Muntean Andrei

30227

**SIMULAREA COZILOR**

**CERINTA:**

Proiectati si implementati o aplicatie java care sa simuleze un sistem de cozi si sa afiseze timpul de asteptare al clientilor pentru fiecare ghiseu.

**OBIECTIVE:**

Se va proiecta si implementa un program cu ajutorul caruia sa se simuleze cozi.

Se va construi un modul aditional aplicatiei care va genera informatii despre obiectele folosite in intreaga aplicatie dar si evolutia in timp a acesteia.

**ANALIZA PROBLEMEI:**

Cozile sunt des intalnite atat in viata reala cat si in modele.Obiectivul principal este de a repartiza un loc de asteptare pentru un client inainte de a fi servit. Sistemul de manipulare al cozilor se bazeaza pe minimizarea timpului de asteptare al unui client in cadrul unei cozi. Un mod de minimizare al timpului este de adaugare de cozi in sistem.

Sistemul trebuie sa simuleze o serie de clienti care ajung la case, intra in cozi, asteapta in ele, sunt serviti dupa care parasesc coada. Aplicatia urmareste timpul mediu pe care clientii il petrec la fiecare coada.

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 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 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 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 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 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 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 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 A a a a a a a a a a A a a a a a a a a a

Pentru calcularea timpului, avem nevoie de timpul de sosire, timpul de iesire si timpul de servire. Timpul de sosire si de asteptare este specific pentru fiecare client (cand intra in coada si cat timp va dura sa fie serviti). Timpul final de iesire depinde de numarul de cozi, numarul de clienti din fiecare coada si de timpul de servire de care au nevoie clientii din coada.

Datele de intrare ale problemei sunt: numarul de clienti; intervalul de sosire minim si maxim; timpul de servire minim si maxim si numarul de cozi.

Tema acestui task este de a implementa un sistem ce utilizeaza coada atat din punct de veder algoritmic, cat si grafic. Este un subiect comun in zilele noastre si acum a devenit o problema majora in toata lumea. Aceasta aplicatie poate fi folosita pentru a ajuta oamenii care sunt in functii de conducere sau sunt responsabili cu repartizarea clientilor intr-un mod eficient la casele de marcat. Ii poate ajuta pentru a realiza cea mai buna situatie pentru a nu-i face pe clienti nervosi datorita asteptarii la o coada pentru o perioada mai lunga de timp. Totodata, aceasta aplicatie furnizeaza ajutor si suport pe baza rezultatelor corecte ori de cate ori o persoana are nevoie de ele. Aplicatia are la baza o interfata utilizator prietenoasa si usor de folosit.

Cand fereastra principala apare pe ecran, utilizatorul va vedea un set de label-uri care indica datele de intrare si un set de textbox-uri, fiecare text field corespunzand unui anumit label. Mai exista si un buton care ruleaza programul atunci cand se da click pe el. Acest buton devine activ de la inceput, dar este dezactivat o data ce utilizatorul introduce date in text field-uri si da click pe butonul de start. Text box-urile corespund unui numar de clienti, timpul minim si maxim de sosire al clientilor (calculat in secunde), timpul minim si maxim de servire al acestora (exprimat de asemenea in secunde) si numarul de cozi care sunt disponibile clientilor. O data ce programul este pornit, pe ecran va aparea rezultatul (sper exemplu, timpul mediu de asteptare al clientilor) si o mica simulare a clientilor la casele disponibile. O data ce rezultatul este afisat, utilizatorul trebuie sa paraseasca aplicatia daca doreste sa o mai execute o data. Mai multe privind modul de functionare al programului sunt prezentate mai jos.

