

Proiectarea si implementarea unei aplicatii de simulare bazata pe cozi de prioritati

Halasz Lorand Daniel

Grupa 30226

Cuprins

[1. Obiectivul temei 3](#_Toc511350013)

[2. Analiza si modelarea problemei 4](#_Toc511350014)

[2.1. Cazuri de utilizare si scenarii 4](#_Toc511350015)

[2.2. Asumptii 5](#_Toc511350016)

[3. Proiectare 6](#_Toc511350017)

[3.1. Decizii de proiectare 6](#_Toc511350018)

[3.2. Diagrama UML 6](#_Toc511350019)

[3.3. Structuri de date 7](#_Toc511350020)

[3.4. Proiectare clase 7](#_Toc511350021)

[3.5. Pachete 8](#_Toc511350022)

[3.6. Interfata utilizator 8](#_Toc511350023)

[3.7. Modul de tratare a erorilor 9](#_Toc511350024)

[4. Implementare 10](#_Toc511350025)

[1. Clasa **Client** 10](#_Toc511350026)

[2. Clasa **CasaDeMarcat** 11](#_Toc511350028)

[3. Clasa **Supermarket** 12](#_Toc511350029)

[4. Clasa **MainFrame** 13](#_Toc511350030)

[5. Rezultate 14](#_Toc511350031)

[6. Concluzii 16](#_Toc511350032)

[7. Bibliografie 16](#_Toc511350033)

1. Obiectivul temei

Cozile de prioritati reprezinta un domeniu vast si intens utilizat, inclusiv in lumea reala. Obiectivul principal al temei consta in propunerea, proiectarea si implementarea unui aplicatii de simulare care vizeaza analiza sistemelor bazate pe cozi de prioritati pentru determinarea si minimizarea timpului de asteptare al clientilor. Obiectivul principal poate fi descompus in mai multe obiective secundare, si anume:

* crearea unui jurnal de evenimente
* generarea aleatoare a clientilor
* determinarea anumitor date semnificative simularii, precum timpul mediu de asteptare, timpul mediu de servire, timpul in care o coada este goala sau momentul in care se afla cei mai multi clienti la coada
* implementarea unei interfete grafice
* realizarea unui documentatii

1. Analiza si modelarea problemei

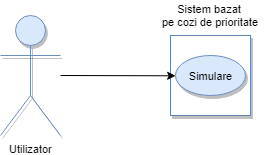
Analiza problemei consta in identificarea domeniului problemei. De cele mai multe ori, problema fiind una complexa urmeaza a fi descompusa in subprobleme pentru a ne usura munca. Aceste subprobleme pot fi, la randul lor, descompuse in alte subprobleme, acest pas repetandu-se pana in momentul in care se ajunge la un nivel de dificultate minim, care poate fi rezolvat. Urmatorul pas consta in identificarea domeniului solutiei. Acest lucru este realizabil prin aplicarea in repetate randuri a unui proces de abstractizare, pana cand se obtine ceea ce se doreste.

Odata analizate aceste lucruri se formeaza o idee generala asupra problemei. Urmatorul pas consta in descrierea cadrului de cerinte functionale, ceea ce se reduce la prezentarea modului in care sistemul reactioneaza in anumite situatii si la anumite intrari. In cazul de fata, pentru sistemul de simulare care urmareste analiza sistemelor bazate pe cozi de prioritati, se cunoaste faptul ca aplicatia trebuie sa realizeze o animatie si sa ofere o interactiune mai usoara pentru utilizator, prin intermediul unei interfete grafice. Tot in cadrul analizei problemei se considera faptul ca o casa de marcat este alcatuita din unul sau mai multi clienti.

Procesul de dezvoltare software se realizeaza pornind de la anumite nevoi. Acestea sunt translatate in cerinte, care urmeaza a fi validate de un client. Pe urma se concepe design-ul prin proiectarea de ansamblu si de detaliu. Se implementeaza componentele si se testeaza separat, dupa care acestea se integreaza in sistem, care la randul sau trebuie validat si testat ca intreg.

