**DOCUMENTATIE TEMA 2**

**SIMULATOR DE COZI**

**Brincoveanu Vasile Vlad**

**30223**

**Profesor Laborator Assist:Antal Marcel**

Contents

[1. Cerinte Functionale 3](#_Toc476131445)

[2. Obiective 3](#_Toc476131446)

[2.1. Obiectiv Principal: 3](#_Toc476131447)

[2.2. Obective Secundare: 3](#_Toc476131448)

[3. Analiza Problemei 3](#_Toc476131449)

[4. Proiectare 3](#_Toc476131450)

[4.1. Structuri de date 3](#_Toc476131451)

[4.2. Diagrama de clase 3](#_Toc476131452)

[4.3. Algoritmi 3](#_Toc476131453)

[5. Implementare 4](#_Toc476131454)

[6. Concluzii si Dezvoltari Ulterioare 4](#_Toc476131456)

[7. Bibliografie 4](#_Toc476131457)

# Cerinte Functionale

* Se vor introduce:timpul minim si maxim de venire intre client, timpul minim si maxim de procesare al fiecarui client, numarul cozilor, timpul simularii, etc
* Se va afisa timpul mediu de asteptare, timpul pana cand coada nu va mai avea clienti, coada desenata visual,se va genera un fisier log cu toate datele despre clienti(timpii etc), se va arata evolutia coziilor .

# Obiective

## Obiectiv Principal:

Dezvoltati si implementati un simulator care va analiza sisteme de cozi pentru a determina si minimiza timpul de asteptare al clientilor

## Obective Secundare:

|  |  |  |
| --- | --- | --- |
| **Obiectiv Secundar** | **Descriere** | **Capitol** |
| Simulator visual din interfata grafica | Aceasta va fi interfata proiectului. | 3 |
| Multithreading: lucru in paralel pe fiecare coada | Fiecare coada va fi un thread separat. | 4 |
| Diferiti timpi: timp de simulare, timp de asteptare mediu,etc | Clientul ,dar si coada, vor tine timpi importanti, pentru statisticile finale. | 4 |

# Analiza Problemei

Cerinta se poate reduce la o simpla si cunoscuta problema din lumea programarii,numita porducator-consumator.In aceasta problema, se vor produce anumite obiecte de catre producator care dupa o anumita logica vor fi consumate de producator. Problema este una destul de generala dar are multe aplicatii practice, precum: statul la coada in magazine, alegerea anumitor cerinte spre a fi executate dintr-un pool mare, etc. Problema se rezolva folosind multithreading, (se va lucre in paralel, folosind fire de lucru, a.k.a Threaduri).

Cerinta noastra ne spune sa simulam un magazin real. Astfel, am generat aleator,pentru a simula cat mai bine realitatea clienti care au anumiti timpi, care vor alege cele mai eficiente cozi unde sa fie procesati.

Vom transforma toate informatiile teoretice de mai sus si vom abstractiza clasele de obiecte necesare.

Acestea vor fi:

* Client
* Coada
* Generator
* Magazin(controller)
* GUI

Programul a fost conceput, folosind concepte din programarea orientate obiect, aplicate prin limbajul JAVA, folosit in mediul de dezvoltare ECLIPSE. Utilizatorul poate fi oricine, el putand sa introduca anumiti timpi de simulare cu scopul de a simula coziile din magazin, vizualizate in interfata.

**Scenarii si user case-uri**

Programul prevede urmatorii actori: utilizatorul, care introduce timpii in milisecunde necesari simularii dorite; seteaza aceste informatii si la apasarea butonului start va putea vizualiza cozile cat si informatiile din ele; controlerul(magazinul) care citeste informatiile scrise de utilizatorul,si le paseaza unui alt actor,modelul(clasa generator) care va genera dupa timpii dati, clienti care vor ajunge la controler; Controlerul dupa o anumita logica va selecta in care coada sa distribuie clientii. Controlerul comunica in permanenta cu interfata ,spunandu-i ce informatii sa modifice.

