Management de cozi
Tehnici de programare
Tema 2

Alexandru Nistor Botolan
An 2 Seria B Grupa 30227
Facultatea de Automatica si Calculatoare
Sectia Calculatoare si Tehnologia Informatiei

1.Obiectivul temei

Obiectivul principal al temei este de a crea un management de cozi care are ca scop simularea pe o perioada determinata plasarea clientilor la cozi.

Obiectivele secundare sunt reprezentate de realizarea unei interfete grafice pentru a introduce parametrii simularii, afisarea statisticilor si a evenimentelor importante din cadrul simularii, dar si de a vizualiza decursul simularii prin intermediul unei animatii.

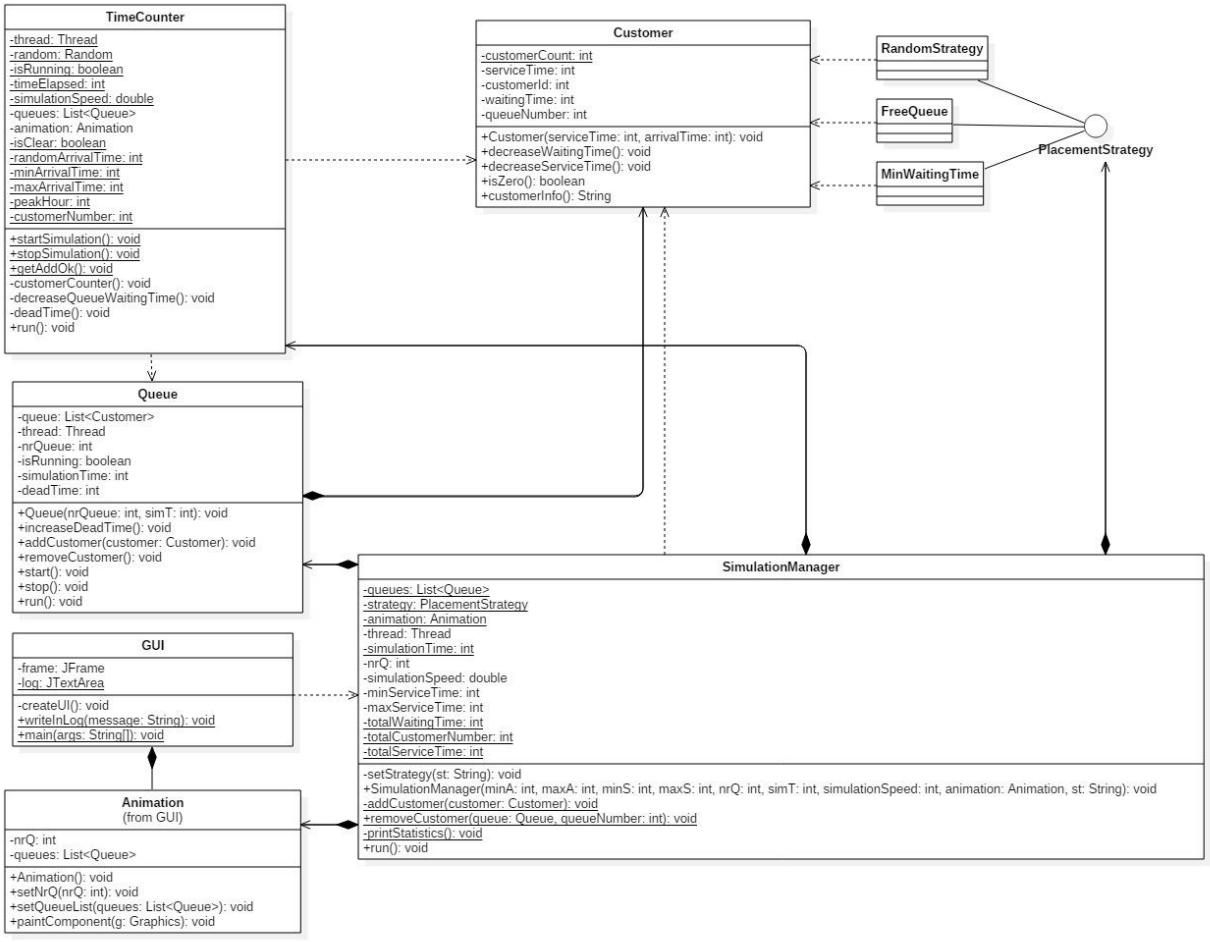
2. Analiza problemei, asumptii, modelare, scenarii, cazuri de utilizare, erori

Acest program care se ocupa de management de cozi trebuie sa reproduca functionarea unor case de marcat cat mai fidel, la o scara de timp cat mai mica. Astfel simularea trebuie sa functioneze cat mai bine pe un set de date de intrare cat mai mare. Fiecare coada va functiona pe principiul „primul venit, primul servit” si va primi clienti cu un anumit timp de servire si un anumit timp de asteptare pana vor parasi coada. Astfel problema principala va contine elementele coada, client si un manager de simulare.

