

**Facultatea de Automatică și Calculatoare**

**Disciplina: Tehnici de programare**

**Sistem de procesare a polinoamelor**

**Bud Denisa Maria**

**Grupa 30227**

**An II**

Contents

[1. Obiectivul temei 3](#_Toc478918282)

[2. Analiza 3](#_Toc478918283)

[2.1. Analiza problemei si modelare 3](#_Toc478918284)

[3. Proiectare 3](#_Toc478918285)

[3.1 Diagrama UML 3](#_Toc478918286)

[3.2 Structuri de date 5](#_Toc478918287)

[3.3 Clase 5](#_Toc478918288)

[3.4 Algoritmi 6](#_Toc478918289)

[3.5 GUI 6](#_Toc478918290)

[4. Rezultate 8](#_Toc478918291)

[5. Concluzii 8](#_Toc478918292)

[6. Bilbiografie 9](#_Toc478918293)

# Obiectivul temei

Enuntul temei de laborator este “Proiectati și implementati o simulare cu scopul analizarii sistemelor de așteptare pentru determinarea și minimizarea timpului de așteptare a clienților”.

# Analiza

## 2.1. Analiza problemei si modelare

Analizarea problemei presupune depistarea eventualele însușiri și comportamente ale proceselor necunoscute, ceea ce este facilitata de programarea orientata pe obiect, dezvoltata pe ideea ca obiectele pe care dorim sa le manipulam sunt mai importante decat logica necesara pentru manipularea lor.

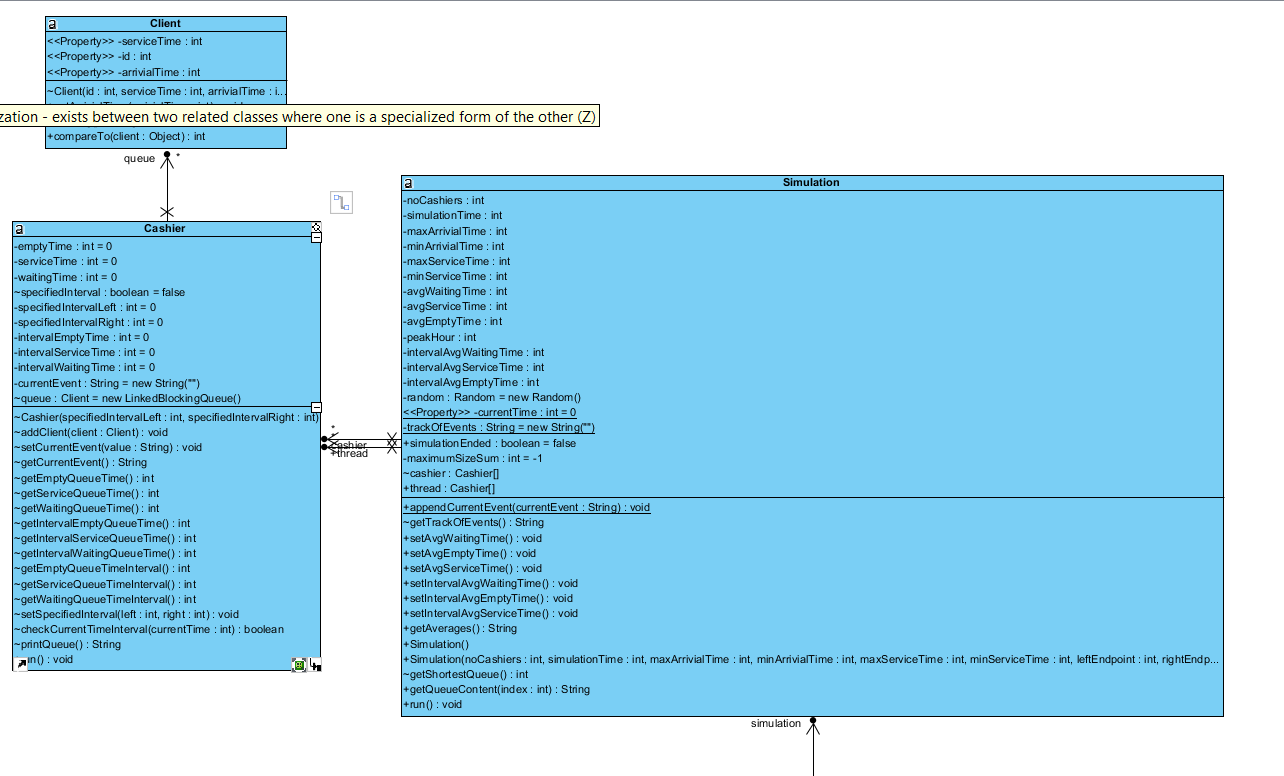
Problema cozilor este întâlnită atât în viața de zi cu zi ( statul la coadă la supermarket) și dorim de fiecare dată să economisim cât mai mult timp, astfel, încercăm să alegem cea mai rapidă coadă. În schimb, chiar dacă numărul de persoane la coadă este relativ mic, nu înseamnă că este și cea mai potrivită pentru a ieși mai repede, deoarece intervine și timpul de servire care depinde strict de personalul magazinului. O metoda de minimizare a timpului de asteptare ar fi prin introducerea unor servere aditionale (cozi).

