**Tema 3 OrderManagement**

**Muntean Andrei**

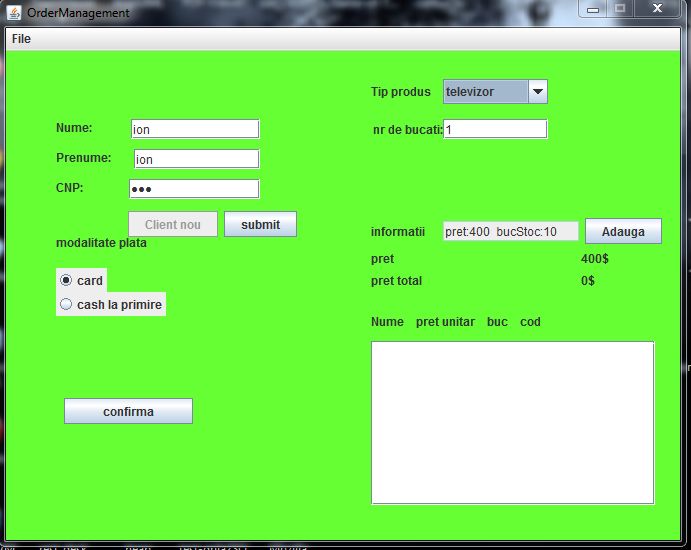
**Gr30227**

Descriere Obiectuala

Aplicatia curenta foloseste in total opt clase: Warehouse, Chitanta, Customer, Order, AllOrders, Product, Fatada si ClickListener.Trebuie mentionat faptul ca aceste clase sunt incorporate in pachetul **OrderManagement**. Fiecare clasa va fi detaliata mai jos dupa cum urmeaza:

**Clasa Fatada**

Oricand ne uitam la o cladire prima data se vede fatada acesteia. Este usor de dedus ca aceasta clasa realizeaza interfata grafica a aplicatiei. Pentru aceasta s-a folosit grafica oferita de javax.swing.



Dupa cum se observa interfata grafica(a nu se confunda cu interfata java) are patru butoane simple, doua butoane radio, un Text Area, doua campuri de text, un combo box si mai multe etichete(labeluri). Pentru o elegant sporita s-a optat si pentru realizarea unui meniu simplu. Meniul FILE permite incheierea aplicatiei si iesirea in sistemul de operare prin apasarea popup meniului exit sau short cut keyul acestuia Alt+F1.

Este o interfata simpla, care ofera o eficienta maxima fara a supraincarca vizual utilizatorul. Pentru a pastra aceasta neutralitate vizuala s-a optat pentru fundal o culoare rece (albastru deschis).

Din interfata grafica se poate concluziona cum functioneaza de fapt aplicatia curenta.

Pentru a putea utiliza programul prima data trebuie sa se introduca un client. Utilizatorul isi completeaza datele in campurile de text corespunzatoare, iar cnp-ul este folosit ca si parola dat fiind ca acesta este un identificator unic pentru persoanele din Romania.

Pentru confirmarea datelor se apasa tasta ”submit”.

Apoi se poate alege un produs si numarul de bucati care se doreste a fi cumparat. In campul informatii se gasesc informatii vitale despre produsul selectat: pretul pe bucata si numarul de unitati disponibile in magazine in acel moment. Apoi sub acest camp sunt afisate doua preturi: primul pret este pentru produsul current iar sub el se afla pretul total al comenzii.

Jos in dreapta se gaseste ceva ce am denumi “cosul de cumparaturi”. Fiecare tranzactie se confirma prin apasarea butonului ADAUGA, lucru care duce la aparitia in cosul de cumparaturi a produsului solicitat de catre comparator.

Dupa ce clientul si-a terminat cumparaturile poate trece la confirmarea cumparaturilor si se trece la creerea unei chitante electronice in format TXT dupa urmatorul model:

Numar chitanta:

Informatii client

Nume:

Prenume:

CNP:

Produse cumparate:

nume pret buc cod

total:

Semnatura:

Toate acestea se realizeaza prin simpla apasare a butonului confirma.

Daca se doreste o noua tranzactie se apasa butonul client nou si toate informatiile vechi se vor reseta.