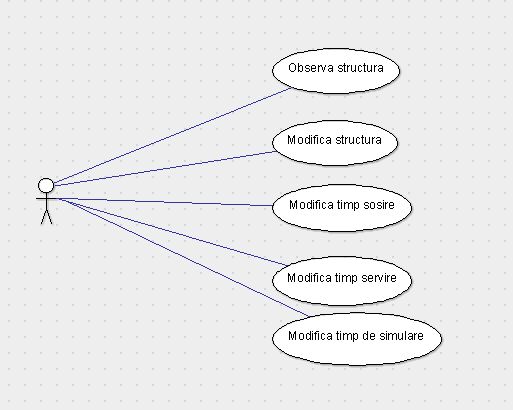
**ANALIZA APLICATIEI:**

Aceasta aplicatie va permite manipularea cozilor la ghiseele unei banci. Mai multe ghisee pot presta aceleasi servicii. Insa durata prestarii diferitelor servicii poate sa diefere. De exemplu dureaza mai mult deschiderea unui cont in banca decat crearea unui extras de cont. Astfel ghiseele se specializeaza, dar se pot „ajuta” reciproc. In aceste cazuri se formeaza cozi de clienti. Cu ajutorul aplicatiei noastre vom putea usor manipula aceste cozi.

Utilizarea aplicatiei:

* aplicatia la inceput ruleaza cu optiunile setate de autor;
* utilizatorul poate schimba numarul de clienti;
* utilizatorul poate schimba numarul de cozi;
* utilizatorul poate schimba intervalul de sosire a clientilor;
* utilizatorul poate schimba intervalul de servire a clientilor;
* utilizatorul poate schimba timpul de simulare

**DIAGRAMA USE – CASE:**

****

**PROIECTARE:**

Programul pune la dispozitie o interfata grafica usor de inteles. Astfel, aplicatia va fi formata din 4 componente: Queue – panoul unde are loc simularea cozilor formate la ghiseele unei banci, Options – fereastra ce contine elementele de manipulare a sistemului de cozi, Log – fereastra ce contine informatiile despre evolutia cozilor si Speed – fereastra cu care se poate modifica viteza simularii.

Queue – contine elemente grafice. Banca este vazuta in plan 2D de sus in jos. Clientii sunt reprezentati cu „smile-icuri” de trei tipuri, care in dependenta de timpul de servire se schimba. In acest panou mai avem o bara de instrumente cu ajutorul careia putem ajunge la celelalte ferestre sau putem iesi din aplicatie.

Options – contine elementele de simulare a cozilor prin: setare numarului clientilor, setarea timpului de sosire a clientilor la ghisee, setarea timpului de servire a clientilor si setarea timpului de simulare a cozilor.

Log – genereaza informatie despre clasele utilizate de catre aplicatie si informatie despre evolutia cozilor.

Speed – in aceasta fereastra se poate micsora timpul de simulare a cozilor, adica se mareste viteza de executie.

**ALGORITMI:**

Aplicatia data lucreaza dupa principiul cozii(FIFO – primul intra primul a iesit) si a stivei (LIFO – primul intra si ultimul a iesit). In continuare vom enumera pasii dupa care lucreaza programul dat.

Primul caz sa presupunem ca banca are mai multi lucaratori la ghisee. Vom stabili modul lor de lucru in urmatorul fel:

1. Un singur lucrator lucreaza;
2. Se verifica daca numarul clientilor nu a depasit o constanta oarecare MAX, in caz contrar incepe a lucra alt ghiseu;
3. Se verifica daca toate cozile active au dimensiune mai mare de MAX. Raspuns pozitiv se mai cheama un lucrator, in caz contrar se trece la pasul urmator;
4. Se verifica daca este vreo coada cu dimensiunea minima mai mica decat un o constanta MIN si daca se poate elibera un lucrator, daca este raspuns pozitiv se elibereaza un lucrator, in caz contrar se trece la pasul – 2;

Al doilea caz este pentru modificarea cozilor:

1. Se verifica daca nu s-a depasit capacitatea bancii. In cazul negativ se genereaza un nou client, in caz contrar se trece la pasul – 3;
2. Se cauta coada cea mai scurta si se adauga cliet la ea;
3. Se verifica daca s-a deservit un client. In cazul pozitiv se sterge clientul din coada, in caz contrar se trece la pasul – 1;

**MODELARE**

In continuare vom implementa algoritmul de mai sus cu ajutorul proiectarii si analizei aplicatiei. Pentru aceasta vom folosi mai multe clase: una pentru implementarea cozilor, una pentru o arhiva in care se va contine informatia generata de program, una pentru interfata, una pentru optiunile care le poate modifica utilizatorul, una pentru miscarile clientilor, una pentru un panel cu ajutorul caruia vom apela alte ferestre, una pentru dizainul interfetei si una pentru informatii despre autor si program.