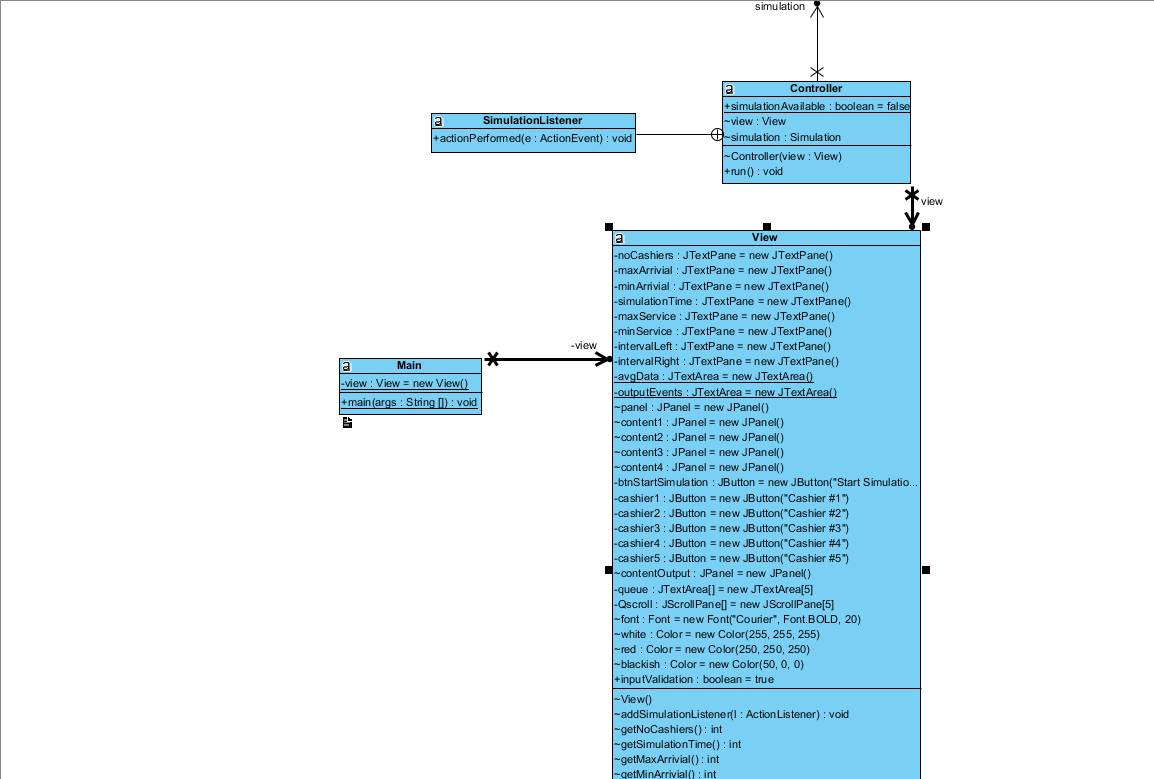
Din textul cerintei se poate depista necesitatea de a folsi cel putin doua clase, pentru structurarea problemei din realitate in entitati. Obiectul client are urmatoarele caracteristici : timpul de sosire la coada, timpul de servire a clientului si ID-ul clientului. Obiectul casier are urmatoarele caracteristici : o lista de clienti care apartin cozii si capetele unui interval de timp specificat din simulare.