Sa discutam un pic si despre metodele clasei *Fatada*.

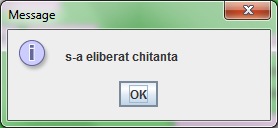
In general clasa aceasta are multe metode de tip get si set pentru configurarea si reconfigurarea spatiului visual al interfetei grafice(ex getPretTotal). Unele dintre aceste metode au fost declarate statice pentru a se elimina constructia unui nou obiect de tip Fatada.

Clasa Fatada mai are si trei clase interioare, toate cele trei clase implementand interfata Action Listener din java.awt.event. Prima clasa interioara ofera Action Listener pentru combobox, al doilea pentru meniul exit si al treilea pentru campul de text in care se introduce numarul de bucati cumparate. S-a optat pentru clase interioare deoarece aceste clase nu-si au rostul de a fi declarate separate deoarece nu sunt folosite de nici o alta clasa in afara de clasa Fatada.

Pentru a realiza actiunile declansate de butoane s-a optat pentru o clasa exterioara separate, clasa ClickListener.

**Clasa ClickListener**

Clasa ClickListener este de fapt clasa principal a aplicatiei deoarece fara aceasta aplicatia nu poate sa existe fizic. Aici prima data se verifica care buton a fost apasat, iar mai apoi se decide ce urmari va avea declansarea butonului. Tot aici se calculeaza pretul total al cosului de cumparaturi, informatie care este predate ulterior clasei Fatada pentru afisarea acesteia. Tot in clasa ClickListener se construieste si comanda propriu-zisa. Un client poate face mai multe comenzi, asa ca s-a apelat la un vector de comenzi, care are ca elemente comenzi. Dupa ce s-a efectuat trtanzactia comanda curenta este adaugata in arhiva(in AllOrders). Clasa aceasta mai are doua campuri esentiale: campul client si chitanta. Obiectul client este creat la apasarea butonului “submit” iar chitanta este create la momentul confirmarii procesului de vanzare. Tot la finalizarea tranzactiei va aparea si un mesaj, care ne instiinteaza ca s-a eliberat chitanta.



Mai departe vom discuta despre cel mai important element al consumului: clientul.

**Clasa Client(sau Customer in engleza)**

Dupa cum se poate observa din explicatiile de mai sus clasa Client contine trei campuri estentiale si anume: nume, prenume si CNP. Clasa Client contine si trei metode de get pentru a respecta principiul incapsularii: getNume, getPrenume si getCNP.

Daca am discutat despre client in cele ce urmeaza se va trece in revista clasa urmatoare ca importanta in societatea consumerista si anume: Produsul

**Clasa Produs(Product in engleza)**

Clasa Product este construita identic cu clasa Client. Are trei campuri: nume produs, pretul acestuia si codul unic de recunoastere. Contine si toate cele trei metode de get: getNumeProdus, getPret si getCod dar mai contine o metoda toString care este folosita pentru pretty printing. Aceasta metoda returneaza un string care contine de fapt toate informatiile clasei adunate la un loc.

Daca am discutat despre produs trebuie sa aruncam o privire si asupra locului unde sunt depozitate produsele si anume in deposit(eng Warehouse).

**Clasa Warehouse**

Clasa Warehouse este cea mai complexa clasa din pachetul OrderManagement, asa ca fiecare clasa va fi detaliata in parte.

1. *Clasa Stoc*
2. *Clasa StockComparator*
3. *Warehouse propriu-zisa*

A: Clasa Warehouse propriu-zisa

In cerinta se specifica faptul ca, clasele AllOrder si Warehouse trebuie sa implementeze un arbore binar de cautare. Asa ca dupa o documentare serioasa s-a optat pentru implementarea clasei predefinite de catre java Sun si anume clasa TreeSet. Dar sa spunem niste cuvinte despre iteratori, generice

La gestiunea seriilor, seturilor si a grupurilor de date, tablourile nu sunt întotdeauna cea mai bună solutie deoarece tablourile nu excelează atunci când datele sunt volatile – în special atunci când mărimea setului de date poate fluctua.Inserarea unui element necesită glisarea elementelor de deasupra punctului de insertie si e nevoie de spatiu suplimentar alocat la sfârsit.