*Scenariul: Simularea*

1. Utilizatorul introduce in casutele interfetei timpii necesari clientiilor(defapt introduce o raza de valori in care se vor genera anumite valori).Utilizatorul apoi va apasa butonul set.
2. Controlerul va primii aceste informatii si le va distribui generatorului(timpi dintre generarea clientiilor,timpii de procesare al fiecarui client etc) cat si interfetei(numarul de cozi ce trebuiesc desenate, pregatiri pentru a rula programul).
3. Utilizatorul, dupa apasarea butonului set, va putea apasa butonul start pentru a incepe simularea.
4. La apasarea butonului start, interfata comunica cu controlerul, care va pune in executie cozile,generatorul, cat si pe el ,pe cate un fir de lucru separat .Astfel, generatorul incepe sa genereze clienti, controlerul ii primeste si dupa o anumita logica ii va distribui in cozi, care la randul lor ,asteapta sa ii fie serviti clienti, pe care sa ii proceseze, iar apoi sa ii consume. Pe langa toate acestea

controlerul comunica in permanenta cu interfata unde va afisa clinetii din feicare coada, cat si niste informatii scurte despre ei + va scrie si un fisier log extern, cu toate datele despre clienti .

1. Utilizatorul va putea urmarii simularea coziilor, sa vada timpul trecut de la inceperea simularii, sa o opreasca fortat utilizand buntonul restart.

# Proiectare

## Structuri de date

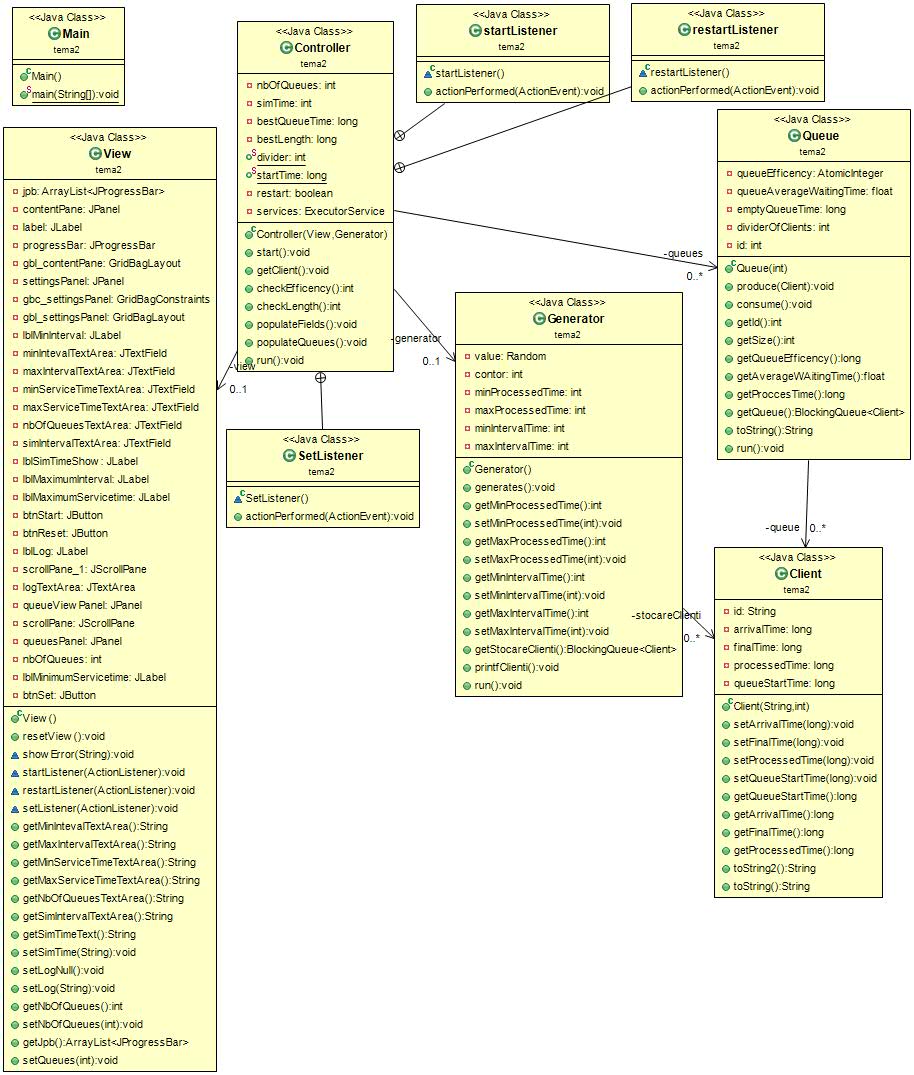
Sunt folosite date simple precum integer, long, pentru memorarea timpiilor, string pentru scierea informatiilor.