# 3. Proiectare

## 3.1 Diagrama UML

Diagrama UML prezintă clasele și metodele prezentate în fiecare clasă.





Proiectarea claselor am realizat-o dupa sablonul Model-View-Controller, datorita izolării logicii programului față de considerentele interfeței cu utilizatorul, rezultând o aplicație unde aspectul vizual și nivelele inferioare ale regulilor sunt mai ușor de modificat, fără a afecta alte nivele. Rolul Modelului este jucat de Simulare.

## 3.2 Structuri de date

In programarea orientata pe obiect, legatura stransa dintre date si operatiile efectuate asupra lor se extinde si asupra structurilor de date. Fiecare obiect este o structura de date, asociata cu o colectie de metode prin intermediul carora se manipuleaza aceste date. Obiectele care au aceeasi structura si aceleasi metode se grupeaza intr-o clasa.  Programul este un ansamblu de clase si obiecte care comunica intre ele prin mesaje.

Pe lângă tipurile de date primitive existente în clasele specificate mai sus, se folosesc cu preponderență și colecții. Una din cele mai importante colecții utilizata in acest proiect se află în clasa Cashier, de tip LinkedBlockingQueue ce conține o listă de obiecte de tipul Client.

Am ales sa folosesc colectia LinkedBlockingQueue deoarece acest tip de implementare este proiectata special pentru cozile producator-consumator, oferind siguranta in cadrul lucrului cu threadurile. La adaugarea elementelor, daca capacitatea cozii este limitata, atunci consumul memoriei este de asemenea limitat. Daca se incearca scoaterea unui element dintr-o coada vida, atunci threadul respectiv este blocat pana cand un alt thread insereaza un nou element in lista.

## 3.3 Clase

Clasele folosite sunt:

* **Client** – clasa care are atributele ID, service time si arrivial time si furnizeaza aceste date private prin mutatori.
* **Cashie**r – extinde Thread, simuleaza activitatea unei cozi, contine colectia LinkedBlockingQueue cu obiecte de tip Client. In metoda run extrage cate un client daca coada nu este goala, ii proceseaza datele de intrare (timpul de sosire la coada, timpul de servire) si le adauga la variabilele de clasa care retin timpul total de asteptare, servire, coada goala, timpul dintr-un interval de asteptare, servire, coada goala.
* **Simulation** - extinde Thread, primeste datele de intrare legate de numarul de casieri, timpul de simulare, timpul maxim si minim de sosire dintre clienti, de servire, un interval specificat. Contine un array de obiecte de tip Cashiers carora, in cadrul metodei run(), le da start. Se foloseste un generator aleator de clienti pentru a fi introdusi in una dintre cozi, aleasa ca fiind coada optima. Se updateaza rezultatele simularii (calculul mediilor) si totodata se incrementeaza timpul curent pana la limita timpului de simulare.
* **View** - Acestui membru îi corespunde reprezentarea grafică, sau mai bine zis, exprimarea ultimei forme a datelor: interfața grafică ce interacționează cu utilizatorul final. Rolul său este de a evidenția informația obținută până ce ea ajunge la controlator.
* **Controlle**r Cu acest element se controleaza accesul la aplicație, este tot un Thread. În acest fel putem diversifica conținutul nostru de o formă dinamică și statică. In același timp. Constructorul instantiaza o variabila de tip view, iar pe variabila de tip button care da start simularii si apartine opiectului view este adaugat un ascultator creat tot in Controller, fapt posibil prin implementarea interfetei ActionListener de catre clasa creata care doreste sa ii urmeze comportamentul.
* **Main** care instantiaza view si controllerul. Da start obiectului de tip Controller instantiat atunci cand primesti informatie de la acesta ca este disponibil, moment care apare doar atunci cand datele introduse de utilizator in GUI sunt valide si atunci cand s-a apasat pe butonul ce da start simularii.

