Descriere

Proiectul realizat are la baza cerintata de a simula si analiza un magazin in care clienti vin, asteapta la coada, sunt serviti si parasesc coada. Proiectul presupune folosirea de threaduri si timere pentru monitorizarea timpilor de sosire la coada, de asteptare, servire si timpul de parasire a magazinului. Pentru simulare e folosita o fereastra grafica JFrame care contine doua panouri: unul pentru datele de intrare, adica numarul de cozi, perioada de simulare, timpul minim si maxim de servire , timpul minim si maxim dintre clientii care vin pentru a fi serviti; si unul pentru desenarea efectiva a cozilor si a clientilor. Clienti sunt reprezentati printr-un patrat de culoare portocalie si id-ul fiecaruia scris in interiorul patratului. Casele, cinci la numar, sunt dreptunghiuri de culoare albastra dupa care sunt aliniati clientii si sunt deschise in functie de numarul specificat inainte de pornirea simularii. In sine simularea dureaza in functie de intervalul de simulare dat si daca dupa secunda specificata mai sunt clienti in coada acestia vor fi tratati si casele se vor inchide.

Tema contine 15 entitati java si urmeaza tiparul Model View Controller cu specificatia ca divizarea atributilor s-a facut in functie de modelul ales. Thread-urile sunt distribuite pe fiecare casa si exista un thread “mare” care comanda simularea, toate fiind sincronizate dupa un timer global afisat pe fereastra. Totul a fost creat folosinduse mediul InteliJ Idea (2017.1) , varianta java jdk1.8.0_112 si librarii precum “javax.swing”, “java.awt” si “java.util”.

Clase Componente

Client

“Client” este clasa ce retine datele referitoare la un client: idClient, arrivalTime, serviceTime, finishTime, waitingTime, served. Ea detine doi constructori si metode de tipul “get” si “set” pentru obtinerea variabilelor private declarate in clasa. Clasa implementeaza interfata Comparable si suprascrie metoda compareTo in functie de timpul de sosire, si va fi folosita ulterior pentru sortarea liste de clienti generata aleator.

Singeorzan Florin
CTI seria B gr. 30227

HomeStore

Clasa detine o coada de clienti aflati intr-un LinkedBlockingDeque, un int care reprezinta numarul casei si o variabila booleana care are semnificatia de a indica daca casa este deschisa sau nu. Ea are un singur constructor in care coada este initializata si noua metode printre care metoda de run(), cea responsabila pentru comportarea thread-urilor. Metoda de adaugare in coada pune un client in capatul cozii si afla timpul care il are de asteptat prin findWaitingTime, in functie de timpul in care a ajuns si timpii de servire a celorlalti clienti din fata lui. HomeStore descrie metodele mustServe si nextServed, care vor fi folosite doar in interiorul clasei de catre metoda run. Aceasta metoda, atat timp cat casa este deschisa, pentru fiecare client seteaza timpul de terminare, faptul ca a fost servit si un logOut obiect de tipul clasei Log care scrie intr-un fisier “logging.txt” date despre clientul procesat.

Scheduler

Aceasta clasa contine partea “hard” a modelului, fiind entitatea ce va reprezenta logic intreg magazinul. Ea detine cate o lista de case (HomeStore) si o lista de threaduri, precum si stategia de adaugare (de alegere pentru clienti casa la care se vor plasa). Constructorul ei initializeaza listele si atribuie fiecarei din cele 5 case disponibile un thread propriu. Clasa are o metoda standar de schimbare a strategiei de adaugare si gettere si settere pentru obiectele de tipul listelor. Metoda dispachClient adauga un client in multimea de case, la o casa in functie de o strategie aleasa.