Perioada de simulare va consta in timpul introdus de catre utilizator, astfel tot ceea ce ar trebui sa se intample dupa timpul limita de catre utilizator nu va mai conta.

3. Proiectare (decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfete, relatii, packages, algoritmi, interfata utilizator, modul de tratare a erorilor)

Diagrama UML



Avem clasa TimeCounter, clasa care se va ocupa cu tot ceea ce e legat de timp in aceasta simulare: decrementarea timpului de asteptare la coada a clientiilor, generarea de clienti la momente de timp aleatoare, calculul timpului mort al caselor dar si timpului mediu de asteptare la coada si servire.

Clasa Queue va implementa concret conceptul de coada simuland intrarea si iesirea clientilor din aceasta. Va fi capabila sa introducere si scoatere a elementelor de tip Customer dintr-o lista.

Clasa Customer va reprezenta clientul, care va contine timpul de asteptare si servire, id-ul acestuia care este unic, dar si casa la care a fost asezat.

Interfata PlacementStrategy care este implementata de alte 3 clase, contine metoda care va genera casa la care clientii vor fi plasati in functie de strategia aleasa de catre utilizator.

Clasa SimulationManager va reprezenta initializatorul simularii. In aceasta clasa se creaza coziile, se adauga sau sterg clienti din cozi, se pornesc firele de lucru pentru toate elementele care necesita aceste elemente si asigura distribuirea tuturor datelor introduse de catre utilizator claselor care vor utiliza aceste date.

Clasa GUI este clasa care face legatura utilizatorului cu simularea, aceasta clasa va primii datele necesare simularii dar va si afisa evenimentele produse in timpul simularii cat si o reprezentare grafica a acestei simulari.

4. Implementare

Clasa GUI va construi o fereastra compusa din 3 campuri mari: campul de input care se ocupa cu preluarea datelor din campuri si transmiterea lor mai departe, campul de reprezentare grafica care se ocupa de afisarea simularii pe ecran pentru a putea fi vizualizata, lucru realizat de o clasa Animation care foloseste un element de tip Graphics. Al 3-lea camp din fereastra este reprezentat de catre textul de log, un TextArea pe care sunt afisate evenimentele importante din simulare, cum ar fi sosirea sau plecarea unul client, dar si statistica de la finalul simularii.

Clasa Queue va reprezenta o coada propriu-zisa prin intermediul unei liste, fiind capabila de stergerea sau inserarea unui element nou, element care este reprezentat de Customer. In aceasta clasa vom folosi un fir de lucru, astfel fiecare obiect de tipul Queue va rula simultan, creandu-se cat mai fidel reprezentarea din viata reala a unei case de marcat. In metoda run() implementata in aceasta clasa se verifica daca timpul de servire al unui client este sau nu 0, in caz afirmativ se

sterge elementul din capul listei si este apelata metoda removeCustomer() din clasa SimulationManager, metoda care va apela la randul ei o metoda din clasa Queue care va sterge fizic elementul din lista.

Clasa Customer reprezinta clientul simularii, identificat printr-un ID unic generat la crearea unui obiect de tipul Customer, astfel se asigura unicitatea ID-ului. Clientul va mai avea campurile serviceTime care reprezinta timpul pe care acesta il va petrece in capul cozii pana cand va fi scos din aceasta, timpul de sosire, cand urmatorul client va fi generat dupa el si waitingTime care va reprezenta timpul pe care acesta il va petrece la coada pana cand o va parasi.