## 3.4 Algoritmi

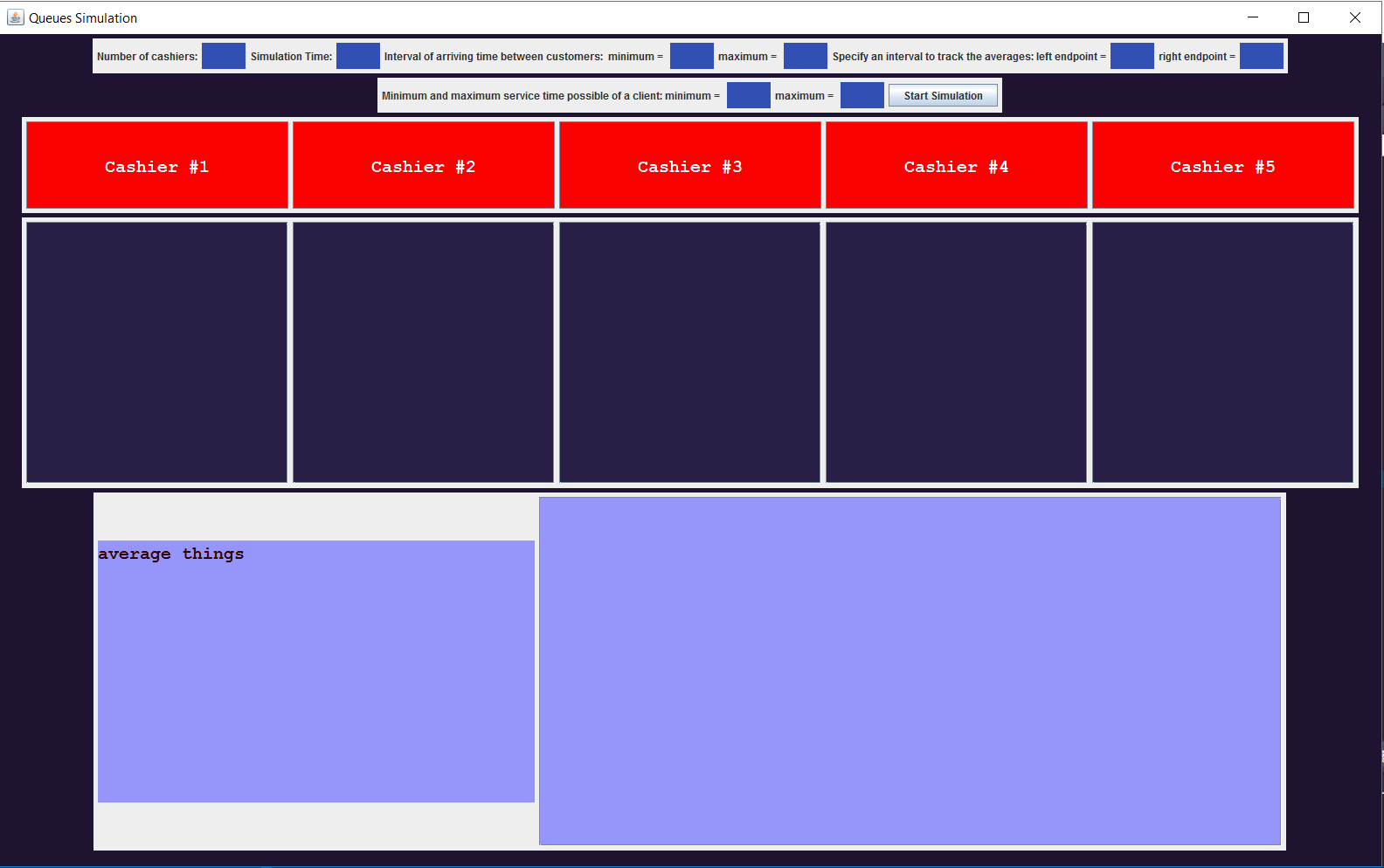
In clasa Cashier in metoda run() au loc o serie ce calcule mai semnficative care contribuie la prezentarea valorilor de iesire curente. Daca coada este vida, se incrementeaza timpul global de coada goala, altfel se extrage primul client din coada, se aduna la timpul de asteptare global timpul de asteptare al acestuia, analog pentru timpul de servire. In cazul intervalului specificat de timp, se verifica daca timpul curent se incadreaza in acest interval, prin apelarea metodei checkCurrentTimeInterval(int currentTime) care trimita valorea true in cazul in care este indeplinita conditia, altfel false. Daca se adevereste incadrarea, atunci adaugam aceleasi date despre client la variabilele globale ce retin valorile temprorare necesare, cu precizia ca sunt pentru intervalul dat. Calculelele care sunt efectuate tin cont ca insumarea valorilor timpilor retinute de clienti sa fie in limita intervalului, astfel, spre exemplu, daca un client are timpul de servire care insumat la timpul curent ar depasi limita superioara a intervalului specificat, atunci s-ar aduna la timpul global de servire din interval doar secundele de servire care apartin acestuia.