Metodele acestor clase vor fi detaliate in continuare:

*Clasa About:*

* void activate() – metoda data ajuta la activarea vizualizarii acestei clase. Corect zis aceasta metoda ajuta la pornirea frame-ului care contine date despre program si autor;

*Clasa Archive:*

* void activate() – metoda data ajuta la activarea vizualizarii acestei clase. Astfel, ea porneste frame-ul ce contine informatii referitoare la simularea cozilor de clienti;
* void describeClass(Object object) – metoda ce descrie aceasta clasa, modul de afisare a informatiei;
* void SetText(String s) – metoda care va permite afisarea textului in frame;

*Clasa Costumer:*

* void setNervous() – metoda folosita pentru afisarea clientilor care asteapta foarte mult timp in coada cu ajutorul unei „smyle” care este nervos;
* void setAngry() – metoda folosita pentru desemnarea clientilor care sunt suparati din cauza timpului pierdut in coada, care sunt la limita de nervozitate;
* void setHappy() – metoda de afisare a clientilor care sunt bine dispusi care petrec in coada foarte putin timp;
* void incX() si void incY() – metoda pentru stabilirea coordonatelor clientului in sala;
* void initTime() – metoda pentru stabilirea timerului;

*Clasa Main:*

* void init() – metoda data permite afisarea frame-ului de baza, care va contine o bara de instrumente din care se vor accesa celelalte ferestre;

*Clasa Mover:*

* void run() – reseteaza coordonatele clientilor in coada si invoca redesenarea;

*Clasa Panel:*

* void initTime(int period) – metoda cu ajutorul caruia stabilim timpul general;
* String putTime() – permite introducerea timpului de sosire si de servire a clientilor la ghisee in minute si secunde;
* void init() – metoda ce sincronizeaza time-erele si porneste in intregime programul;
* void Start() – la activarea acesteia incep sa mearga clienti la ghisee, astfel, se incep a forma cozi;
* void viewArchive() – se activeaza arhiva cu informatie;
* void viewSpeed() – se activeaza fereastra ce raspunde de modificarea timpului programului;
* void viewSettings() – se activeaza optiunile pe care utilizatorul le poate schimba;
* void run() – porneste in mod concurent metoda init si metoda Start;