* 1. Cazuri de utilizare si scenarii

Cazurile de utilizare reprezinta un set de scenarii legate de modul in care este utilizat sistemul. Acestea ilustreaza modul in care un actor (in cazul de fata un utilizator) interactioneaza cu sistemul. Diagrama de utilizare (diagrama use-case) ne ajuta sa descoperim entitatile sitemului, atributele, actorii si comportamentul sistemului.

Utilizatorul va incepe prin a introduce datele necesare simularii, si anume numarul de case de marcat deschise ( de la 1 pana la 5, unde 1 reprezinta numarul minim de case necesare pentru ca supermarket-ul sa poata sa functioneze, iar 5 reprezinta numarul maxim de case disponibile), capetele intervalului timpului de sosire dintre clienti, intervalul care contine timpul de servire al fiecarui client, si nu in ultimul rand, timpul de simulare, dupa care va apasa pe butonul de start.

Animatia se va afisa in partea din stanga a interfetei, in timp ce in partea dreapta se va afisa jurnalul de evenimente al simularii, iar in partea de jos, date semnificative, precum timpul mediu de asteptare al unei cozi sau timpul mediu de servire al acesteia.

Introducerea datelor in mod corect reprezinta un best scenario. Utilizarea sitemului in mod adecvat presupune introducerea datelor intr-un mod corect din punct de vedere fizic si matematic si respectarea unor cerinte minime. Totusi, acest lucru nu este sigur si ne putem astepta la diferite lucruri, dintre care:

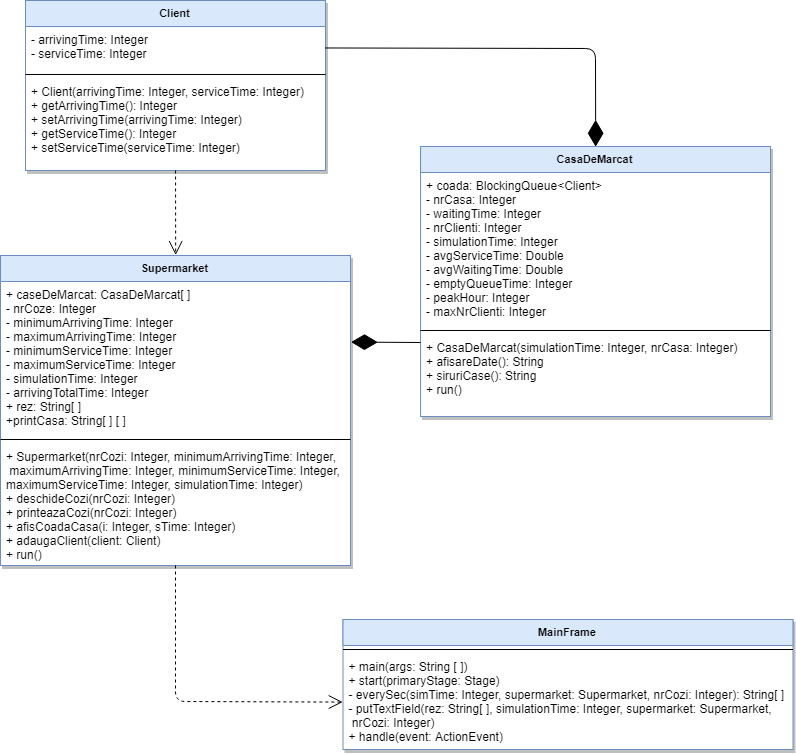
* Introducerea unui numar mai mare de cozi decat cel disponibil
* introducerea unui text in loc de valori
* introducerea unor spatii sau a unor semne de punctuatie
* introducerea unor valori negative
  1. Asumptii

Asumptiile stabilesc cadrul in care se dezvolta aplicatia. In cadrul proiectului s-a stabilit urmatoarea asumptie: presupunem ca datele de intrare sunt introduse in mod corect, conform specificatiilor anterioare.