## 3.5 GUI

**GUI = Graphical User Interface**, este legătura dintre utilizator și progam. Folosim această interfață deoarece nu orice persoană este obișnuită cu mediul de lucru Eclipse și are cunoștințe de progamare. Astfel, interfața implementează toate metodele prezentate mai sus prin butoane și TextField, pentru a putea fi utilizate de oricine.

Datele sunt introdusă de la tastatură în textField-urile corespunzătoare, care se pot observa în dreptul fiecărui Label. Se asteapta valori intregi pozitive si nenule, iar in campurile ce constituie capetele intervalelor, se asteapta ca acestea sa fie diferite si crescatoare. Simularea incepe prin apasarea butonului Start Simulation, doar daca datele sunt valide, altfel se afisea o casuta de avertizare.

Pentru butonul de start s-a făcut o metodă nouă care primeste ca parametru un ascultator ActionListener și este adaugat la buton.



11

10

9

8

7

6

5

4

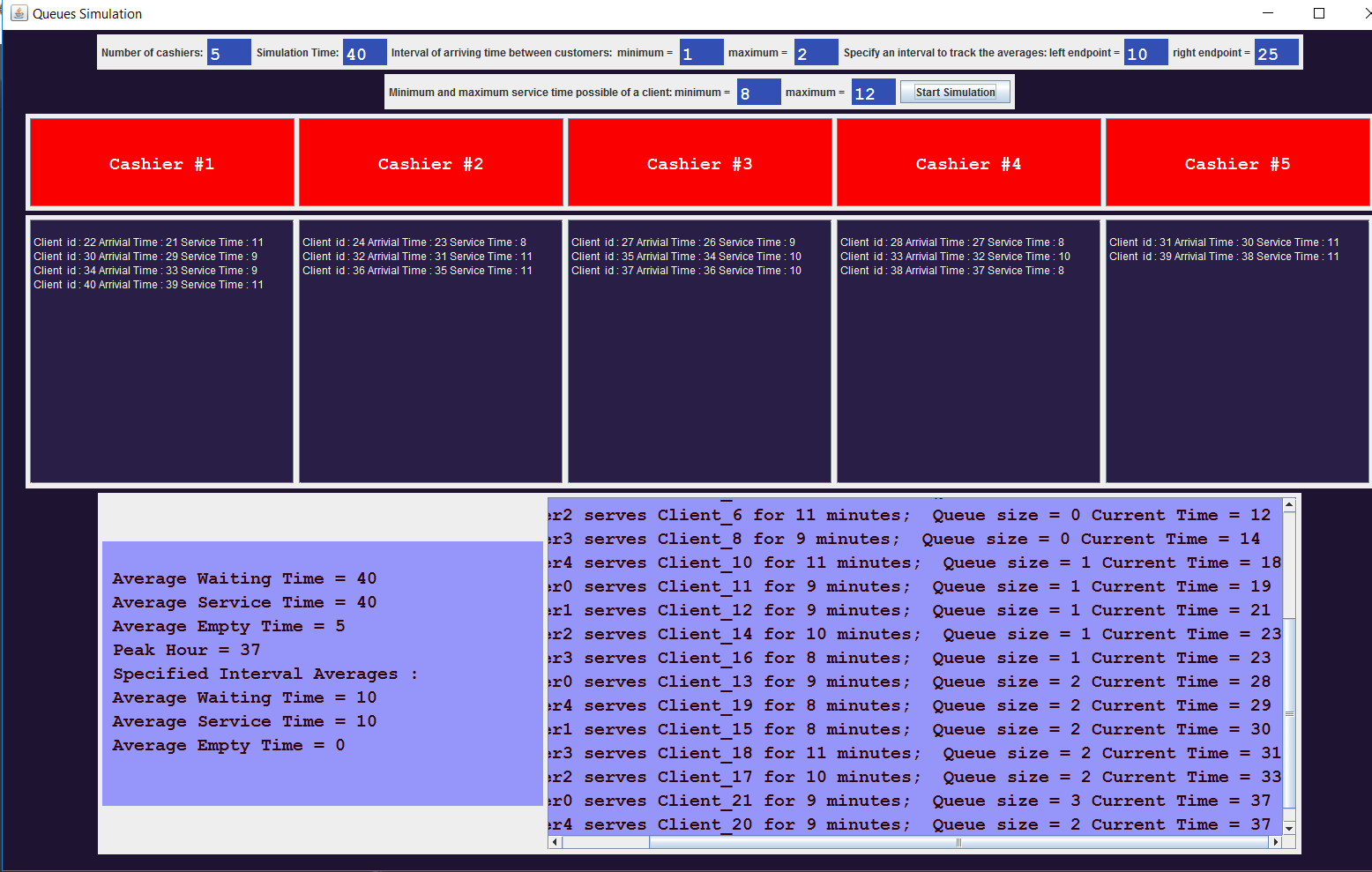
3

1

2

1

1. Camp de introducere a unei valori intregi pozitive reprezentand numarul de casieri -maxim 5 – TextField;
2. Camp de introducere a unei valori intregi pozitive timpul total de simulare- TextField;
3. Camp de introducere a unui numr natural reprezentand timpul minim de sosire dintre clienti - TextField;
4. Camp de introducere a unui numr natural reprezentand timpul maxim de sosire dintre clienti - TextField;
5. Camp de introducere a unui numar natural reprezentand capatul stang al intervalului specificat intre care sa se efectueze calculul mediilor de timp de sosire si asteptare si coada goala- TextField;
6. Capatul drept al aceluiasi interval;