Pentru a stoca clientii in cozi am folosit ArrayBlockingQueue , care este o coada, thread safe (introduce un client numai dupa ce va fi loc, scoate un client numai cand exista unul ,etc), in forma de vector (are lungime predestinata = 10).

S-a mai folosit ArrayList(care este un array implementat sub forma de lista, deci fara limita implicita) pentru a stoca JBar-urile din interfata.

S-a folosit si HashMap pentru a verifica cozile in functie de id-ul lor si timpul de procesare, care este cea mai eficienta pentru a putea introduce un client.

## Diagrama de clase



## Algoritmi

Programul contine simplii algoritmi de aflare a cea mai eficiente cozi, de introducere a clientului in coada respectiva , etc

Cel mai evident algortim, cel de aflare a cea mai eficiente cozi, este efectuat de metoda checkEfficency() din clasa controler(a.k.a magazin).

1. Se va instantia un HashMap ce va primii doi Integers.
2. Se va trece cu un for prin toate coziile existente si se vor retine,in HaspMap-ul facut,id-ul si eficienta fiecarei cozi( eficienta se calculeaza in clasa coada, parcurgand fiecare client din ea si adunand processTime-ul fiecaruia).
3. Pentru fiecare intrare din HashMap vom verifica care are cel mai bun timp si vom salva id-ul acelei cozi, pe care il vom returna la final.

Algoritmul de adaugare a clientului intr-o coada anume este strans legat de algoritmul de verificare a eficientei fiecarei cozi.

1. Se va extrage cate un client din generator.
2. Ii se va seta timpul de intrat in magazin(controler).
3. Vom itera prin toate coziile si vom verifica care id este egal cu ce returneaza metoda checkEfficency(), iar acolo vom introduce clientul proaspat generat.

# Implementare

**Clasa Client**

public class **Client**

extends java.lang.Object

Clasa client este core-ul acestui proiect, intrucat ,tot programul se invarte in jurul ei. Ea va tine toti timpii necesari unei simulari bune si unor statistici de dupa simulare, Clientul are metoda toString() suprascrisa pentru a arata informatii suplimentare despre el, si va avea un id unic de recunoastere.

|  |  |
| --- | --- |
| **Fields** | |
| **Modifier and Type** | **Field and Description** |
| private long | [**arrivalTime**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Client.html#arrivalTime)  spune cand clientul intra in magazin(controller) |
| private long | [**finalTime**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Client.html#finalTime)  spune cand clientul iese de la coada |
| private java.lang.String | [**id**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Client.html#id)  numarul unic al clientului |
| private long | [**processedTime**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Client.html#processedTime)  spune cat ii va lua cozii sa proceseze clientul |
| private long | [**queueStartTime**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Client.html#queueStartTime)  spune cand clientul intra in coada |

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [**Client**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Client.html#Client-java.lang.String-int-)(java.lang.String id, int processedTime)  contructorul clasei Client |

|  |  |
| --- | --- |
| **All Methods** | |
| **Modifier and Type** | **Method and Description** |
| long | [**getArrivalTime**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Client.html#getArrivalTime--)()  returneaza timpul de intrare in magazin |
| long | [**getFinalTime**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Client.html#getFinalTime--)()  returneaza timpul de iesire din coada |
| long | [**getProcessedTime**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Client.html#getProcessedTime--)()  returneaza timpul de procesare al clientului |
| long | [**getQueueStartTime**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Client.html#getQueueStartTime--)()  returneaza timpul de intrare in coada |
| void | [**setArrivalTime**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Client.html#setArrivalTime-long-)(long time)  seteaza timpul de intrare in magazine |
| void | [**setFinalTime**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Client.html#setFinalTime-long-)(long time)  seteaza timpul de iesire din coada |
| void | [**setProcessedTime**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Client.html#setProcessedTime-long-)(long time)  seteaza timpul de procesare al clientului |
| void | [**setQueueStartTime**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Client.html#setQueueStartTime-long-)(long time)  seteaza timpul de intrare in coada |
| java.lang.String | [**toString**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Client.html#toString--)()  un rezumat al informatiei purtate de client.(poate fi scrisa in interfata)  returneaza id-ul clientului si o virgula in format String |
| java.lang.String | [**toString2**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Client.html#toString2--)()  returneaza toata informatia despre timpi, pe care o poarta clientul in  format String |