1. Proiectare
   1. Decizii de proiectare

Pornind de la specificatia proiectului, am luat decizia descompunerii problemei in patru clase: clasa MainFrame ocupandu-se de lansarea aplicatiei si de implementarea interfetei grafice, clasa Supermarket reprezentand un controller, care gestioneaza cozile de prioritati, si clientii, clasa CasaDeMarcat care creeaza coada de prioritati si clasa Client care reprezinta cea mai mica componenta a proietului.

* 1. Diagrama UML

Urmatoarea diagrama UML descrie clasele si obiectele sistemului si relatiile dintre acestea:

* 1. Structuri de date

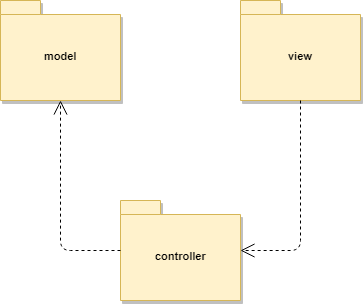
Pentru definirea cozilor de prioritati se utilizeaza o structura de tip coada care respecta principiul FIFO (first in, first out). Acest lucru este realizabil utilizand colectii, implicit clasa LinkedBlockingQueue<T> care implementeaza toate metodele din interfata BlockingQueue<T>.

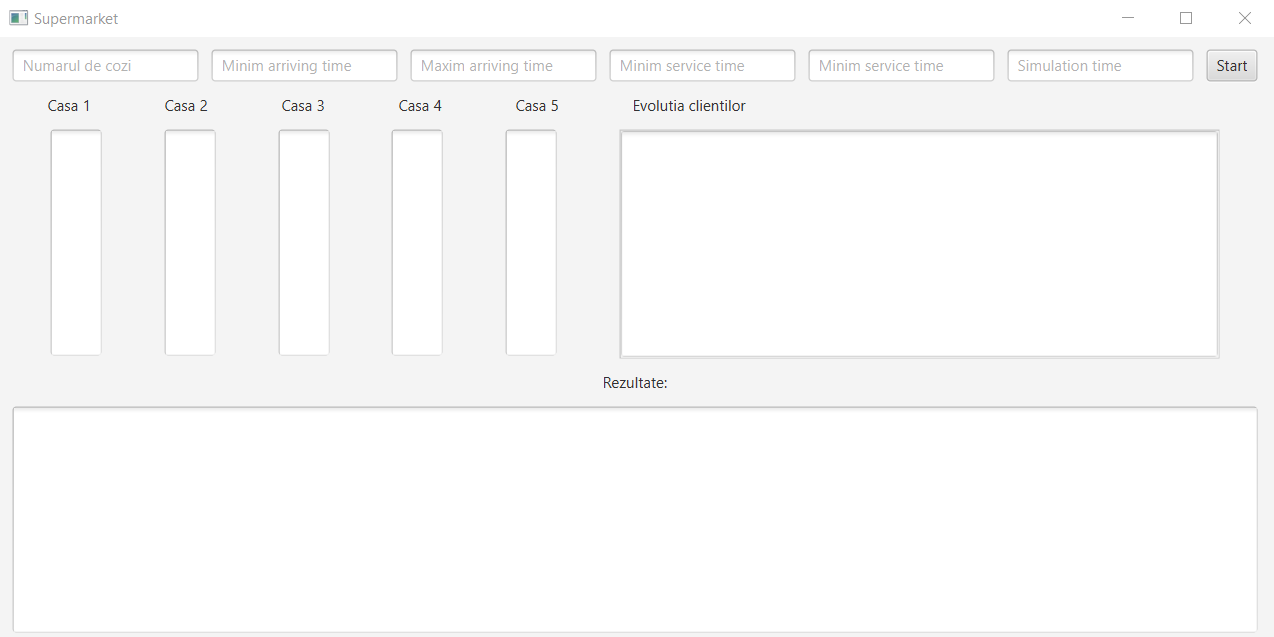
* 1. Proiectare clase

Sistemul contine 4 clase:

1. Clasa **Client** – clasa de baza a proiectului ce defineste structura unui client, care contine timpul la care acesta soseste fata de clientul de dinainte si timpul de servire al acestuia. De asemenea mai contine un constructor care instantiaza campurile.
2. Clasa **CasaDeMarcat** – clasa care defineste structura unei cozi de prioritati, numarul casei de marcat, timpul total de asteptare si alte atribure reprezentative ale acestei clase.
3. Clasa **Supermarket** – clasa care are rolul de controller al sistemului si pune totul in miscare. Aceasta contine un vector de cozi.
4. Clasa **MainFrame** – clasa care sta la baza interfetei grafice si care practic construieste un mod mai usor de interactionare a utilizatorului cu sistemul, acest lucru ducand la „punerea in miscare” a intregului sistem
   1. Pachete

Diagramele de pachet ne ajuta la descompunerea sistemelor mari in subsisteme, lucru necesar pentru a putea realiza o implementare sistematica. Sistemul de simulare, care urmareste analiza cozilor de prioritati, este impartit in 3 pachete:

1. **Pachetul model** – contine clasele Client si CasaDeMarcat
2. **Pachetul** **view** – contine clasa MainFrame
3. **Pachetul** **controller** – contine clasa Supermarket
   1. Interfata utilizator

Interfata grafica reprezinta un mod mai usor de interactionare dintre utilizator si program. Fara aceasta, utilizarea aplicatiei de simulare care vizeaza analiza sistemelor bazate pe cozi de prioritati devine mai greoaie si destul de dificila pentru un utilizator nespecializat. Interfata este realizata cu ajutorul clasei MainFrame care extinde clasa Application.

Interfata este compusa dintr-o fereastra cu titlul „Supermarket” si care este impartita in trei zone, astfel:

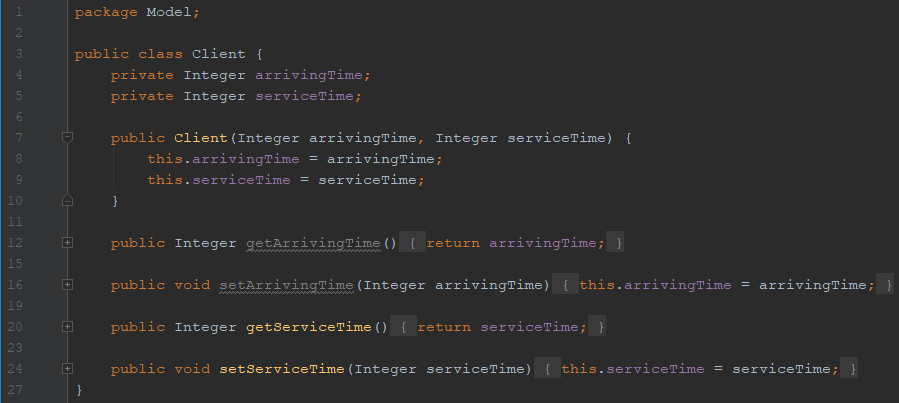
1. Zona 1 – zona compusa din sase textfield-uri, in care utilizatorul va introduce numarul de cozi, intervalul pentru arriving time, intervalul pentru service time si timpul de simulare, si dintr-un buton care porneste simularea aplicatiei
2. Zona 2 – in partea stanga se afla zona in care se poate vedea simularea sistemului, si anume modul in care se umple si se golesc cozile, iar in partea dreapta se afla jurnalul de evenimente, creat pentru a putea observa mai usor modificarile din cadrul cozilor
3. Zona 3 – partea de jos a interfetei, in care se afiseaza, la finalul simularii, date semnificative acesteia, precum timpul mediu de asteptare, timpul mediu de servire, timpul in care o coada este goala sau momentul in care se afla cei mai multi clienti la coada
   1. Modul de tratare a erorilor

Ca in orice sistem, aparitia unor erori este foarte posibila si trebuie luata in calcul. In cazul de fata o eroare destul de posibila consta in introducerea unor date de intrare eronate, mai precis introducerea unui numar de cozi mai mare decat cel disponibil. Acesta eroarea a fost tratata special in cadrul programului prin compararea numarului de case cu numarul maxim acceptat, iar in caz contrar sistemul urmand sa raspunda cu un mesaj de specific care va fi afisat in interfata grafica in zona de rezultate.

