when the client finished shopping) and 𝑡𝑠𝑒𝑟𝑣𝑖𝑐𝑒 (time interval or duration needed to serve the client by the cashier; i.e. waiting time when the client is in front of the queue). The application tracks the total time spend by every customer in the queues and computes the average waiting time. Each client is added to the queue with minimum waiting time when its 𝑡𝑎𝑟𝑟𝑖𝑣𝑎𝑙 time is greater than or equal to the simulation time (𝑡𝑎𝑟𝑟𝑖𝑣𝑎𝑙 ≥ 𝑡𝑠𝑖𝑚𝑢𝑙𝑎𝑡𝑖𝑜𝑛).

Obiective secundare:

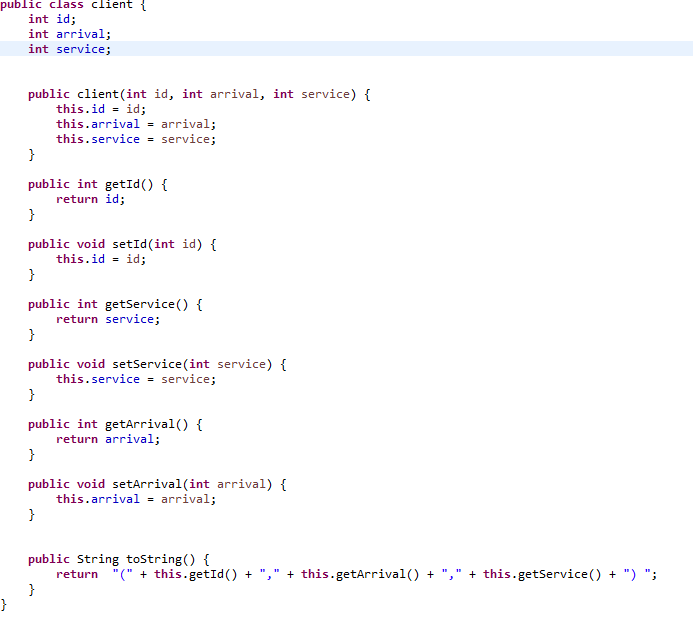
1. Generarea clientilor random - 3
2. Creerea thread-urilor - 3
3. Citirea datelor din fisier - 3
4. Scrierea datelor in fisier - 3
5. Procesarea clientilor in parallel de catre thread-uri - 3
6. Proiectare (decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfete, relatii, packages, algoritmi, interfata utilizator) Se va prezenta proiectarea OOP a aplicatiei, diagramele UML de clase si de pachete, structurile de date folosite, interfetele definite si algoritmii folositi (daca e cazul)

Am decis sa implementez majoritatea metodelor folosite in acest assign-ment in clase diferite. Fiecare clasa este folosita pentru indeplinirea unuia sau a mai multor scopuri. Structurile de date folosite in tema sunt ArrayList-urile care ma ajuta la memorarea listelor de clienti si memorarea clientilor aflati la fiecare queue.

1. Implementare

Se va descrie fiecare clasa cu campurile si cu metodele importante. Se va descrie implemantarea interfetei utilizator.

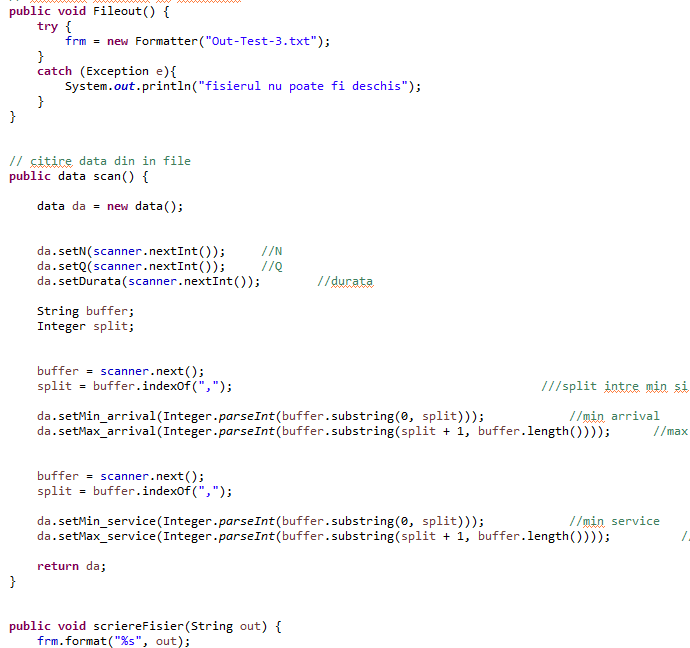
Clasa Client are rolul de a-mi defini clientii ce urmeaza a fi create aleatoriu. Contine 3 variabile de tip int pentru id-ul, timplu in care clientul ajunge si timpul in care clientul este servit. In aceasta clasa am implementat pe langa constructors, setters si getters si o metoda toString() cu ajutorul caruia scriu in output file clientii care se afla la coada.