**Clasa Generator**

public class **Generator**

extends java.lang.Object

implements java.lang.Runnable

Clasa generator implementeaza Runnable, astfel va avea o metoda run() care dupa apelarea instantiei cu metoda .start() va incepe sa ruleze , paralel fata de celelalte operatii. Generatorul, precum spune si numele, genereaza clienti,dupa un interval de timp si le seteaza timpul de procesare, iar apoi ii pastreaza intr-un linkedblockingqueue pana cand magazinul(controlerul) extrage cate unul si il trimite in coada.

|  |  |
| --- | --- |
| **Fields** | |
| **Modifier and Type** | **Field and Description** |
| private int | [**contor**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Generator.html#contor)  numarul unic al fiecarui client generat in metoda generate |
| private int | [**maxIntervalTime**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Generator.html#maxIntervalTime)  timpul maxim dintre doi clienti generati |
| private int | [**maxProcessedTime**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Generator.html#maxProcessedTime)  timpul maxim de procesare al clientului |
| private int | [**minIntervalTime**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Generator.html#minIntervalTime)  timpul minim dintre doi clienti generati |
| private int | [**minProcessedTime**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Generator.html#minProcessedTime)  timpul minim de procesare a clientului |
| private java.Util.concurrent.  BlockingQueue<[**Client**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Client.html)> | [**stocareClienti**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Generator.html#stocareClienti)  clientii generati sunt stocati aici,asteptand sa fie dati controlerului |
| private java.util.Random | [**value**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Generator.html#value)  genereaza variatii de timpi pentru o simulare cat mai reala |

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [**Generator**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Generator.html#Generator--)()  contructorul clasei generator |

|  |  |
| --- | --- |
| **All Methods** | |
| **Modifier and Type** | **Method and Description** |
| void | [**generates**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Generator.html#generates--)()  genereaza cate un client, ii seteaza timpul de procesare, ales aleatory din intervalul dat de timpul minim de procesare si cel maxim; il stocheaza in stocareClienti  ca mai apoi sa-l transmita controlerului(magazinului),  iar mai apoi doarme cu o valoare luata aleatory din intervalul dat de timpul minim de interval si timpul maxim |
| int | [**getMaxIntervalTime**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Generator.html#getMaxIntervalTime--)()  returneaza timpul maxim intre generarea  a doi clienti |
| int | [**getMaxProcessedTime**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Generator.html#getMaxProcessedTime--)()  returneaza timpul maxim posibil de procesare al clientului |
| int | [**getMinIntervalTime**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Generator.html#getMinIntervalTime--)()  returneaza timpul minim intre generarea a doi clienti |
| int | [**getMinProcessedTime**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Generator.html#getMinProcessedTime--)()  returneaza timpul minim posibil de procesare al clientului |
| java.util.concurrent  .BlockingQueue<[**Client**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Client.html)> | [**getStocareClienti**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Generator.html#getStocareClienti--)()  returneaza lista cu clientii generate din care, magazinul  ia tot cate unul si il proceseaza |
| void | [**printfClienti**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Generator.html#printfClienti--)()  printeaza toti clientii generati de generator |
| void | [**run**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Generator.html#run--)()  generatorul va produce clienti in paralel fata de celelate task-uri,aceasta metoda intra in functiune cand vom porni threadul pe clasa generator, se va face tot din controler(magazine) |
| void | [**setMaxIntervalTime**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Generator.html#setMaxIntervalTime-int-)(int maxIntervalTime)   seteaza timpul maxim intre generarea  a doi clienti |
| void | [**setMaxProcessedTime**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Generator.html#setMaxProcessedTime-int-)(int maxProcessedTime)  seteaza timpul maxim posibil de procesare al clientului |
| void | [**setMinIntervalTime**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Generator.html#setMinIntervalTime-int-)(int minIntervalTime)  seteaza timpul minim intre generarea a doi clienti |
| void | [**setMinProcessedTime**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Generator.html#setMinProcessedTime-int-)(int minProcessedTime)  seteaza timpul minim de procesare a clientului |