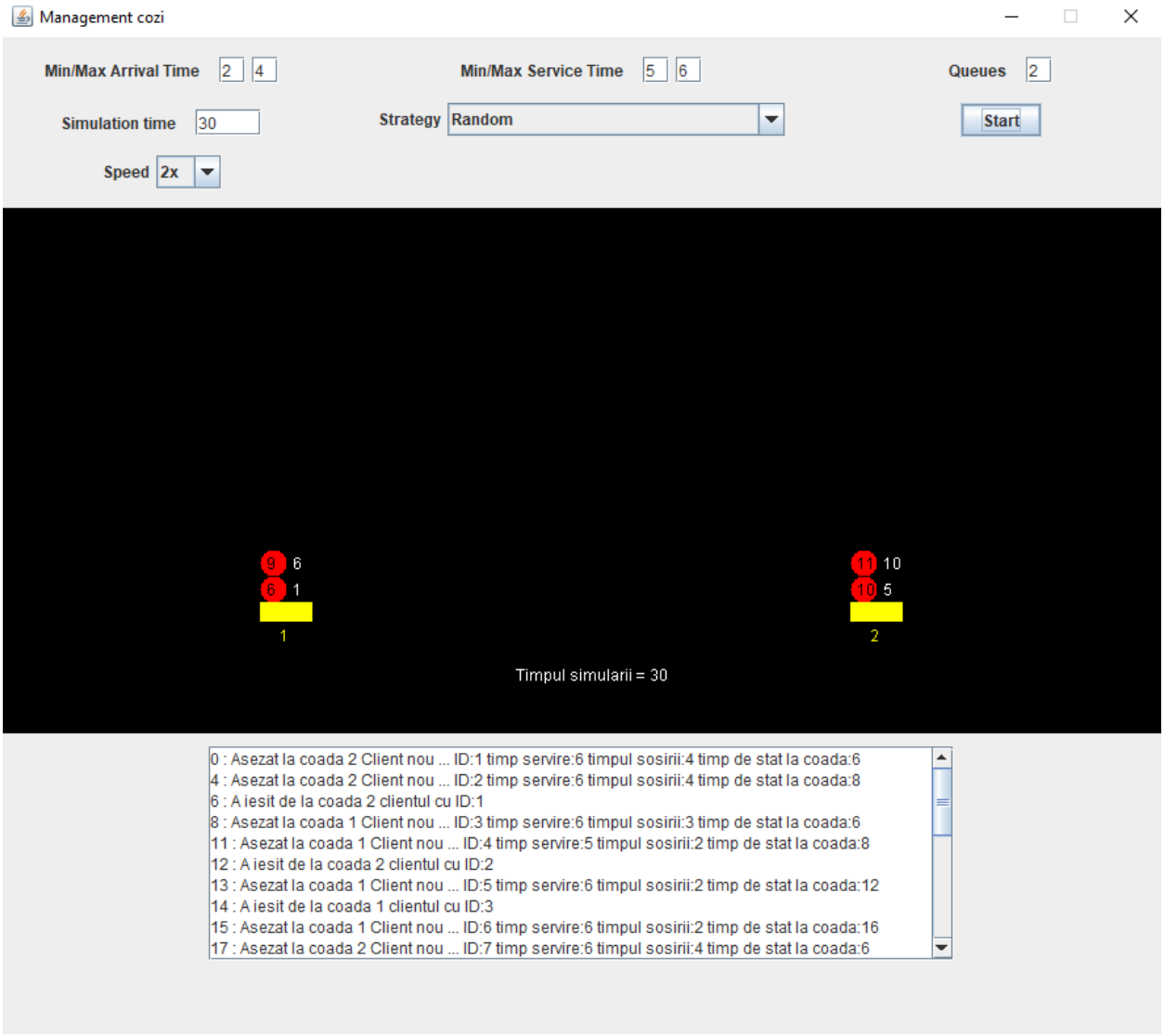
Clasa TimeCounter se ocupa cu sincronizarea procesului de simulare. Cu ajutorul acestei clase vom stii mereu in ce moment al simularii ne aflam si va face mai usoara urmarirea parcursului simularii. In aceasta clasa este implementat un fir de lucru, iar in metoda run este descrisa functionarea simularii, dupa fiecare unitate de timp care trece este decrementat timpul de asteptare la coada pentru fiecare client, este calculat timpul total de asteptare la coada dar si timpul total de servire care va fi utilizat la calculul statisticilor de la finalul simularii. Tot in aceasta metoda este procesat timpul pana cand va fi generat un nou client, valoare generata in intervalul introdus de catre utilizator in cadrul interfetei.

Interfata PlacementStrategy este implementata de catre 3 clase: RandomStrategy care va genera un numar aleatoar intre 0 si numarul de case introdus de catre utilizator, cu ajutorul acestui numar clientul nou generat va fi plasat la casa corespunzatoare. FreeQueue va genera un numar aleator intre 0 si numarul de case bazat pe numarul de clienti de la fiecare casa si va returna indexul casei care are cei mai putini clienti asezati la coada. MinWaitingTime va genera un numar aleator intre 0 si numarul de case bazat pe timpul de asteptare de la fiecare casa, calculat parcurgand toti clientii unei case, si va returna indexul casei care are timpul de asteptare cel mai mic.