Strategy

Interfata contine o singura metoda, cea de adaugare si prin enumeratia SelectionPolicy formeaza doua strategii de adaugare: una care presupune ca un client nou venit se ca plasa la casa cu cei mai putini clienti si una care are ca scop calcularea timpului de asteptare (in functie de cel de servire a clientior deja aflati la coada) si plasarea in cea care ar presupune un timp cat mai mic de asteptare. ConcreteStrategyQueue si ConcreteStrategyTime sunt clasele ce implementeaza interfata descrind modul in care se face adaugarea. Prima clasa parcurge multimea de clase, retine index-ul clasei deschisa cu cei mai putini clienti in coada,

seteaza numarul case (index-ul) si adauga clientul in coada prin folosirea metodei de adaugare descrisa in clasa HomeStore. Cealalta clasa, se deosebeste prin faptul ca ea calculeaza pentru fiecare coada in parte timpul de asteptare si adauga clientul in coada care l-ar avantaja din vederea timului petrecut pentru a fi servit. Pe partea de simulare s-a folosit metoda de adaugare din cea de a doua clasa deoarece ea minimizeaza cel mai eficient waitingTime-ul pentru fiecare client.

TimerSec

Aceasta clasa detine un int secondPassed initializat cu 0 si prin metoda run din TimerTask incrementeaza aceasta variabila. Incrementarea se face odata la o secunda astfel incat clasa implemeteaza un timer care incepe din secunda 1 si monitorizaeaza secundele trecute de la apelul metodei start().

Log

Clasa detine doua obiecte, un logger si un fileHandeler. In constructor, daca nu exista un fisier creat, creaza unnul (specific logging.txt) si adauga logger-ului obiectu fileHendeler si un obiect de tipul SimpleFormatter.

Customer

Clasa retine date referitoare la un client, de asisat pe fereastra grafica. Obiectele detinute sunt: un client, pozitia acestuia pe axa x si pe axa y in fereastra, forma lui (Rectangle), casa la care clientul a ales sa astepte si un speficicator care transmite daca clientul a fost servit sau nu (initializat cu false). Clasa detine metode de obtinere a obiectelor private si un constructor fara parametri care care instantiaza Rectangle-ul “shape” si clientul in cauza.

MenuPanel

Extinzand clasa JPanel, clasa devine un panou cu atribute specifie care vor fi desenate in fereastra principala. Ca obiecte, sunt folosite un boton, un combo box, cinci zone pentru text si un tablou de sase label-uri. In constructor sunt setete dimensiunea panoului, culoarea si sunt apelate trei metode private din aceasta clasa:

- Init() – metoda instantiaza containarele create, adauga componente bombo box-ului, seteaza culoarea de fundal pentru fiecare ca fiind WHITE.darker(), si adauga textul specific pentru fiecare label din cele sase. Ea apeleaza urmatoarea metoda;
- place() – are atributia de a plasa efectiv (prin metoda addBounds) fiecarui container locul predestinat in panou, astfel ca butonul apare primul (ordine up - down), dupa JComboBox-ul iar apoi zonele predestinate datelor de intrare, fiecare avand putin deasupra lor label-ul corespunzator care specifica pentru ce sunt datele care trebuie introduse;
- addContainers() – seteaza layout-ul ferestrei la null si adauga fiecare container in panou this.add(< container >)
- action() – aceasta metoda specifica atributile pe care trebuie sa le aiba butonul si combo box-ul. Lor li se adauga un nou ActionListener si in interiorul parantezelor acestuia se suprascrie metoda actionPerformed. Butonul, in fereastra grafica seteaza starea ferestrei (timer) la true, prin metoda getText() ia datele introduse (timpii de simulare, si numarul cozilor de deschis) si seteaza in clasa de timpul JFrame-ul obiectele aferente. Dupa introducerea datelor in fiecare container i se transmite prin setEditable(false) faptul ca in timpul de rulare nu se pot modifica datele de intrare, lucru care, nefiind tratat poate duce la comportarea incorecta a simularii. In cazul in care datele nu au fost introduse corect se va transmite pentru a putea fi afisat un mesaj pentru reintroducerea input-urilor. Listener-ul atasat combo box-ului actioneaza prin transmitere stringului numeric ce reprezinta numarul de cozi, ferestrei grafice.