**Clasa Queue(Coada)**

public class **Queue**

extends java.lang.Object

implements java.lang.Runnable

Clasa Queue implementeaza Runnable, astfel va avea o metoda run() care dupa apelarea instantiei cu metoda .start() va incepe sa ruleze , paralel fata de celelalte operatii. Clasa, numita coada, va avea functia de a primii cate un client, de a-l procesa,de a-l consuma si de a-i seta timpii de venire in coada si timpul de plecare, Coada astfel va simula pe cat de real posibil, o coada dintr-un magazine adevarat. Ea isi va calcula eficienta, scazand timpul final (de iesire din coada a clientului) cu timpul de venire, isi va calcula timpul mediu de asteptare, impartind eficienta la numarul de client care ies din coada. Coada are metoda toString() suprascrisa pentru a arata informatii suplimentare despre ea, si va avea un id unic de recunoastere.

|  |  |
| --- | --- |
| **Fields** | |
| **Modifier and Type** | **Field and Description** |
| private int | [**dividerOfClients**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Queue.html#dividerOfClients)  folosit in calculul timpului mediu |
| private long | [**emptyQueueTime**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Queue.html#emptyQueueTime)  timpul in care o coada se va goli |
| private int | [**id**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Queue.html#id)  numarul unic al cozii |
| private java.util.concurrent.BlockingQueue<[**Client**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Client.html)> | [**queue**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Queue.html#queue)  colectia Java folosita pentru a pastra clientii in coada Un BlockingQueue este o coada care suporta, aditional, operatii precum, asteapta ca o coada sa aibe un element apoi extragel,ori asteapta ca sa fie suficient spatiu pentru a insera un nou client in coada. |
| private float | [**queueAverageWaitingTime**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Queue.html#queueAverageWaitingTime)  timpul mediu de asteptare al clientului la coada |
| private java.util.concurrent.atomic.AtomicInteger | [**queueEfficency**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Queue.html#queueEfficency)  cat de buna este o coada in a-si gestiona clientii |

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [**Queue**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Queue.html#Queue-int-)(int id)  contructorul cozii |

|  |  |
| --- | --- |
| **All Methods** | |
| **Modifier and Type** | **Method and Description** |
| void | [**consume**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Queue.html#consume--)()  scoate un client din coada, ii seteaza finalTime, seteaza si queueEfficency si queueAverageWaitingTime,precum si scrie intr-un fisier extern detalii despre fiecare client scos |
| float | [**getAverageWAitingTime**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Queue.html#getAverageWAitingTime--)()  returneaza timpul mediu de asteptare |
| int | [**getId**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Queue.html#getId--)()  returneaza id-ul cozii |
| long | [**getProccesTime**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Queue.html#getProccesTime--)()  obinte timpul de procesare de la fiecare client al cozii |
| java.util.concurrent  .BlockingQueue<[**Client**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Client.html)> | [**getQueue**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Queue.html#getQueue--)()  returneaza coada unde vor fi stocati clientii |
| long | [**getQueueEfficency**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Queue.html#getQueueEfficency--)() |
| int | [**getSize**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Queue.html#getSize--)() |
| void | [**produce**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Queue.html#produce-tema2.Client-)([**Client**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Client.html) client)  insereaza un client in coada si seteaza queueStartTime |
| void | [**run**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Queue.html#run--)()  coada fiind un thread, va rula metoda run in paralel ( incearca sa scoata clienti din coada ) , pentru fiecare instanta a clasei |
| java.lang.String | [**toString**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Queue.html#toString--)()  folosita pentru a afisa in interfata coada cu clientii din ea, clientiilor li se va afisa id-ul , iar cozii ii se va afisa timpul pana cand coada va fi goala si eficienta ei |