Colectiile sunt obiecte cu metode care trebuie invocate pentru a citi sau scrie elemente.

Colectie Java: orice clasă care păstrează obiecte si implementează interfata **Collection.**

Interfata **SortedSet<T>** si clasa concretă **TreeSet<T>** sunt destinate să implementeze

interfata **Set<T>** si să ofere regăsirea rapidă a elementelor

Implementarea clasei este asemănătoare cu un arbore binar, dar inserarea păstrează echilibrul arborelui.

Clasa Warehouse contruieste un arbore de obiecte de tipul stoc. Stocul este compus dintr-un produs si numarul de bucati asociat acestuia. Clasa Stoc a fost construita ca o clasa interioara.

B: Clasa Stoc

Cum am scris mai inante constuctorul acestei clase primeste doi parametrii: primul este de tipul produs iar al doilea este un int in care se specifica numarul de unitati disponinibile la inceput. Clasa interioara are metode care returneaza produsul pus in stoc precum si numarul acestuia si permite facerea de modificari in cantitatea unui produs prin metoda setNrUnitati. O alta metoda importanta este metoda vanzare care permite sa se faca operatia de vanzare (se decrementeaza continutul campului nrUniti).

Pentru a putea utiliza un arbore binar de Obiecte Stoc a fost necesara construirea unui comparator. Deoarece TreeSet pune intr-o anumita ordine elementele, daca programatorul nu specifica pe ce se face comparatia , ea se face pe ordinea naturala a elementelor. Cum nu este ordine naturala a obiectelor de tip Stoc s-a construit o noua clasa care implementeaza interfata Comparator. Clasa aceasta se numeste *StockComparator.*

C: Clasa StockComparator

Clasa aceasta suprascrie o singura metoda din interfata si anume metoda compare. Aceasta metoda primeste ca parametrii doua Obiecte. Pe aceste doua obiecte se face mai apoi un cast spre tipul Stoc si se seteaza ca comparatia a doua elemente de tip Stoc sa se faca in ordinea alfabetica a numelui produselor continute.

public class StockComparator implements Comparator {

public int compare(Object a, Object b) {

String A = ((Stoc) a).getProduct().getNumeProdus();

String B = ((Stoc) b).getProduct().getNumeProdus();

return A.compareTo(B);

}

}

Dup ace am discutat despre constructia clasei Warehouse trebuie sa amintim si care este rolul acestei mari clase. Clasa Warehouse are o metoda care citeste dintr-un fisier datele despre produse si le transforma in obiecte de tip stoc. Sa ne reamintim cum arata un obiect Stoc: Produs(nume produs,pret unitar, cod unic de identificare ) si numar de bucati. In fisierul de produse pe fiecare linie sunt scrise detalii despre produs si acestea sunt despartite prin spatiu deoarece algoritmul de constructive delimiteaza obiectele dupa spatiu . Fisierul are de exemplu urmatorul continut:

televizor 400 12 10

calculator 800 13 10

laptop 2000 14 10

monitor 500 15 10

imprimanta 400 16 10

scanner 500 17 10

memorii 200 18 10

stick 40 19 10

Pentru clasa Fatada clasa Warehouse poate trimite un String , care contine numele tuturor produselor. Acest String se foloseste la constructia Combo boxului di interfata grafica ex televizor,calculator,laptop.

Pentru a permite o mai buna cautare si parcurgere a Tree Setului de Stoc se adauga acestuia un iterator.

Sa nu uitam faptul ca aceasa clasa mai are si o metoda de cautare a Stocului.

Acum vom trece mai departe si vom vorbi despre o alta clasa care face legatura intre client si produs si anume clasa Comanda(eng Order)

**Clasa Order**

Clasa Order este contruita asemenea Clientului si Produsului. Are trei campuri principale . In aceasa clasa se face legatura dintre Client si produs asa ca constructorul acestei clase primeste ca si parametrii un obiect de tip Client, un alt obiect de tip Produs si un int care simbolizeza numarul de bucati comandate. In afara de metodele get traditionale clasa mai contine o metoda de tip String care returneaza intr-un singur String toate elementele vitale ale unei comenzi.

