Academia de Studii Economice București

Facultatea de Cibernetică, Statistică și Informatică Economică

Proiect

Calitate și testare software

**Profesor:** **Student:**

Dr.Nisioiu Codrin Pricope Sabina-Cristiana

An suplimentar

Cuprins

[Introducere 3](file:///E:\FACULTATE\CTS\proiectctsioana\Proiect-CTS.docx#_Toc419808630)

[Design patternuri utilizate și detalierea lor 3](file:///E:\FACULTATE\CTS\proiectctsioana\Proiect-CTS.docx#_Toc419808631)

[Definirea si detalierea metodelor testate prin Unit Testing 5](file:///E:\FACULTATE\CTS\proiectctsioana\Proiect-CTS.docx#_Toc419808632)

[Definirea si descrierea Test Case-urilor și Test Suite-urilori 7](file:///E:\FACULTATE\CTS\proiectctsioana\Proiect-CTS.docx#_Toc419808633)

[Descrierea sumara a functiilor aplicatiei cu referire la pattern-uri si la metodele testate 8](file:///E:\FACULTATE\CTS\proiectctsioana\Proiect-CTS.docx#_Toc419808634)

[Bibliografie 9](file:///E:\FACULTATE\CTS\proiectctsioana\Proiect-CTS.docx#_Toc419808635)

# Introducere

Am implementat proiectul in Java si am folosit mediul Eclipse. Aplicatia gestioneaza produsele unui supermarket.

În implementarea proiectului am folosit 3 design pattern-uri respectiv: Singletone, Memento si Observer.

# Design pattern-uri utilizate și detalierea lor

1.Singletone

Conceptul de singleton.reprezinta o tehnica prin care se limiteaza numarul de instante ale unei clase la una singura. Cu alte cuvinte nu pot construi decat un singur obiect din aceasta clasa. Tehnica de singleton se foloseste in special pentru a reduce consumul de memorie si pentru a centraliza anumite operatii catre un singur obiect. Daca avem nevoie sa mentinem o referinta catre un obiect in mai multe clase, putem sa il facem Singleton si sa obtinem acel obiect in fiecare componenta in care avem nevoie de el sau ca sa facem codul mai curat si cu un flow mai usor de urmarit, sa il instantiem doar intr-un singur loc si sa il transmitem ca argument.

La baza pattern-ului Singleton sta o metoda ce permite crearea unei noi instante a clasei daca aceasta nu exista deja. Daca instanta exista deja, atunci intoarce o referinta catre acel obiect. Pentru a asigura o singura instantiere a clasei, constructorul trebuie facut **protected** (un constructor privat impiedica reutilizarea sa sau accesul unei unitati de testare).

Singleton creaza premizele ca o anumita clasa sa fie instantiata doar o singura data, permitandu-se un acces global la instanta respectiva.

În aplicație, am implementat clasa Supermarket folosind design pattern-ul Singletone ,deoarece gestionează lista de produse ale supermarketului astfel ca folosind o singură instanță pot sa o gestionez mai ușor și pot să obțin mereu lista produselor cu toate modificările suferite.

2.Memento

Modelul Memento este un model de proiectare software , care oferă posibilitatea de a restabili un obiect la starea sa anterioară.

Cele 3 componente ale acestui design pattern sunt:

1.Memento –este creat de Originator și gestionat de CareTaker si gestioneaza starea internă a obiectului care este Originator pentru un anumit moment dat.

Originator- este obiectul are poate genera un Memento cu starea lui la momentul respectiv.

CareTaker- gestionează obiectele de tip Memento si nu are acces pe conținutul lor.

Am folosit acest design pattern pentru a gestiona modificarile suferite de produsele din supermarket .De asemenea prin acest pattern , pot readuce produsele la starea lor initiala dupa diferitele modificari.

3.Observer

Modelul observer este un model de proiectare software , în care un obiect , numit subiectul , menține o listă de persoane pe care o subordoneaza , cu numele de observatori , și îi notifică automat cu privire la orice modificări , de obicei, prin apelarea uneia dintre metodele lor .

Modelul observer poate provoca pierderi de memorie , pentru că în implementarea de bază necesită atât înregistrarea explicită și radierea explicita, pentru că subiectul are trimiteri puternice la observatori , menținându-i în viață . Acest lucru poate fi prevenit prin obiectul care deține trimiteri slabe la observatorilor .

Am folosit un Observer care primeste notificari atunci cand au loc modificari la nivelul listei de produse.Vom utiliza Observer atunci cand afisam produsele din supermarket intr-o interfata create ulterior .

**Definirea si detalierea metodelor testate prin Unit Testing**

In clasa Produse am testat 3 metode : adaosCategorie(),reducere(), impozit().

Metoda adaosCategorie()returneaza un float si reprezinta adaosul adaugat produselor in functie de categoria lor.

Metoda reducere() returneaza un float si reprezinta reducerea aplicata produselor in functie de statusul acestora : EsteInStoc sau NuInStoc.

Metoda impozit() calculeaza impozitul in functie de categorie.

In clasa Supermarket am testat urmatoarele metode:

-Supermarket(ArrayList<Produse>listaNoua)-am creat un supermarket bazat pe o lista primita ca parametru si am verificat daca lista este nula.