**Clasa Controller(Magazinul)**

public class **Queue**

extends java.lang.Object

implements java.lang.Runnable

Clasa Controller implementeaza Runnable, astfel va avea o metoda run() care dupa apelarea instantiei cu metoda .start() va incepe sa ruleze , paralel fata de celelalte operatii. Magazinul va avea functia de a primii cate un client de la generator, de a-I seta timpul de venire in magazin, si de a-l trimite dupa o anumita logica in coada cea mai buna. Astfel, magazinul, primeste prin metoda generate a clasei Generator cate un client care apoi il va trimite in functie de ce va returna metoda checkEfficency in coada cea mai buna. Metoda checkEfficency va stabili care coada, din toate cele functionale, va avea eficienta cea mai bun, acesta va fi calculate astfel vom face media aritmetica dintre eficienta cozii din clasa Queue cu timpul de procesare total al tuturor clientilor din clasa Queue.

|  |  |
| --- | --- |
| **Nested Classes** | |
| **Modifier and Type** | **Class and Description** |
| (package private) class | [**Controller.restartListener**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Controller.restartListener.html)  Inchide toate threadurile apeland metoda shutdown() din clasa ExecutorService , seteaza JBar-urile la 0 , si logul la null |
| (package private) class | [**Controller.SetListener**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Controller.SetListener.html)  metoda set ce va initializa toate variabilele date de interfata, precum: timpi de simulare , timp de procesare al clientilor, intervavul de stat intre genereare clientilor , numarul cozilor etc |
| (package private) class | [**Controller.startListener**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Controller.startListener.html)  Incepe simularea propiu zisa, executa cozile, controlerul ,generatorul din ExecutorService, porneste timpul simularii , etc. |

|  |  |
| --- | --- |
| **Fields** | |
| **Modifier and Type** | **Field and Description** |
| private long | [**bestLength**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Controller.html#bestLength)  lungimea dupa care se alege unde se va insera clientul |
| private long | [**bestQueueTime**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Controller.html#bestQueueTime)  timpu dupa care se alege unde se va insera clientul |
| static int | [**divider**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Controller.html#divider)  folosit la aflarea timpului curent |
| private [**Generator**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Generator.html) | [**generator**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Controller.html#generator)  instanta generator controlata de aceasta clasa |
| private int | [**nbOfQueues**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Controller.html#nbOfQueues)  numarul cozilor |
| private java.util.ArrayList<[**Queue**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Queue.html)> | [**queues**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Controller.html#queues) |
| private boolean | [**restart**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Controller.html#restart) |
| private java.util.concurrent.ExecutorService | [**services**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Controller.html#services)  un obiect care executa si inchide obiectele ce implementeaza runnable(threads, ruleaza paralel) |
| private int | [**simTime**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Controller.html#simTime)  timpul simularii |
| static long | [**startTime**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Controller.html#startTime)  timpul de inceput al simularii |
| private [**View**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html) | [**view**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Controller.html#view)  instanta view controlata de aceasta clasa |

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [**Controller**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Controller.html#Controller-tema2.View-tema2.Generator-)([**View**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html) v, [**Generator**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Generator.html) g)  contructorul Controlerului instanteaza listeneri pentru interfata dar si leaga generatorul si interfata de acesta |

|  |  |
| --- | --- |
| **All Methods** | |
| **Modifier and Type** | **Method and Description** |
| int | [**checkEfficency**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Controller.html#checkEfficency--)()  metoda care verifica eficienta fiecarei cozi in rulare si decide care coada sa fie aleasa, in functie de eficienta, pentru a adauga un client in ea |
| int | [**checkLength**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Controller.html#checkLength--)()  metoda care verifica eficienta fiecarei cozi in rulare si decide care coada sa fie aleasa, in functie de clienti deja aflati in coada, pentru a adauga un client in ea |
| void | [**getClient**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Controller.html#getClient--)()  metoda de baza a clasei, scoate cate un client din generator ii seteaza anumiti timpi apoi decide la ce coada sa-l trimita |
| void | [**populateFields**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Controller.html#populateFields--)()  scoate toate informatiile din interfata introduse de utilizator, apoi instanteaza clientul si controllerul cu acestea |
| void | [**populateQueues**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Controller.html#populateQueues--)()  scoate toate informatiile din interfata introduse de utilizator, apoi instanteaza coziile cu acestea |
| void | [**run**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Controller.html#run--)()  scoate clienti din generator, decide in care coada sa ii introduca, actualizeaza interfata cu informatii privind cozile,verifica cand timpul simularii a ajuns la final si inchide Runnables. |
| void | [**start**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Controller.html#start--)()  metoda ce executa generatorul,cozile precum si controlerul folosind clasa Executor |

**Clasa Main**

public class **Main**

extends java.lang.Object

Clasa Main instanteaza clasele din program

.

