Queues Simulator

Nita Iosif-Gabriel

Grupa 30223

# CUPRINS

[1. Obiectivul temei 3](file:///C:\Users\Gabriel\Downloads\PT2021-2022_Documentation_Template%20(2).doc#_Toc95297885)

[2. Analiza problemei, modelare, scenarii, cazuri de utilizare 3](file:///C:\Users\Gabriel\Downloads\PT2021-2022_Documentation_Template%20(2).doc#_Toc95297886)

[3. Proiectare 3](file:///C:\Users\Gabriel\Downloads\PT2021-2022_Documentation_Template%20(2).doc#_Toc95297887)

[4. Implementare 3](file:///C:\Users\Gabriel\Downloads\PT2021-2022_Documentation_Template%20(2).doc#_Toc95297888)

[5. Rezultate 3](file:///C:\Users\Gabriel\Downloads\PT2021-2022_Documentation_Template%20(2).doc#_Toc95297889)

[6. Concluzii 3](file:///C:\Users\Gabriel\Downloads\PT2021-2022_Documentation_Template%20(2).doc#_Toc95297890)

[7. Bibliografie 3](file:///C:\Users\Gabriel\Downloads\PT2021-2022_Documentation_Template%20(2).doc#_Toc95297891)

**Obiectivul temei**

Obiectivul acestei teme este proiectare si implementarea unei aplicatii care simuleaza problema cozilor intalnita zilnic in viata de zi cu zi. Aceasta problema este problema cozilor.

Aplicatia dezvoltata rezolva aceasta problema, determinand timpul minim necesar de asteptare al clientilor.

1. Dezvoltarea unor scenarii in care aplicatia va fi dezvoltata
2. Alegerea structuriilor de date, structura proiectului trebuie sa fie una eficienta
3. Impartirea corespunzatoare a claselor si a rolurilor acestora
4. Implementarea si dezvoltarea algoritmilor corespunzatori pentru toate cazurile
5. Implementarea solutiilor, combinarea algoritmilor
6. Testarea si gasirea cazurilor care pot cauza probleme aplicatie, remedierea pe cate se poate a acestora.

Analiza problemei

Cozile sunt unele dintre cele mai intalnite probleme din viata de zi cu zi. Obiectivul acestora este de a furniza un loc de asteptare pentru scopul obtinerii unui serviciu. Aceste sisteme trebuie administrate in asa fel incat timpul petrecut la coada sa fie minimizat, o metoda simpla de rezolvare ar fi angajarea mai multor operatori, dar aceasta aboradare creste costul serviciilor oferite.

Ca si date de intrare care trebuie inserate de catre utilizator în interfata grafica avem urmatoarele:

* Numarul de clienti (N);
* Numarul de cozi (Q);
* Intervalul de simulare;
* Timpul de sosire minim si maxim;
* Timpul de servire minim si maxim.

Folosind datele de intrare pe care le furnizeaza utilizatorul vom genera N clientri cu ID-ul incepande de la 1 si pana la numarul ales de client, cu timpul de sosire ales aleator din intervalul propus de utilizator la fel se va proceda si la timpul de servire tot aleator se va alege din celalat interval.

Se vor porni un numar de cozi in parallel care vor primi clienti pentru a fi procesati. Un alt thread va fi pornit pentru a sustine pasii de simulare si pentru a distribui fiecare client la coada cu cel mai mic timp de astepatre cand timpul de simulare este egal cu timpul de sosire al celui client.

# Proiectare(decizii de proiectare, structuri de date utilizate, interfete, packages, algoritmi, interfata utilizator)

|  |  |
| --- | --- |
| **Package** | **Class** |
| main -> **java** | MVC |
| main -> java -> DataM**odel** | Customer  MasterPlan  Simulator  Scheduler  MyQueues |
| main -> java -> Controller | ControllerQueues |
| main -> java -> View | View |