Clasa DATA o folosesc pentru a retine datele citite din fisierul ce contine informatiile de intrare. Am declarant cate un integer pentru fiecare variabila de care am nevoie(numar client, numar cozi etc) si apoi am scris setters is getters pentru a-I avea la dispozitie in toate clasele ce urmeaza a fi create.

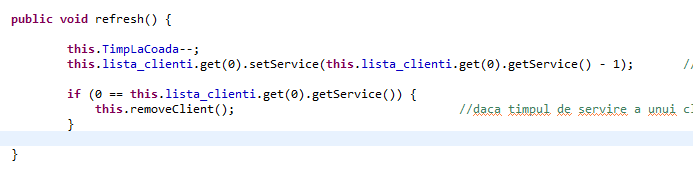


In clasa outFile am definit un scanner(cu care citesc din fisierul de date de intrare) si un Formatter cu ajutorul caruia scriu in fisierul cu rezultate. In aceasta clasa am implementatmetode cu ajutorul carora deschid/inchid fisierele (in si out), o metoda cu ajutorul caruia scriu in fisier si o metoda(cea mai importanta) cu care imi creez un obiect de tip data care contine parametrii cititi din fisierul de intrare.

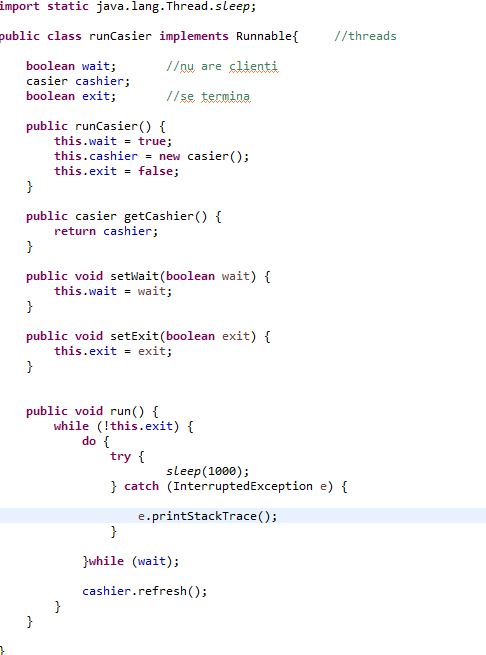


In clasa CASIER am 2 variabile. O ista de clienti cu ajutorul careia tin evidenta clientilor din coada fiecarui casier si un int care reprezinta timpul total pe care un client trebuie sa il astepte la coada pana cand ajunge la casier.

In aceasta clasa am implementat setters si getters si o metoda cu ajutorul careia in momentul in care, daca un client a terminat service-time-ul in fata casierului, atunci acesta este scos din lista si urmatorul client este adus in fata casierului(timpul total al cozii scazand).

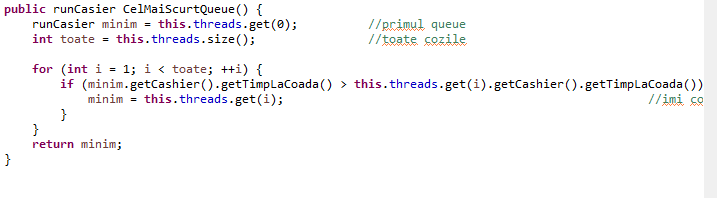


Clasa runCasier reprezinta clasa cu thread-urile care sunt adresate fiecarui server. In aceasta clasa am 3 variabile, un constructor si metoda run(), clasica pentru thread-uri.

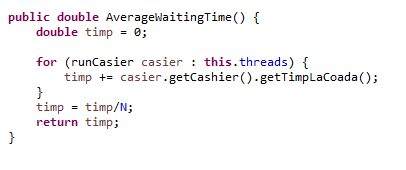


In clasa Simulare am implementat majoritatea metodelor de calcul de care am nevoie pentru a optimiza asezarea la coada a clientilor.

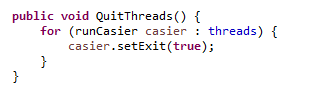
Aici am implementat o metoda care imi calculeaza coada la care se asteapta cel mai putin. In aceasta metoda am luat 2 variabile. Un obiect runCasier(thread) initializat ca si primul queue si un int care reprezinta numarul tuturor cozilor. Caut apoi prin toate queue-urile timpul total de asteptare iar daca acesta este mai mic decat cel al obiectului mentionat mai sus atunci el se updateaza.