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [**Main**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Main.html#Main--)() |

|  |  |
| --- | --- |
| **All Methods** | |
| **Modifier and Type** | **Method and Description** |
| static void | [**main**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\Main.html#main-java.lang.String:A-)(java.lang.String[] args)  instanteaza interfata controlerul si generatorul |

**Clasa View**

public class **Main**

extends java.lang.Object

Clasa View este interfata programului. Este constituita dintr-un panel mare cu un layout gridbox, unde la vest am facut la panel unde am adaugat campurile pentru timpi si logger-ul dedesupt, iar la est am construit din JBar-uri cozile care se updateaza de la controler.

|  |  |
| --- | --- |
| **Fields** | |
| **Modifier and Type** | **Field and Description** |
| private javax.swing.JButton | [**btnReset**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#btnReset) |
| private javax.swing.JButton | [**btnSet**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#btnSet) |
| private javax.swing.JButton | [**btnStart**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#btnStart) |
| private javax.swing.JPanel | [**contentPane**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#contentPane) |
| private java.awt.GridBagConstraints | [**gbc\_settingsPanel**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#gbc_settingsPanel) |
| private java.awt.GridBagLayout | [**gbl\_contentPane**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#gbl_contentPane) |
| private java.awt.GridBagLayout | [**gbl\_settingsPanel**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#gbl_settingsPanel) |
| private java.util.ArrayList<javax.swing.JProgressBar> | [**jpb**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#jpb) |
| private javax.swing.JLabel | [**label**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#label) |
| private javax.swing.JLabel | [**lblLog**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#lblLog) |
| private javax.swing.JLabel | [**lblMaximumInterval**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#lblMaximumInterval) |
| private javax.swing.JLabel | [**lblMaximumServicetime**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#lblMaximumServicetime) |
| private javax.swing.JLabel | [**lblMinimumServicetime**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#lblMinimumServicetime) |
| private javax.swing.JLabel | [**lblMinInterval**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#lblMinInterval) |
| private javax.swing.JLabel | [**lblSimTimeShow**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#lblSimTimeShow) |
| private javax.swing.JTextArea | [**logTextArea**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#logTextArea) |
| private javax.swing.JTextField | [**maxIntervalTextArea**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#maxIntervalTextArea) |
| private javax.swing.JTextField | [**maxServiceTimeTextArea**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#maxServiceTimeTextArea) |
| private javax.swing.JTextField | [**minIntevalTextArea**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#minIntevalTextArea) |
| private javax.swing.JTextField | [**minServiceTimeTextArea**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#minServiceTimeTextArea) |
| private int | [**nbOfQueues**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#nbOfQueues) |
| private javax.swing.JTextField | [**nbOfQueuesTextArea**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#nbOfQueuesTextArea) |
| private javax.swing.JProgressBar | [**progressBar**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#progressBar) |
| private javax.swing.JPanel | [**queuesPanel**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#queuesPanel) |
| private javax.swing.JPanel | [**queueViewPanel**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#queueViewPanel) |
| private javax.swing.JScrollPane | [**scrollPane**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#scrollPane) |
| private javax.swing.JScrollPane | [**scrollPane\_1**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#scrollPane_1) |
| private javax.swing.JPanel | [**settingsPanel**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#settingsPanel) |
| private javax.swing.JTextField | [**simIntervalTextArea**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#simIntervalTextArea) |

a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a aa a a a a a a a a a a a aa a a a a a a a a a a a a aa a a a a aa a a a a a a aa a a a a a a aa a a a a a a aa a a a a a a a a a a a aa a aa a aa aa a a a a a aa aa a a a a aa a a a aa a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a a aa a a a a a a a a a a a aa a a a a a a a a a a a a aa a a a a aa a a a a a a aa a a a a a a aa a a a a a a aa a a a a a a a a a a a aa a aa a aa aa a a a a a aa aa a a a a aa a a a aa a

