Tema 2 – TEHNICI DE PROGRAMARE

*Sistem de management al unui restaurant*

Student: Bologa Horatiu

1. Obiectivul temei . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

2. Analiza problemei, modelare, scenarii, cazuri de utilizare . . . . . . . . . . .

3. Proiectare . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

* decizii de proiectare . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
* diagrame UML . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
* structuri de date . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
* proiectare clase . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
* interfețe . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
* relații . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
* packages . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
* algoritmi . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
* interfața utilizator . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

4. Implementare și testare . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

5. Rezultate . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

6. Concluzii . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

7. Bibliografie . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

1. **OBIECTIVUL TEMEI**

Scopul acestei teme este de a implementa o aplicație care are ca scop simularea unei ordini a clienților. Sistemul de cozi are ca rol principal diminuarea intervalului de timp în care clienți care își așteaptă rândul au fost serviți. De aceea, proiectul urmărește realizarea acestei aplicații într-un mod eficient. Acest lucru se poate realiza apelând la următoarea strategie: se adaugă mai multe cozi, astfel, clienți sunt serviți mult mai rapid și nu timpul de așteptare a clienților scade. Prin intermediul acestei strategii, se facilitează servirea și se sporește numărul de comenzi. Când adăugăm o nouă coadă, clienții sunt distribuiți în mod egal tuturor cozilor disponibile.

Aplicația are ca scop simularea unei mulțimi de clienți care intră în coadă pentru a fi serviți, așteaptă un anumit timp după care sunt serviți, iar la final se decrementează numărul de clienți din coadă, întrucât clientul părăsește locul de așteptare deoarece el și-a terminat tema. Se urmărește în scop principal timpul pe care clienții îl petrec la coadă. Astfel, pentru ca să putem urmări acest lucru avem nevoie de cunoașterea timpului de sosire a clienților, timpului de servire și timpul de așteptare. Timpul de sosire și timpul de servire depinde de fiecare client, însă timpul de simulare nu depinde în mod direct de el deoarece se raportează în mod principal la numărul de cozi, la numărul de clienți. Parametrii de care depinde timpul de simulare nu sunt prestabiliți însă se iau în mod direct de pe frame-ul principal.

Datele de livrare esențiale sunt următoarele:

* Numărul de clienți;
* Numărul de cozi;
* Timpul limită;
* Timpul maxim de servire;
* Timpul minim de servire;
* Timpul maxim de sosire;
* Timpul minim de sosire.

Dacă aplicația simulează în mod corect și respectă cerințele impuse, pe ieșire ar trebui să fie afișate cozile în timp real, așa cum evoluează ele, cu clienții care intră în coadă și care ies din coadă.

1. **PROIECTARE**

Pentru a facilita stocarea elementelor, acest proiect se bazează întrutotul pe colecțiile din Java. Una dintre cele mai esențiale colecție folosită se referă la ArrayBlockingQueue care are rolul de a stoca taskurile dintr-un server si de a se ocupa cu managementul lor in contextul utilizării firelor de execuție separate si astfel sincronizării datelor intre acestea deoarece pot apărea probleme in cazul accesărilor simultane de informații. De asemenea, o altă colecție esențială este reprezentată de ArrayList, avantajul principal al acestei colecții fiind reprezentat de faptul că se poate accesa elementele cu ușurință din acesta. Aplicația nu prezintă niciun algoritm la baza ei, ea fiind implementată doar pe baza raportării la viața reală și făcându-se analogii cu acestea, punându-se mai apoi logica în cod. Singura dificultate a acestei teme este reprezentată de utilizarea mai multor fire de execuție și utilizarea a mai multor threduri din aplicație.

După cum spuneam , aplicația se bazează pe cozi si pe threduri. Conceptul de coadă definește o structură liniară care lucrează cu anumite colecții de date. Fiind o coadă, operațiile care se efectuează asupra acestora se realizează doar la capetele cozii. Astfel, coada prelucrează elementele sub formă de FIFO (Primul venit, primul servit), ceea ce se potrivește perfect aplicației noastre, drept pentru care s-a decis utilizarea ei. Asupra cozii se pot implementa următoarele operații:

* Push – această funcție are ca scop adăugarea unui element în coadă. Când coada nu mai are locuri disponibile, se aruncă un mesaj de eroare care afirmă faptul că coada este prea plină. De asemenea, locul în care este introdus elementul în coadă se află la finalul acesteia pentru a se respecta principiul FIFO.
* Pop- această funcție are ca scop ștergerea elementului din coadă. Ștergerea are loc la începutul cozii.
* Front – este funcția care afișează primul element din coadă. Este foarte asemănătoare cu peek, care returnează primul element din coadă fără a îl șterge
* isEmpty – verifică dacă lista este goală.