Toate comenzile facute client sunt memorate in istoric(Clasa AllOrder). Nu trebuie uitat insa ca un client poate sa faca mai multe comenzi asa ca in clasa de arhivare se vor memora de fapt vectori de comenzi.

**Clasa AllOrder**

Ca si clasa Warehouse si clasa AllOrder foloseste tot un arbore de cautare binar pentru memorarea datelor. Ca si in Warehouse si aici a fost nevoie de implementarea unei clase interioare care sa implementeze interfata Comparator Aici ordonarea elementelor se face in ordinea alfabetica a primului produs din Vectorul de comenzi. Si aici I se adauga un iterator TreeSetului in cauza pentru a permite o parcurcere si o cautare logaritmica.

Si in fine vom zice cateva cuvinte si despre clasa Chitanta

**Clasa Chitanta**

Clasa Chitanta nu are mare importanta in economia celorlalte clase, ea face doar scierea in fisier a informatiilor despre fiecare comanda in parte. Despre formatul chitantei am discutat chiar la inceput asa ca nu vom relua aici. Amintim doar faptul ca fiecare chitanta se emite pe numele clientului si de acea numele chitantei va fi identic cu numele clientului.

Daca am discutat despre clasele pachetului OrderManagement nu mputem incheia prezenta documentatie fara a detalia algoritmul de functionare.

**Algoritmul**

Algoritmul de functionare este unul relative simplu. Dar inainte sa discutam despre algoritm trebuie mentionat faptul ca o astfel de aplicatie se realizeaza cel mai bine pe un support fizic de baze de date, care au diverse relatii intre ele siau chei primare si straine. In aplicatia de fata nu s-au folosit baze de date traditionale ci s-a optat pentru un surogat, fisierul din care se extrag produsele.

Clasa Fatada nu este doar clasa pentru interfata grafica ci este in acelasi timp si clasa care uneste toate celelalte clase. Tot in clasa Fatada este si metoda statica main a aplicatiei.

Algoritmul funcitoneaza dupa urmatorul model: prima data se construieste clientul, care apoi are dreptul de comanda. Toate comenzile intra mai intai in cosul de cumparaturi unde pot fi vizualizate. Daca clientul nu este multumit de cum si-a ales produsele poate oricand opta pentru resetarea procesului prin apasarea tastei Client nou. Daca clientul este satisfacut se trecfe la selectarea modalitatii de plata. Plata cu card bancar este implicita. Daca se doreste plata la sosire se selecteaza aceasta. Odata selectate aceste lucruri se selecteaza butonul de confirmare.

Comanda clientului este construita prin apasarea butonului adauga iar acesta este finalizata prin apasarea butonului de confirmare. Tot la apasarea butonului de confirmare se transmite un mesaj clasei Warehouse in care se specifica care stoc trebuie decrementat si cu cat. Tot aici se construieste si chitanta in format electronic.

**Posibilitati de extindere**

Aplicatia curenta nu este complete. Se poate inbunatati atat modul de comandare cat si modul in care se realizeaza eliberarea chitantei. In lumea reala exista de cativa ani principiul comertului on line. Aplicatia curenta a vrut sa simuleze doar o mica parte din pricipiile care guverneaza aceasta lume a cumparaturilor on line. Cat despre modul de eliberare a chitantei ar fi utila sa se genereze si in format pdf sis a existe optiunea de tiparire a acesteia pe o imprimanta. Ar fi utila si o modalitate prin care acest document sa poata fi trimis si la o adresa de mail specificata de catre utilizator. Dar cum am zis mai- nainte o aplicatie de genul acesta se preteaza cel mai bine pe un support de baze de date gen sql .

Pentru o mai buna intelegere a programului s-a facut si o documentatie de tip javadoc.

**Testare**

Se introduce urmatoarele informatii:

Nume: Ion

Prenume: Pop

CNP:132435353

Se selecteaza produsele televizor una bucata si doua sticuri. Se apasa confirma. Mai jos apare rezultatul tranzactiei in fisier:

Numar chitanta:72491000

Informatii client

Nume:Ion

Prenume:Pop

CNP:132435353

Produse cumparate:

nume pret buc cod

televizor 400 1 12

stick 40 2 19

total:480

Semnatura:

**Diagrama UML**