|  |
| --- |
| **Constructors** |
| **Constructor and Description** |
| [**View**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#View--)() |

|  |  |
| --- | --- |
| java.util.ArrayList<javax.swing.JProgressBar> | [**getJpb**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#getJpb--)() |
| java.lang.String | [**getMaxIntervalTextArea**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#getMaxIntervalTextArea--)() |
| java.lang.String | [**getMaxServiceTimeTextArea**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#getMaxServiceTimeTextArea--)() |
| java.lang.String | [**getMinIntevalTextArea**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#getMinIntevalTextArea--)() |
| java.lang.String | [**getMinServiceTimeTextArea**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#getMinServiceTimeTextArea--)() |
| int | [**getNbOfQueues**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#getNbOfQueues--)() |
| java.lang.String | [**getNbOfQueuesTextArea**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#getNbOfQueuesTextArea--)() |
| java.lang.String | [**getSimIntervalTextArea**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#getSimIntervalTextArea--)() |
| java.lang.String | [**getSimTimeText**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#getSimTimeText--)() |
| void | [**resetView**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#resetView--)()  reseteaza textArea, SimTime-ul si progressBarurile |
| (package private) void | [**restartListener**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#restartListener-java.awt.event.ActionListener-)  (java.awt.event.ActionListener restart) |
| (package private) void | [**setListener**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#setListener-java.awt.event.ActionListener-)  (java.awt.event.ActionListener set) |
| void | [**setLog**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#setLog-java.lang.String-)(java.lang.String s) |
| void | [**setLogNull**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#setLogNull--)() |
| Void | [**setNbOfQueues**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#setNbOfQueues-int-)(int nbOfQueues) |
| void | [**setQueues**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#setQueues-int-)(int nb)  contruieste progressBarurile in functie de cate queue-uri introduce utilizatorul |
| void | [**setSimTime**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#setSimTime-java.lang.String-)(java.lang.String simTimeTextArea) |
| (package private) void | [**showError**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#showError-java.lang.String-)(java.lang.String errMessage)  afiseaza un pop up cu mesaj de eroare |
| (package private) void | [**startListener**](file:///C:\Users\Vlad\Desktop\Vlad%20Brincoveanu@An2-Sem1\JAVA%20WORK\tema2\doc\tema2\View.html#startListener-java.awt.event.ActionListener-)(java.awt.event.ActionListener start) |

# Concluzii si Dezvoltari Ulterioare

Acest proiect m-a familiarizat cu threadurile si in special cu problema producer consumer, pe care am observant ca este foarte utilizata in aplicatiile de zi cu zi . Threadurile sunt folosite peste tot , in toate aplicatiile, pentru eficientizarea lor. Astfel, simularea cozilor , se face eficient , fiecare coada ruland in paralel fata de cealalta.

Proiectul poate sa fie implementat mai eficient prin alegerea mai buna a anumitor timpi a clientilor.

Am evitat instantierea threadurilor in mod normal, si am folosit clasa ExecutorService care este mult mai eficienta in aceasta privinta, stiind cand sa ruleze threadurile si avand si metode de oprire , delay etc.

As fi putut sa transmit direct din generator clientul, prin magazin, in coada, fara a le mai stoca pe parcurs , astfel reducand drastic timpul parcurs de client prin fiecare clasa.

Interfata se putea construi, estetic, mai frumos, folosind partea grafica a swingu-lui modificand componentele .

In concluzie, acest proiect este benefic oricarui student care invata programarea orientat obiect, intrucat ii prezinta o conceptie puternica , foarte folosita , numita multi-threading.

# Bibliografie

<https://stackoverflow.com>

<https://en.wikipedia.org/wiki/Thread_(computing)>

<http://coned.utcluj.ro/~marcel99/PT/3%20-%20Presentation%20-%20Java%20Concurrency-3.pdf>

<https://www.tutorialspoint.com/javaexamples/thread_procon.htm>

https://en.wikipedia.org/wiki/Producer–consumer\_problem