|  |  |
| --- | --- |
| **Class** | **Descriere** |
| Customer | Clasa are trei atribute principale, respectiv ID-ul, timpul de sosire și timpul de servire, un constructor și metodele corespunzătoare de set și get. Am mai adaugat suprascrierea metodei compareTo si suprascrierea metodei toString. |
| MyQueues | Clasa are patru atribute principale, respectiv lista de clienti, timpul de asteptare, ID-ul corespunzator și o variabila care sa ne arate dacă este deschisa coada sau nu, un constructor și metodele standard de set și get. Pe langa metodele de bază am adaugat o metoda pentru a adauga clienții la coada, o metodă care suprascrie metoda de run din interfata Comparable, o metoda care ajuta la opreirea thread-urile si metoda suprascrisa toString. |
| MasterPlan | Clasa nu are niciun atribut, dar are o singura metoda importanta, metoda cu ajutorul careia adaug clientii la coada cu cel mai mic timp de asteptare. |
| Scheduler | Clasa are patru atribute principale, numarul de clienți, numarul de cozi, lista de cozi si un obiect de tipul clasei MasterPlan. Pe langa metodele de baza de set ,get și constructorul, am adaugat o metoda care apeleaza metoda de adaugare din clasa MasterPlan, o metodă care verifică daca toate cozile sunt goale și metoda suprascrisa toString. |
| Simulator | Clasa are ca atribute toate datele de intrare din cerința problemei plus câteva atribute care să mă ajute la implementare: timpul mediu de așteptare, timpul mediu de servire, ora de vârf, un obiect de tipul Scheduler și lista de clienți generată aleator. Pe lângă metodele de bază și constructor, am adăugat metoda care generează clienții aleator, metoda run și metoda suprascrisă toString. |
| View | Clasa are ca atribute toate componentele de la nivel de interfata care necesita manipularea ulterioara. Contine constructorul interfetei, metode care returneaza textul din TextField-uri, metode care setează rezultatul in eticheta corespunzătoare si in TextArea si metoda care adauga ascultatori pentru butonul de start si reset. |
| ControllerQueues | Clasa care leagă clasele din pachetul model cu clasa View. Are un singur atribut, și anume un obiect de tip View. În cadrul constructorului inițializez atributul de tip View si asociez metodele ascultătoare pentru butonul de start si cel de reset cu o clasa anonima. |
| MVC | Clasa care contine metoda main. Aici creez un obiect de tip View pe care il pasez constructorului SimulatorManager-ului. |

Alegerea proiectarii claselor in acest mod m-a ajutat sa inteleg legaturile dintre clase si sa dezvolt metodele necesare pentru dezvoltarea aplcatiei. In pachet-ul DataModels am pus clasele Customer, MasterPlan, MyQueues, Scheduler si Simulator pentru ca acestea se ocupa de probleme mai importante in aplicatia dezvoltata. In pachetul “View” am pus clasa View in care se afla interfata care operationeaza cu utilizatorul si metodele necesare, iar in pachetul Simulator am pus doar clasa SimulatorManager care citeste/afiseaza informatiile dorite

Diagrama UML:

O imagine care conține text, interior, negru

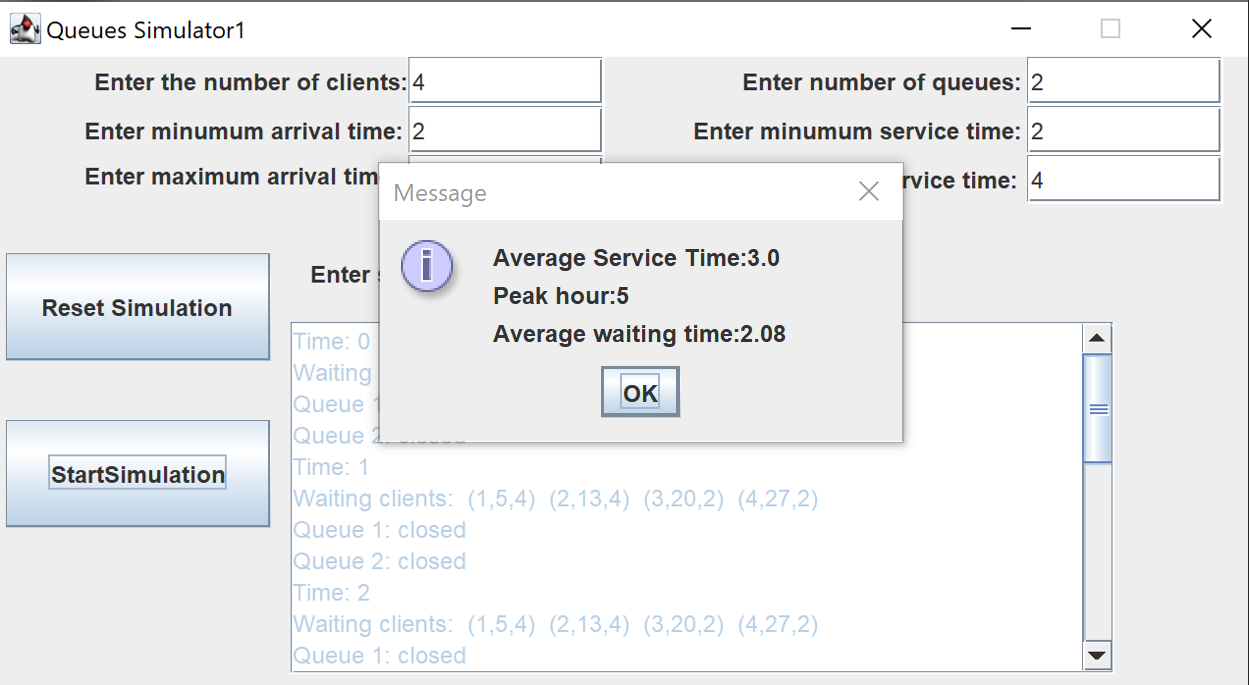
Descriere generată automat

Interfata grafica:

O imagine care conține text

Descriere generată automat

Asa este afisat rezulatul:



Implementare

Clasa Customer

Clasa Customer are trei atribute principale care descriu clientul si inca un atribut static care este de folos atunci cand setez ID-ul primului client. In constructor cu ajutorul metodei randomArrivSerTime creez cu ajutorul intervalurilor setate de utilizator din interfata grafica un timp de servire random si un timpul la care ajunge clientul acesta tot random fiind.

O imagine care conține text

Descriere generată automat

Clasa MyQueues

Clasa MyQueues reprezinta thred-urile fiecarei cozi corespunzatoare unei liste de clienti si are 6 atrubute, cele mai importante ar fi lista de client si timpul de asteptare pentru fiecare coada in parte. ID-ul fiecare cozi doua variabile Boolean una pentru a ma ajuta sa tin deschisa coada. In constructor instantiez fiecare atribut in parte

O imagine care conține text

Descriere generată automat

Metoda run care este o metoda suprascrisa reprezinta functionalitatea unui thread reprezentat de o coada la care clientii o sa intre si sa astepte. Atat timp cat mai sunt clienti in lista li se ia timpul de servire al primului client sosit(capul cozii) si dupa pune thread-ul la somn pentru o secunda, aceasta ne ajuta sa nu mai folosim metoda synchronized. Dupa ce acesta se trezeste, timpul de asteptare al cozii este decrementat, decrementam si timpul de servire al clientului. Daca timpul de servire devine zero clientul paraseste coada.

O imagine care conține text

Descriere generată automat

Metoda addCustomer care adauga in lista de client un client, acesata seteaza coada ca fiind deschisa adauga clientul in lista si seteaza noul timp de asteptare

O imagine care conține text

Descriere generată automat

Inca 2 metode destul de importante este metoda de afisare si metoda care seteaza variabila run pe false, aceaste metoda practic opreste thred-ul

O imagine care conține text

Descriere generată automat

Clasa MasterPlan

Clasa MasterPlan are o singura metoda care reprezinta metoda care adauga un client intr-o lista de cozi. Salvez id-ul primei cozi si timpul de asteptare si daca gasesc o coada cu un timp mai mic de asteptare adaug clientul la aceea coada aceasta avand cel mai mic timp.

O imagine care conține text

Descriere generată automat

Clasa Scheduler

Clasa Scheduler are ca parametri lista de cozi numarul de client numarul de cozi si o legatura cu clasa MasterPlan. Metoda startThread care este apelata in constructor pe langa initializarea fiecarui atribut creeaza coada o adaug in lista si dupa pornesc thred-ul corespunzator cozii create

O imagine care conține text

Descriere generată automat

Metoda sendCustomer apeleaza metoda de adaugare din clasa MasterPlan pentru un client anume si pentru lista din clasa Scheduler

Metoda qClosed acesata metoda returneaza un Boolean daca cozile sunt inchise. Aceasta metoda este folositoare daca toate cozile sunt inchise si nu mai este nici un client pentru a opri simularea.

O imagine care conține text

Descriere generată automat

Metoda Simulator

Metoda Simulator are ca atribute datele de intrare de la interfata pe care o folosete utilizatorul, alte atribute necesare.

In constructor instantiez atributele corespunzatoare si apelez o metoda care genereaza clientii.

O imagine care conține text

Descriere generată automat

Metoda generateCustomers genereaza clientii si ii seteaza in lista, Arrival time si Service time sunt generati random din constructorul clasei Customer. Dupa ce clientii au fost generate acesti vor fi sortati cu ajutorul Colectiei sort, clasa Customer implementeaza aceasta metoda specifica compareTo. Deasemenea in aceasta metoda mai apelez o metoda avgServTime care calculeaza timpul mediu de servire acesta fiind suma timpiilor de servire a clientilor impartita la numarul acestora.

O imagine care conține text

Descriere generată automat

Metoda run care creeaza un fisier in care scriem rezultatele si apelam alte doua metode foarte importante in dezvoltarea aplicatiei.

Metoda mainLoop in care se gaseste un loop-ul principal care ruleaza atata timp cat timpul de simulare este mai mic sau egal cu variabila cTime.