SimulationManager se ocupa cu initializarea tuturor componentelor simularii, trimitand toti parametrii introdusi de catre utilizator claselor care-i vor utiliza. Aceasta clasa are un fir de executie care se ocupa cu generarea timpului

pana cand un nou client va fi generat. Cand este momentul adaugarii unui nou client la o coada noua va fi apelata metoda addCustomer, metoda care se ocupa cu apelarea metodei care genereaza coada la care va fi plasat in functie de strategia aleasa de catre utilizator, de apelarea metodelor de generare unui client nou dar si scrierea evenimentului in campul de log. La fel functioneaza si si metoda removeCustomer care se ocupa cu stergerea unui client de la o coada anume dar si scrierea evenimetnului de stergere intr-un log. In metoda run se genereaza timpul nou de sosire a unui client, cand acest timp ajunge la 0 se apeleaza metoda de adaugare si se genereaza un nou timp. Metoda run se executa atata timp cat durata simularii este mai mica decat timpul masurat de la inceperea simularii pana intr-un anumit punct. Cand acest lucru devine fals, toate firele de executie create la inceputul simularii sunt oprite si se opreste simularea.

5. Testare/Rezultate



Pentru testare vom rula programul propriu-zis. In imaginea de mai sus, in partea de sus a ferestrei se observa campurile de parametrii pe care utilizatorul le va introduce. Min/Max arrival time reprezinta timpul minim si maxim intre care un client nou va fi generat, Min/Max service time reprezinta timpul minim si maxim intre care clientul nou generat va fi servit, Queues reprezinta numarul de cozi pe care simularea il va utiliza, Simulation time reprezinta durata simularii in

unitatii de timp, Strategy permite selectarea strategia dupa care clientii noi generati vor fi plasati la coada, Speed permite utilizatorului de a selecta viteza cu care simularea va decurge, optiune care permite rularea simularii sa ruleze mai repede, fapt care duce la generarea rezultatelor mai repede. Partea din mijloc a ferestrei este ocupata de catre partea grafica care permite o animatie interactiva a simularii. Bulinele rosii reprezinta clientii, numarul din interiorul acestora reprezinta id-ul clientului, lucru care ne permite identificarea lor mai usoara. Numarul aflat in dreapta bulinelor reprezinta timpul dupa care bulina in dreptul careia se afla numarul va parasi coada. Dreptunghiul galben reprezinta o casa de marcat, iar numarul care se afla dedesuptul dreptunghiului reprezinta indentificatorul casei. Sub aceste elemente se afla timpul simularii, lucru care ne permite identificarea momentului de timp in care ne aflam in cadrul simularii. Ultimul element al acestei ferestre ne permite identificarea momentelor cheie ale simularii. Pentru fiecare linie, primul numar reprezinta momentul la care s-a petrecut evenimentul, dupa care este urmat ori de evenimentul de introducere sau scoatere a clientiilor de la coada. In cazul in care se face o asezare la coada a unui client, se va specifica numarul cozii la care acesta va fi asezat, ID-ul acestuia, timpul de servire dar si timpul cat acesta va sta la coada. In cazul scoaterii unui client de la coada se specifica timpul la care acesta este scos, coada de la care este scos si ID-ul clientului care este scos. Toate aceste date ne permit sa identificam mai bine evenimentele petrecute. La finalul simularii este afisata statistica calculata pe durata simularii, si anume: ora la care au fost cei mai multi clientii per total, timpul mediu de asteptare la coada a clientilor, timpul mediu de servire al clientiilor si timpul pe care fiecare casa l-a petrecut goala, adica fara un client care sa fie servit.

6. Concluzii

Aceasta aplicatie este foarte utila pentru o firma care detine o afacere care implica clienti si cozi. Avand o statistica probabila continand timpul la care un client ajunge, timpul in care acesta a fost servit dar si numarul de cozi se poate realiza o simulare apropriata de realitate si o posibila corectare a managementului afacerii in caz ca rezultatele simularii nu corespund cu cele asteptate. Dezvoltarile ulterioare care pot fi aduse aplicatiei pot fi optimizarea si simplificarea codului, o sincronizare mai buna a firelor de lucru utilizate pentru a nu exista desincronizari sau curse care vor duce la rezultate eronare, crearea unei

interfete mai interactive si mai sugestiva, dar si a mai multor optiuni de parametrii care pot fi introdusi pentru a realiza o simulare cat mai apropriate de nevoie utilizatorul. Din aceasta tema am invatat sa utilizez si sa aprofundez libraria care se ocupa cu grafica desenand niste primitive si actualizarea acesteia in timp real pentru a corespunde cu ceea ce se intampla pe parcursul executiei programului, utilizarea firelor de lucru dar si sincronizarea acestora pentru o viteza crescuta de executie al programului, dar si conceptele de baza necesare dezvoltarii unei astfel de aplicatii.