Renderer

Aceasta clasa este celalalt panou adaugat ferestrei. Are o singura metoda “paintComponent” care seteaza dimensiunea ferestrei, seteaza layout-ul la null. apeleaza metoda paintComponet din parinte (super) si metoda repaint(din fereastra grafica). Metoda de desenare din parinte are rolul de a desena o scena de obiecte, dupa care prin metode dispose() le inlatura.

SimulationFrame

Clasa implementeaza interfata actionListener. Ea ara o lista de Consumer (cei ce urmeaza a fi desenati), variablie care retin datele de intrare, obictul ferestrei in sine, obiecte de tipul celor doua panouri, contante pentru dimensiunea ferestrei, doua variabile booleene ce specifica daca fereastra trebuie sa simuleze si daca datele sunt corecte si doua timere, unul standard si unul de tipul clasei TimerSec. In constructorul acestei clase sunt create obiectele de tipul celor declarate, sunt adaugate caracteristici ferestei: vizibilitate, titlu, dimensiune, sunt adaugate cele doua panouri ferestei, se seteaza starea simularii la false (clientii nu vin la cozi) si timer-ul pornit prin metoda start(). Acest timer are doi parametri: 500 si this; primul reprezinta un numar de milisecunde si specifica la cat timp este apelat (automat) un Listener, iar “this” indica faptul ca acel listener este clasa in sine. Metode:

- paintHomeStores printr-un switch verifica numarul de case pe care metoda trebuie sa le deseneze. Fiecare din cele cinci case sunt dreptunghiuri de culoarea albastra si sunt plasate la baza ferestrei grafice in pozitie orizontala;
- Metoda detX are rolul de a determina pozitia pe axa Ox pentru fiecare client in parte. Folosind tot un switch se determina remura care reprezinta pozitia la casa aleasa si in functie de axa casei se seteaza prin setxPosition pozitia aferenta fiecarui client;
- detY() calculeaza pozitia pe axa Oy dintre clienti. Porneste de la o pozitie initiala si scade acea pozitie cu spatiul necesar incadrarii unui alt client la coada. Atunci cand coada se schimba se replaseaza primul client la pozitia initiala;
- Metoda responsabila cu adaugarea formei addForm(), parcurge multimea de clienti si seteaza caracteristicile rectangle-ului “shape” la pozitile din clasa si dimensiunea 30 x 30;
- PaintClients seteaza pentru un client culoarea portocalie, si umple un patrat dupa atributele figurei ”shape” din clasa Consumer. In interiorul acestui patrat este desenat un string ce reprezinta id-ul clientului

- actionPerformed este metoda ce se va executa incontinu pe parcursul simularii. Initial verifica daca fereastra trebuie sa simuleze si daca da, atunci timerul de tipul TimerSec este pornit prin metoda start(). Pentru fiecare client se verifica daca este timpul ca el sa asjunga la coada si seteaza la true variabila mustPaint (din Consumer) daca acesta trebuie desanat. Pe urma se verifica daca exista clienti care si-au terminat timpul de servire si ii elimina pe acestia din lista de clienti din magazin. Se apeleaza metodele de determinare a pozitiei fiecarui client, se seteaza forma si in cazul in care totul este in regula se apeleaza metoda repaint() fara parametri pe obiectul de tipul clasei panoului de animatie.
- Metoda “repaint” primeste un obiect grafic si pe el deseneaza tot ce trebuie pus in panou. Seteaza un rectangele alb pe toata fereastra grafica, atata timp cat simularea ruleaza un string ce reprezinta secundele scurse de la inceput este pictat. In cazul in care datele sunt invalide se printeaza un “Invalid data set” pe fereastra si daca exista clienti care trebuie pictati prin metoda paintClients acestia sunt generati;
- Clasa mai contine metode pentru preluarea si setarea obiectelor private.