1. Implementare

Dupa stabilirea obiectivului principal si a obiectivelor secundare si dupa analiza, modelarea si proiectarea proiectului urmeaza implementarea lui.

1. Clasa **Client**



Clasa Client, are ca si variabile de instanta timpul de sosire si timpul de servire necesar pentru un client in magazin. In cadrul acestei clase este descris atat constructorul clasei, cat si metodele de get si set necesare procesului de incapsulare. Aceasta este o clasa de baza deoarece clientul reprezinta o parte a unei cozi, iar fara el, coada ar fi mereu goala, lucru care ar face inutila o astfel de aplicatie.

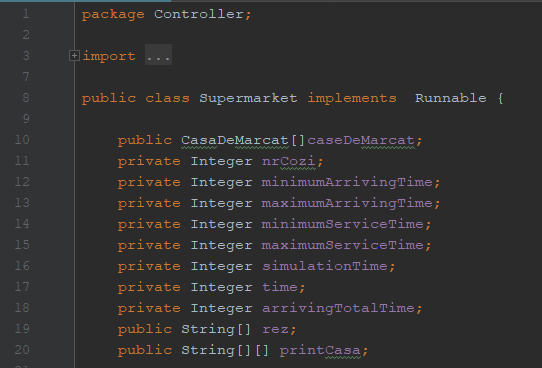
1. Clasa **CasaDeMarcat**

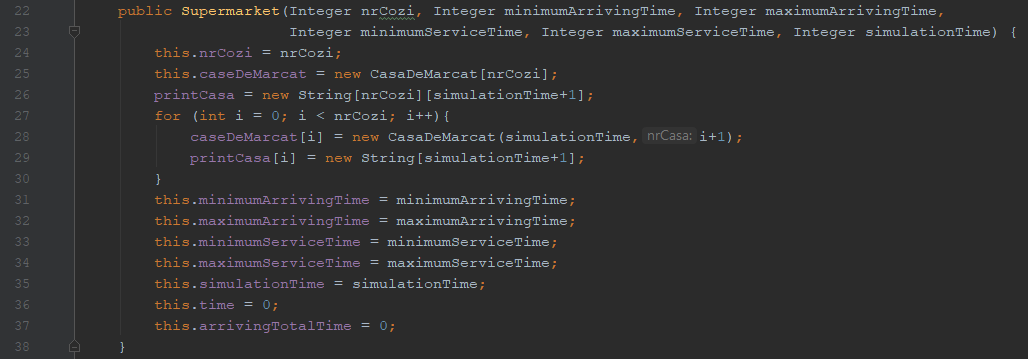


Clasa CasaDeMarcat are ca variabile de instanta o coada de clienti si alte date reprezentative pentru sistem, precum timpul de asteptarea total la o coada, numarul total de clienti care au fost la coada respectiva, timpul total de servire si de asteptare al cozii, timpul in care o coada a stat fara clienti si secunda in care numarul de clienti a fost maxim. Pe langa constructorul in care se initializeaza atributele si metodele get si set necesare, se definesc de asemenea inca trei metode: metoda afisareDate si metoda siruriCase care se ocupa de partea de afisare si simulare, si de asemenea metoda run care este suprascrisa si care are rolul de scoate clientul din coada in cazul in care timpul acestuia ajunge la 0 si de a calcula timpul de asteptare la coada si timpul in care coada a fost lipsita de vreun client.