In interoriul acestui for se afla un while care atat timp cat coada nu este goala si Arrival time este egal cu timpul current(cTime) apeleaza metoda de adaugare din clasa Scheduler care trimite capul cozii (coada fiind sortata), adica primul client, la cea coada cu cel mai mic timp de asteptare. Dupa calculez ora de varf ca fiind suma clientilor la momentul respectiv de timp din fiecare coada in parte. Daca suma calculata este mai mare decat suma calculate anterior setez variabila auxilara g cu valoarea sumei calculate iar ora de varf ia valoare timpului current.

O imagine care conține text

Descriere generată automat

Dupa afisez in interfata grafica pentru a fi vizibila miscarea cozii in timp real de catre utilizator.

Daca lista de clienti este goala oprim aplicatia.

Punem thread-ul la somn pentru sincronizarea aplicatiei.

Metoda afisare scrie timpul de asteptare mediu(acesta va fi suma tuturor timpurilor de asteptarea a tutoror cozilro la fiecare moment, impartit la cati client au fost in total la toate cozile), timpul mediu de servire si ora de vraf atat in interfata grafica cat si in fisier.

O imagine care conține text

Descriere generată automat

Metoda ControllerQueues

Aceasta clasa contine un singur atribut acesta fiind o instanta a clasei View. In interiorul constructorului acestei clase, ControllerQueues, se intializeaza singurul atribut adica view se implenteaza ascultatorii pentru butoanele interfetei.

O imagine care conține text

Descriere generată automat

In clasa secundara StartSimListener se creeaza un obiect de tipul Thread si unul de tipul Simulator se verifica apasarea butonului din interfata grafica si se citeste imput-ul utilizatorului obiectul de tip simulator se intializeaza cu variabilele create iar obiectul thred se intializeaza cu obiectul de tip simulator apoi thread-ul este pornit. Afisarea constanta a rezultatelor la fiecare clipa a simularii este realizata de thred-ul de simulare.

O imagine care conține text

Descriere generată automat

In clasa secundara ResetSimListener se verifica apasare butonului si se reseteaza fiecare casuta din interfata si este resetat si index-ul clasei MyQueues si al fiecarui client din clasa Customer.

O imagine care conține text

Descriere generată automat

Clasa View

Clasa View care implementeaza JFrame are ca atribute toate componentele din cadrul interfetei grafice.

O imagine care conține text

Descriere generată automat

In interiorul constructorului este creata fereasta si setata dimensiunea acesteia si sunt intializate toate componentele interfetei, in interior este declarant si un jScrollPane pentru a face citirea datelor din interfata mai usoara. Este adaugat fiecare element in fereastra.

O imagine care conține text

Descriere generată automat

In aceasta clasa mai sunt declarate metodele de get si am adaugat butoanelor ActionListener pentru a putea fi utilizate corespunzator.

O imagine care conține text

Descriere generată automat

Clasa MVC

Clasa MVC contine metoda main in aceasta metoda se afla un obiect de tipul View care este argumentul obiectului de tipul ControllerQueues si este setata interfata grafica pentru a putea fi vizibila utilizatorului.

O imagine care conține text

Descriere generată automat

Rezultate Testare

Am testat implementarea folosind datele de intrare propuse de materialul ajutator prezentat la laborator.

Rezultatul pentru primul test, cel cu 4 clienti este:

O imagine care conține text

Descriere generată automat

Rezultatul pentru cel de al doilea teste si anume acel cu 50 de client:

O imagine care conține text

Descriere generată automat

Iar rezultatul pentru testul final cel cu 1000 de client este:

O imagine care conține text, masă

Descriere generată automat

Concluzii

In urma dezvoltarii acestei aplicatii, am inteles mai bine conceptele din spatele realizarii unei cozi cat mai eficient cu putinta am aprofundat cunostintele din semestrul trecut si am acumulat constinte noi care vor fi folositare pentru viitor si anume utilizarea si manipularea thred-urilor. Dezvoltarea treptata ma ajutat sa inteleg mai bine modul de gandire si sa gandesc la randul meu stric la oiectivul problemei si la un mod cat mai usor de inteles si de implementat.

Dezvoltarile ulterioare probabil ar fi adaugarea execptiilor si mesajelor de eroare daca utilizatorul nu introduce bine datele de intrare si o interfata mai placuta ochilor.

Bibliografie

Nu am avut nici o sursa din care sa ma inspir la redactarea acestei documentatii. Acestea sunt ideile mele care au venit pe parcursul proiectarii si implemetarii proiectului si anumite ideei care mi-au venit pe moment sau am citit in documentatia proiectului pentru a-mi limpezi unele neclaritati in legatura cu thread-urile sau alte chestiuni putin mai complicate la prima vedere.