7. Camp de introducere a unui numar natural reprezentand timpul minim de servire a unui client - TextField;
8. Camp de introducere a unui numar natural reprezentand timpul maxim de servire a unui client - TextField;
9. Afisare evolutia fiecarei cozi;
10. Afisare rezultatele medii – se reactualizeaza constant;
11. Afisare Evolutia evenimentelor care au loc in cadrul tuturor cozilor;



# 4. Rezultate

Aplicația de simulare a unor cozi în supermarket poate efectua următoarele operații:

1. Introducerea datelor pentru care dorim să facem simularea – timpii si numarul de casieri, de la 1 pana la maxim 5 case;
2. Începerea simulării prin apăsarea butonului de start;
3. Afisarea evolutiei cozilor pentru fiecare in parte
4. Afisarea evenimetelor globale;
5. Afișarea rezultatelor în timpul simulării in continua actualizare( media timpului de așteptare, servire, ora de vârf, mediile pentru un interval precizat ).

# 5. Concluzii

In concluzie datorita acestui proiect am avut ocazia sa cunosc un tip de structuri de date nou in java , LinkedBlockingQueue. De asemenea am asimilat aspecte esentiale despre lucrul cu threadurile, am aflat despre problematica sincronizarii in aplicatiile multiThread, precum si indispensabilitatea lor.

# 6. Bilbiografie

<http://download.oracle.com/javase/1.4.2/docs/api/javax/swing/JScrollPane.html>

<http://users.utcluj.ro/~igiosan/teaching_poo.html>

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

<http://stackoverflow.com>

https://google.github.io/styleguide/javaguide.html

http://www.mkyong.com/tutorials/junit-tutorials/

https://www.youtube.com/results?search\_query=java+swing