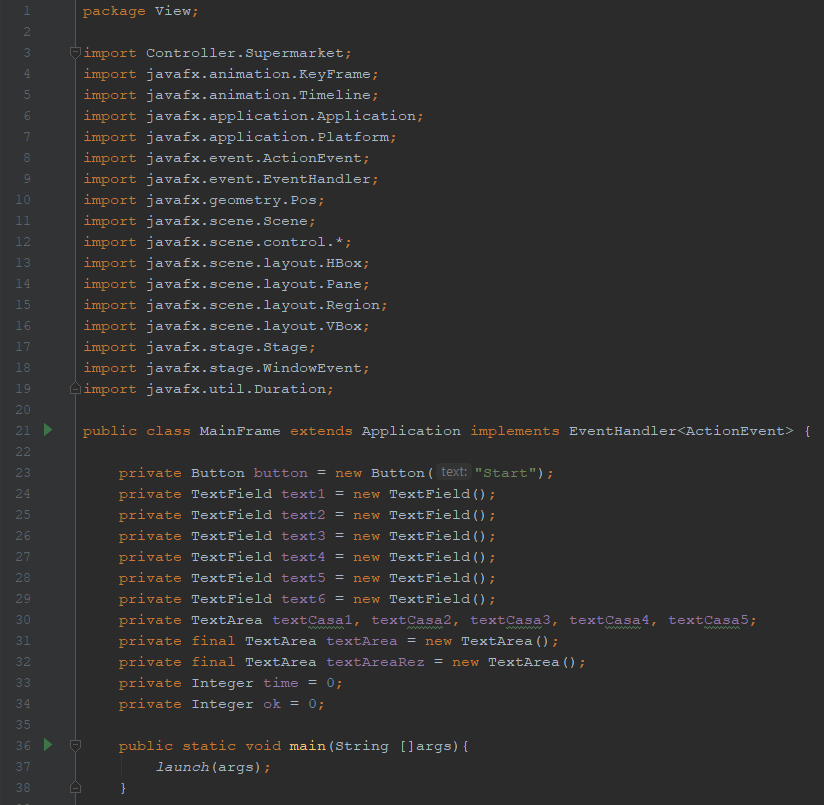
1. Clasa **Supermarket**

Clasa Supermarket indeplineste rolul de controller al acestei aplicatii. In interiorul constructorului se instantiaza si se creeaza un vector de case de marcat si de asemenea se initializeaza restul atributelor definite. In interiorul clasei s-a implementat metoda deschideCozi care creeaza cate un thread pentru fiecare coada, dupa care se produce inceperea lor prin apelarea metodei start. De asemenea mai sunt implementate metode pentru printarea datelor dintr-o coada, pentru afisarea tuturor cozilor, pentru adaugarea unui client in coada si suprascrierea metodei run, in care se genereaza un client nou si se adauga in coada in momentul in care soseste.