*Clasa Queue:*

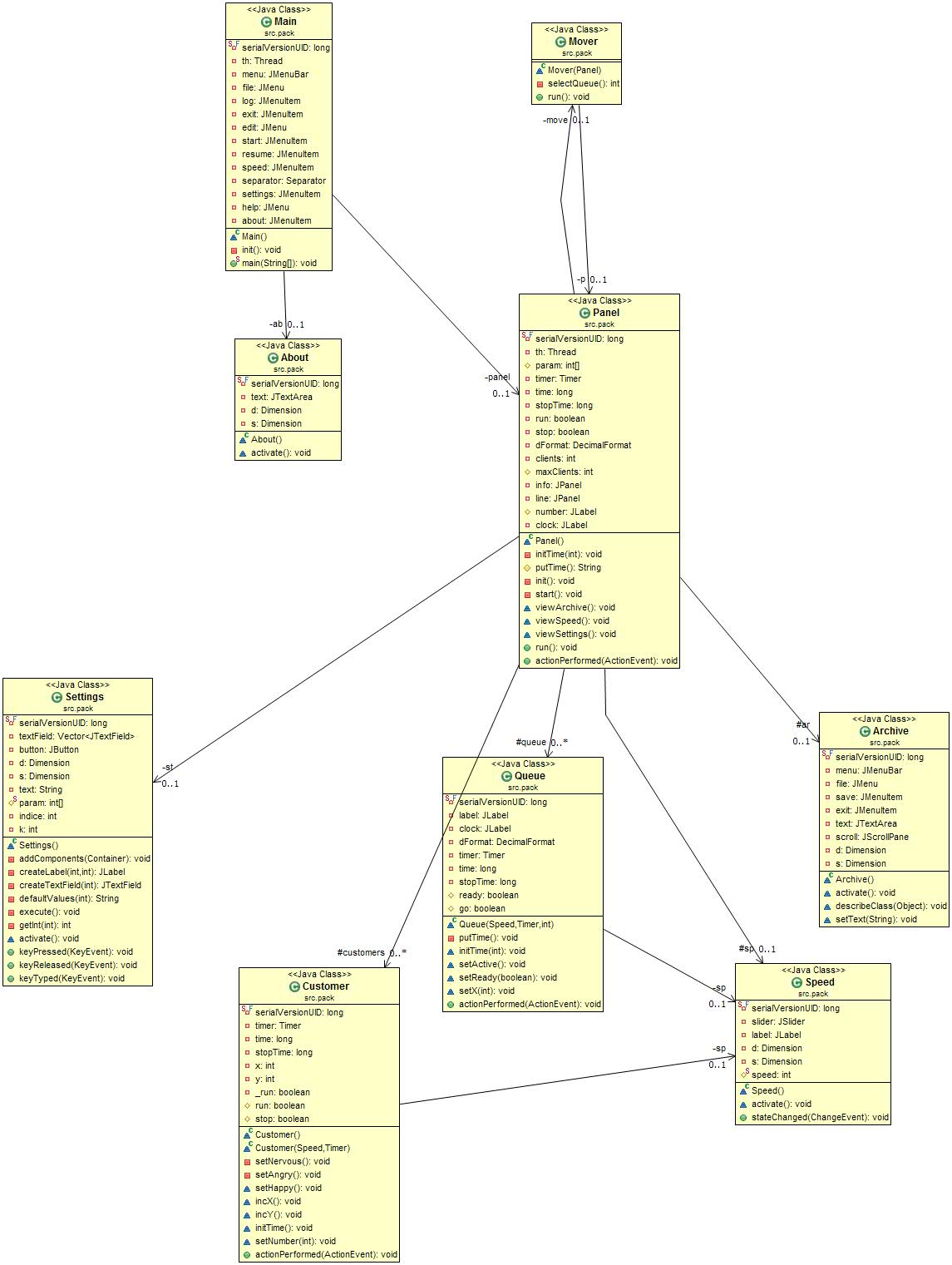
* void putTime() – se introduce timpul in minute si secunde;
* void initTime(int period) – se introduce timpul total de simulare a cozilor;
* void setActive() – se activeaza ghiseele;
* void setReady() – daca ghiseele au clienti atunci ele sunt active, in caz contrar nu;
* void setX(int x) – sunt date coordonatele ghiseelor;

*Clasa Settings:*

* void addComponents(Container pane) – se va aduga un container in care se va contine toate optiunile ce se pot modifica;
* JLabel createLabel(int i, int tip) – se va crea un camp inaintea textField-urilor unde se va scrie ce trebuie de introdus in acel loc;
* JTextField createTextField(int i) – se creeaza campul unde va trebui introdusa informatia;
* String defaultValues(int i) – metoda data va pune in campurile testFiled-urilor niste valori predefinite;
* void execute() – se vor lua toate datele introduse si se vor verifica daca sunt corecte;
* void activati() – metoda data ajuta la activarea vizualizarii acestei clase;