SimulationManager

Clasa implementeaza interfata Runable si suprascrie implicit metoda run(). Rolul major al acestei clase este cel de Controller, ea find cea care distribuie sarcinile celorlalte clase din proiect. Are un obiect de tip Random din Java.util care este folosit pentru generarea aleatoare de clienti, un obiect scheduler un altul public static de tipul ferestrei, o lista unde vor fi plasati clientii generati aleator si un int static pentru modificarea id-ului de la client la client. Constructorul creaza obiecte de tipul Scheduler si SimulationFrame. Metoda setEffectiv parcurge fiecare clasa deschisa si creaza un nou obiect Consumer pe care il adauga in fereastra pentru a semnala existenta lui in magazin. StartThreads are obligatia de a porni thredurile (metoda start()) daca nu sunt in viata isAlive, daca casa e deschisa si daca parametrul ce semnaleaza validitatea datelor de intrare e setat pe false (de la intrare nu au fost introduse date valide). Un bloc try-catch inglobalizeaza metoda start deoarece aceasta poate arunca o exceptie neprevazuta. Metoda openHomeStore parcurge multimea de case din

magazin si le seteaza drept deschise (doar atatea cate au fost introduse). Functia optima a acestei clase este generateClients, functie care are rolul de a genera clienti cu date aleatoare in functie de parametri descrisi la intrare. Metoda trateaza cazul in care simularea poate ajunge intr-o stare invalida/nedorita astfel: daca timpul minim dintre clienti e mai mare decat timpul maxim, daca timpul minim de servire e mai mare decat timpul maxim su daca timpul minim dintre clienti e setat la zero (pentru 0 s-ar genera toti clientii odata). In metoda sunt generati un numar de 100 de clienti. Daca starea ferestrei este una invalida atunci pana cand va deveni valida este blocata intr-o bucla goala {}. Se calculeaza timpul de servire dupa metoda forService = frame.getMinServiceTime() + random.nextInt(frame.getMaxServiceTime() - frame.getMinServiceTime()) , cel de sosire: inInterval = random.nextInt(frame.getMaxInterval() - frame.getMinInterval()) + frame.getMinInterval() | arrival = lastArrival + inInterval | lastArrival = arrival. Id e crescut din unu in unu la fiecare client daca timpul de sosire este mai mare ca si timpul pe care simularea trebuie sa il intretina minus timpul de servire al ultimului client atunci se va iesi din bucla while cu break. Datele create sunt asociate unui nou client care este adaugat listei de clienti.

Metoda waitRun “adoarme” threadul curent cate 500 mili secunde atata timp cat fereastra se afla intr-o stare de asteptare (timpul in care se introduc datele de intrare). Metoda run() este o metoda sincronizata care asteapta datele de intrare prin metoda waitRun, creaza lista de obiecte cu clienti generati, deschide casele si generaza cei 100 de clienti. Ca si strategie de selectie a cozi este setata metoda de adaugare prin minimizare a timpului de asteptare (SHORTEST_TIME). Atata timp cat fereastra ruleaza prin dispachClients clientii generati, dupa ce au fost sortati dupa timpul de sosire sunt adaugati caselor pe care si le aleg. Clienti sunt scosi din lista generata si in blocul catch folosit pentru adugarea clientilor pana la limita null pointer exception sunt adaugati (setEffective) in lista consumatorilor ce urmeaza a fi desenati.Threadurile sunt pornite si in cazul in care timpul de simulare dat depaseste timpul actual al timer-ului ferestrei atunci metoda run este parasita.

Singeorzan Florin
CTI seria B gr. 30227

MainSimulation

Clasa detine metoda principala main de unde se simuleaza proiectul. Aceasta contine un obiect de tipul clasei SimulationManager si un Thread pe acest obiect care porneste la apelul metodei start().

Obseratii

La introducerea datelor de intrare panoul de clasa Renderer pare a se etinde pe toata fereasta, fapt datorat metodei repaint() apelata pe obiect, cere la randul ei apeleaza repaint(0, 0, 0, width, height) care specifica faptul ca dimeniune ferestrei incepe de la (0, 0), coltul din stanga sus. Threadurile desi au o ordine aleatoare, manifestarea lor depinde intr-o oarecare masura de durata pe care threadul o petrece in starea de stagnare.

Links

- http://stackoverflow.com/
- http://coned.utcluj.ro/~marcel99/PT/
- https://thenewboston.com/videos.php

Singeorzan Florin
CTI seria B gr. 30227