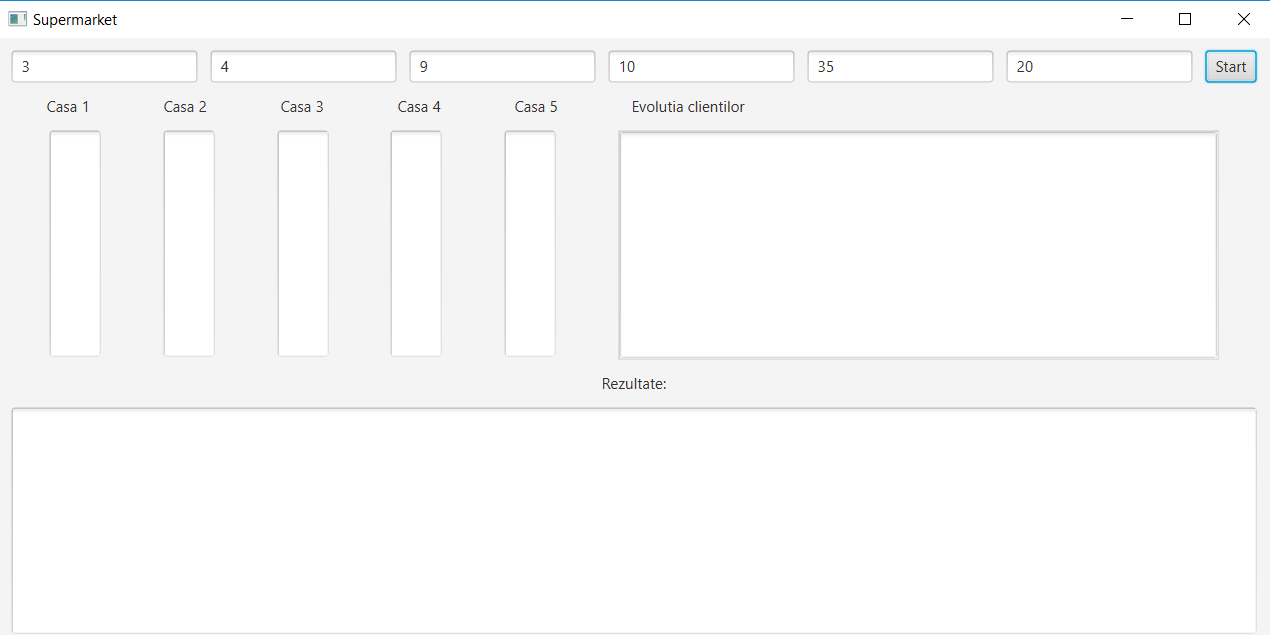


1. Clasa **MainFrame**

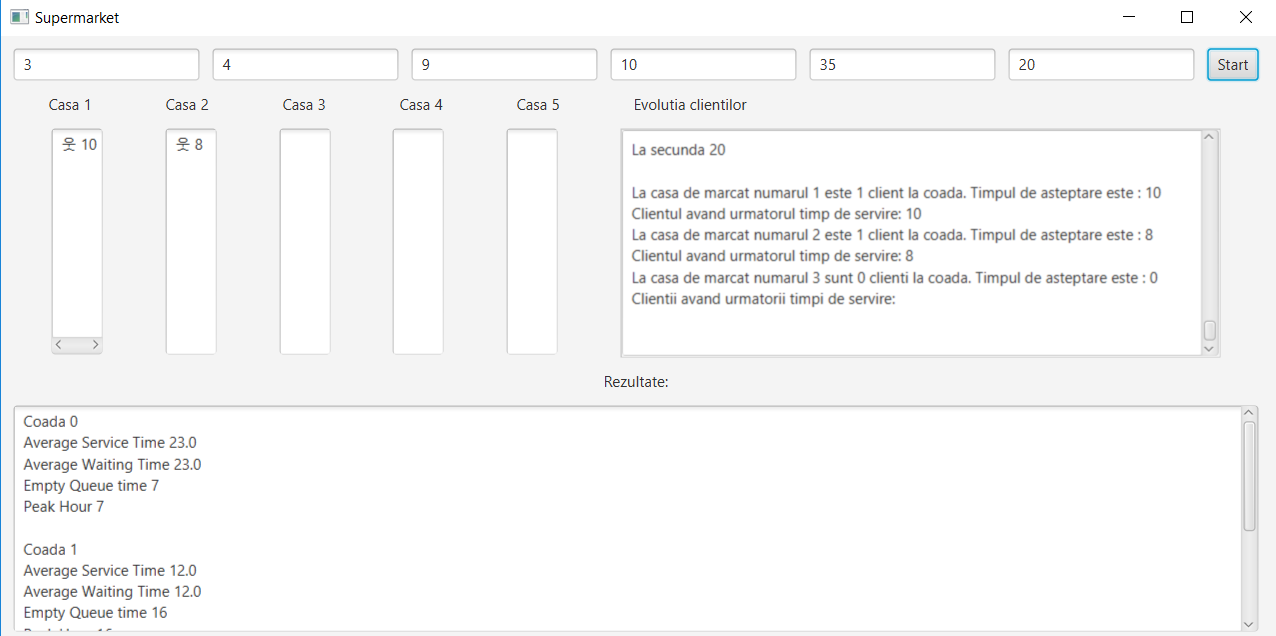
Clasa MainFrame realizeaza practic toata interfata grafica. Aceasta clasa extinde clasa Application si contine metoda main, in interiorul careia se utilizeaza metoda launch. Dupa care urmeaza suprascrierea metodei start care creeaza interfata grafica. Mai apoi, se suprascrie metoda handle care va fi apelata in momentul in care se apasa pe butonul de start, utilizatorul provocand astfel un eveniment. In afara celor trei metode se mai implementeaza alte doua metode care se ocupa de partea de animatie din interfata grafica.