*Clasa Speed:*

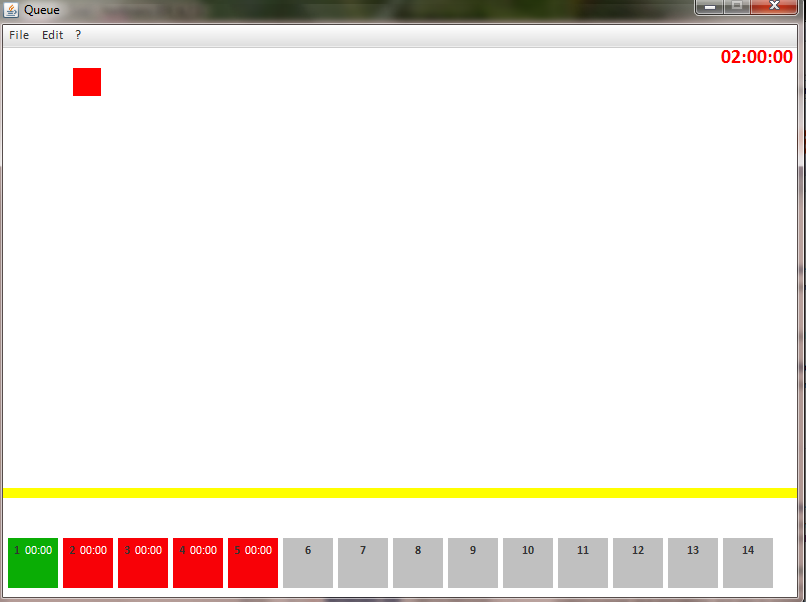
* void activati() – metoda data presupune vizualizarea ferestrei in care vom putea micsora sau mari timpul de simulare a programului

**DIAGRAMA DE CLASE**

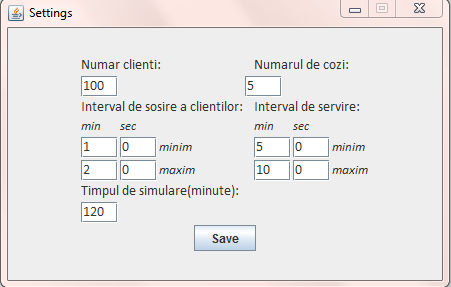
**IMPLEMENTARE SI TESTARE**

In rezolvarea acestui program am folosit mai multe medii de programare Java: Eclipse si NetBeans. Mediul de programare Eclipse l-am folosit si la crearea diagramelor de clase si use – case, pe care le genereaza din program.

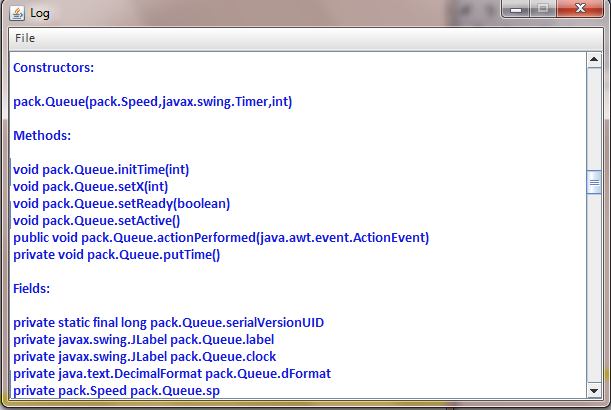
In imaginea urmatoare este reprezentat cadrul principal al aplicatiei unde este reprezentata simbolic banca:

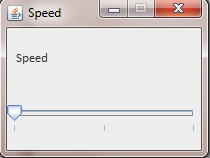


Urmatoarea componenta care ne ajuta efectiv sa testam ce am implementat este componenta Settings, de unde putem modifica intreg sistemul(de fapt, tot ce tine de cozi). Settings este in figura de mai jos.



Vom mai avea si urmatoarele componente:



****

**CONCLUZII**

In concluzie proiectul de fata este un bun program pentru simularea unui sistem de cozi si afisarea timpul de asteptare al clientilor pentru fiecare ghiseu, reprezinta un solid punct de pornire pentru aplicatii ulterioare care il pot dezvolta cu usurinta

**BIBLIOGRAFIE**

- <http://users.utcluj.ro/~jim/OOPR/>

- <http://java.sun.com/>