Aplicația mai are la bază și utilizarea thredurilor. Asemănătoare cu procesele, thredurile reprezintă un mecanism prin care un calculator poate să ruleze simultan. Pentru a putea executa thredurile, se folosesc următoarele metode de control: start(cu rolul de a începe execuția unui thred), sleep(cu rolul de a pune în repaus un thred), setPriority(cu rolul de a stabili prioritatea thredurilor), getPrirority(culege prioritatea unui thred). Există doua modalități de creare a unui thred: prin intermediul clasei Thread sau implementând interfața Runnable.

Avantajele principale ale thredurilor îl reprezintă următoarele lucruri:

* crearea/distrugerea unui thread durează mai puțin decât crearea/distrugerea unui proces
* timpul context switch-ului între thread-urile aceluiași proces este foarte mic, întrucât nu e necesar să se "comute" și spațiul de adrese;
* comunicarea între thread-uri are un overhead minim (practic se face prin modificarea unor zone de memorie din spațiul de adresă)
* ele simplifică structura unui program și ajută la utilizarea unui număr mic de resurse.

Diagrama UML

Unified Modeling Language (UML) este un limbaj standard pentru descrierea de modele și specificații pentru software. Acesta a fost dezvoltat pentru reprezentarea complexității programelor orientate pe obiect, al căror fundament este structurarea programelor pe clase și instanțe ale acestora (obiecte).

Diagrama pe clase este o structură statică care descrie de fapt structura unui sistem prezentând clasele sale, atributele acestora, metodele conținute în fiecare clasă și relațiile dintre obiecte.

3.3 Pachete și clase

Acest proiect este structurat în mai multe pachete: Control, GUI, Model. Nucleul central al acestui proiect se afla în interiorul pachetului Model.

Fiecare pachet este structurat în modul următor:

* Control – conține: ConcreteStrategyQueue, ConcreteStrategyTime, Main, Strategy.
* Gui- conține interfața.
* Model – Client, Manager, Que.

Deoarece Model este cel mai important, în interiorul acestuia am declarat subiecții principali ai acestei aplicații.Astfel, în clasa client avem în felul următor: variabilele pentru identifficarea acestuia (id ul clientului, timpul de sosire, și timpul de servire), constructorul principal al clasei precum și settere și gettere pentru a putea fi accesate variabilele private și în exteriorul acestei clase.

În interiroul clasei Manager s-au definit următoarele variabile care se găsesc pe interfața principală a acestui proiect. Limita de timp numărul de cozi, numărul maxim de servire, numărul minim de servire, numărul minim de sosire, numărul maxim de sosire. În această clasă are loc randomizarea clienților. Id-ul clienților trebuie să fie între 1 si numărul introdus în caseta de textfield de pe interfață. Randomizarea clienților se face cu ajutorul funcției Random, bazându-se pe minimul și maximul timpului de servire, precum și timpul de sosire.

În main are loc pornirea thredului principal. Viața unui thred oarecum se află în interiorul funcției run, deoarece clasa Manager folosește Runnablle.

1. **CONCLUZIE**

Ca o concluzie generală asupra acestui proiect pot afirma că m-a ajutat să îmi dezvolt abilitățile de a programa în Java. De asemenea, ce am constatat la acest proiect a fost faptul că cea mai grea parte a consituit-o lucrarea cu threduri. Astfel, fiindcă proiectul s- a bazat pe multe analogii cu viața reală, acest lucru a reprezentat un punct forte deoarece scrierea codului a devenit mult mai simplă pentru că totul a venit de la sine. Totuși, este de menționat faptul că thredurile nu reprezintă niște funcții ușor de manipulat.

Comparând cu celelalte proiecte, se pot observa următoarele îmbunătățiri.

* Implementarea interfeței a devenit mult mai complexă întrucât se folosesc mai multe JFrame uri care sunt legate între ele;
* Pachetul de resurse a fost gestionat mult mai ușor.
* Numărul de clase implementate este mult mai mic.
* Proiectul s- a realizat pe niște clase generice al căror cod s-a putut folosi în orice clasă.
* Dezvoltarea managementului de gestionare clienților și a cozilor.
* Preluarea comenzilor și gestionarea datelor clienților pentru o bună identificare a comenzilor.
* Structurarea codului pe clase astfel încât înțelegerea codului se poate face cu ușurință și fără probleme.
* Scrierea codului într-un mod structural, astfel încât fiecare clasă să reprezinte