Am implementat de asemenea aici si o metoda care imi calculeaza average waiting time



O alta metoda folositoare implementata aici este cea care imi inchide toate thread-urile



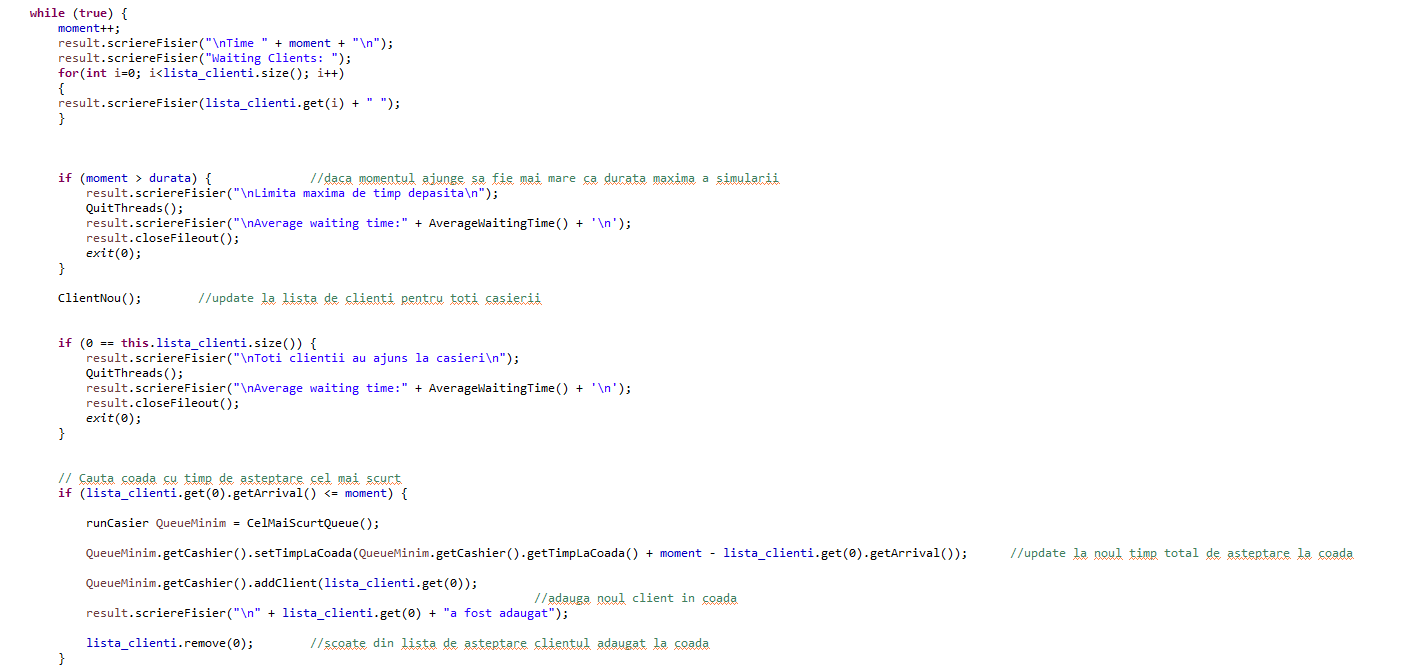
Cea mai importanta metoda implementata in aceasta clasa este metoda run()

In aceasta metoda scriu in fisierul rezultat in functie de cazul la care ma aflu. Cat timp thread-ul care imi tine simularea nu este inchis, la fiecare while timpul se itereaza(trece o secunda)

La fiecare parcurgere a while-ului se scrie in fisier momentul din timp si clientii care sunt in asteptare (nu se afla la cozi) in acel moment. Daca momentul din timp depaseste durata maxima, iar clientii nu au reusi sa ajunga toti la casieri, atunci thread-urile se inchid si apare o eroare(Limita de timp atinsa). Apoi se calculeaza Average waiting time.

La fiecare parcurgere a while-ului daca timpul momentan nu a deposit limita de timp se updateaza clientii pentru fiecare coada iar apoi se verifica daca mai exista client in cozi. Daca nu mai exista atunci apare in fisier un mesaj (Toti clientii au ajuns la casieri) iar thread-urile se inchis. AVG se calculeaza si apare pe ecran.

De asemenea de fiecare data cand se trece prin while,daca exista un client care a ajuns(momen>=arrival Time atunci pentru acesta se calculeaza coada la care trebuie sa astepte cel mai putin.



1. Rezultate

Se vor prezenta scenariile pentru testare cu Junit sau alt framework de testare.

Am testat aplicatia pentru toate cele 3 fisiere de intrare iar acestea functioneaza, insa pentru al 3-lea fisier se pare ca de fiecare data se depaseste limita de timp impusa.

1. Concluzii

In concluzie pot supne ca Assignment 2 functioneaza in proportie de 90%, deoarece nu am reusi sa implementez si situatia fiecarei cozi in fiecare moment insa acestea se inchid/deschid dynamic. Din aceasta tema am invatat sa lucrez cu thread-uri. Dezvoltarile ulterioare constta din adaugarea starii fiecarei cozi la fiecare moment din timp.

1. Bibligrafie

<https://docs.oracle.com/en/>

<https://stackoverflow.com/>

<https://www.geeksforgeeks.org/>