1. Rezultate

Dupa introducerea numarului de cozi, a capetelor celor doua intervale, arriving si service time si a timpului de simulare se apasa pe butonul de start.



In urma simularii s-au obtinut urmatoarele rezultate



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Minim arriving time | Maxim arriving time | Minim service time | Maxim service time | Simulation time |
| 3 | 10 | 7 | 29 | 60 |

|  |  |
| --- | --- |
| Numar Cozi | Rezultate |
| 1 | Coada 1  Average Service Time 15.375  Average Waiting Time 50.75  Empty Queue time 10 seconds  Peak Hour 55 |
| 2 | Coada 1  Average Service Time 21.166666666666668  Average Waiting Time 43.833333333333336  Empty Queue time 6 seconds  Peak Hour 54  Coada 2  Average Service Time 23.5  Average Waiting Time 45.25  Empty Queue time 16 seconds  Peak Hour 40 |
| 3 | Coada 1  Average Service Time 19.25  Average Waiting Time 26.5  Empty Queue time 6 seconds  Peak Hour 38  Coada 2  Average Service Time 23.0  Average Waiting Time 27.0  Empty Queue time 9 seconds  Peak Hour 27  Coada 3  Average Service Time 19.666666666666668  Average Waiting Time 25.0  Empty Queue time 18 seconds  Peak Hour 31 |

1. Concluzii

O concluzie foarte importanta, a acestui proiect, consta in importanta descompunerii problemei in subprobleme si usurinta cu care o problema poate fi rezolvata in acest mod. De asemenea este foarte importanta realizarea unei analize amanuntite si formarea unei idei generale asupra problemei inaintea de a incepe implementarea ei.

Din acest proiect am invatat lucruri precum:

* Imbogatirea cunostintelor in materie de interfata grafica
* Utilizarea colectiilor intr-un cadru mai amanuntit
* Importanta utilizarii proceselor de modularizare, abstractizare si incapsulare
* Utilizarea thread-urilor pentru a realiza o manipulare mai rapida a datelor
* Complexitatea realizarii unui proiect de la 0 care, pe langa partea de rezolvarea a problemei, necesita de asemenea o interfata grafica, o unitate de testare si o documentatie laborioasa a intregului proiect

Sistemul de simulare care vizeaza analiza sistemelor bazate pe cozi de prioritati pentru determinarea si minimizarea timpului de asteptare al clientilor are o gama larga de posibilitati de dezvoltare ulterioara, dintre care amintim:

* Posibilitatea de a putea selecta unitatea de masura a parametrilor introdusi (secunde, minute)
* Utilizarea unui numar mai mare de cozi
* Posibilitatea de a putea inchide si deschide case de marcat in timpul simularii
* Posibilitatea de a avea un intreg lant de magazine (selectarea numarului de magazine)
* Dezvoltarea interfetei grafice

1. Bibliografie
2. <http://www.coned.utcluj.ro/~salomie/PT_Lic/>
3. <http://users.utcluj.ro/~igiosan/teaching_poo.html>
4. <https://www.draw.io/>
5. <http://365programperday.blogspot.ro/2013/07/dynamic-textarea-updation-using-javafx.html>
6. <https://docs.oracle.com/javase/8/javase-clienttechnologies.htm>