-public ArrayList<Produse>getListaProduse-metoda preia toate produsele din lista

-public void setListaProduse(ArrayList<Produse>list)-metoda seteaza lista din supermarket cu o lista primita ca parametru si o verifica daca este nula/goala.

-public Produse getProduse(Produse a)-metoda returneaza un produs din supermarket si parcurge lista pentru a gasi produsul ce trebuie returnat,iar daca nu il gaseste arunca exceptii si returneaza null.

-public void adaugareProduse(Produse a)-metoda adauga produse in lista si verifica daca exista deja produsul.Anunta observerii dupa adaugare si arunca o exceptie daca lista este nula.

-public void eliminaProduse(Produse a)-metoda elimina un produs daca el exista ,iar daca lista este nula arunca o exceptie.

-public Produse celMaiMarePret()-metoda returneaza cel mai are pret din cadrul produselor din lista ,iar daca lista este nula arunca exceptii.

-public float adaosPret()-metoda returneaza adaosul preturilor din supermarket si parcurge lista pentru a aduna preturile produselor,iar daca lista este goala returneaza 0 sau arunca exceptii pentru lista nula.

-public float calculeazaImpozit()-metoda calculeaza impozitul platit de supermarket prin adunarea impozitului pentru fiecare produs.Daca lista este goala returneaza 0,idar daca lista este nula arunca exceptii.

-public float calculeazaAdaosCategorie()-metoda calculeaza adaosul in functie de categoria produselor.Daca lista este goala returneaza 0,iar daca lista este nula arunca exceptii.

-public float calculeazaPlatiTotale-metoda calculeaza totalul platilor efectuate de catre supermarket adunand pretul produselor cu adaosul pe categorie.Daca lista este goala returneaza 0,iar daca lista este nula arunca exceptii.

-public void afiseazaProduse()-metoda afiseaza produsele din supermarket

-public void modificaPret(Produse a, int s)- metoda modifica pretul unui produs primit ca parametru cu alt pret primit ca parametru.Dupa ce se parcurge lista ,cand produsul este gasit ,inainte sa ii modific pretul il adaug intr-o lista si ii salvez starea.

-public void afiseazaProduseModificate()-metoda afiseaza produsele care au suferit modificari

-public void citireProduseDinFisier()-metoda adauga in lista produselor, acele produse citite din fisierul .txt

-public int numarElectronice()-metoda returneaza numarul produselor electronice din supermarket ,care sunt identificate pe baza codului de bare.Daca lista este goala returneaza 0, iar daca lista este nula arunca exceptii.

-public int numarProdusePatiserie()-metoda returneaza numarul produselor de patiserie din supermarket ,care sunt identificate pe baza codului de bare.Daca lista este goala returneaza 0, iar daca lista este nula arunca exceptii.

-public int numarDecoratiuni()-metoda returneaza numarul produselor decorative din supermarket ,care sunt identificate pe baza codului de bare.Daca lista este goala returneaza 0, iar daca lista este nula arunca exceptii.

-public int numarProduseVestimentare()-metoda returneaza numarul produselor vestimentare din supermarket ,care sunt identificate pe baza codului de bare.Daca lista este goala returneaza 0, iar daca lista este nula arunca exceptii.

-public int numarAlimente()-metoda returneaza numarul produselor alimentare din supermarket ,care sunt identificate pe baza codului de bare.Daca lista este goala returneaza 0, iar daca lista este nula arunca exceptii.

-public int numarProduse(int s)-metoda afiseaza produsele cu pretul mai mare sau egal decat pretul de la furnizor si se parcurge lista produselor pentru a creste cu o unitate pretul produselor care se incadreaza in aceasta categorie.Daca lista este goala returneaza 0 ,iar daca lista este nula arunca exceptii.

**Definirea si descrierea TestCase-urilor si Test Suite-urilor**

*1.AllTestsProduse*

In acest Test Suite sunt incluse cele 2 Test Case-uri utilizate pentru testarea metodelor clasei Produse. Cand rulam acest Test Suite se testeaza metodele clasei Produse prin cele 11 teste implementate.

TestAdaos- este Test Case-ul care testeaza metoda adaosCategorie() din clasa Produse. Am realizat 7 teste pentru aceasta metoda.Unul testeaza adaosulGeneral , 5 testeaza adaosul pe categorii iar ultimul testeaza adaosul in cazul produsului inexistent.

TestReducere- este Test Case-ul care testeaza metoda reducere() din clasa Produse. Am realizat 4 teste pentru aceasta metoda, dintre care unul este testul pentru reducereGenerala, 1 pentru produse in stoc, 1 pentru produse care nu sunt in stoc iar ultimul pentru produs inexistent.

*2.AllTestsSupermarket*

In acest Test Suite sunt incluse cele 14 Test Case-uri utilizate pentru clasa Supermarket si testeaza 41 de teste implementate.

TestAdaosPret- testeaza metoda de calcul a adaosului pretului produselor folosind o lista de produse.Am testat si pentru lista goala sau nula.

TestAdaugareProdus- testeaza metoda de adaugare a produselor in supermarket.Daca lista este nula arunca exceptii.

TestCalculeazaAdaosCategorie- testeaza metoda de calcul a adaosului produselor din supermarket in functie de categoria lor pe baza unei liste implementate.Am testat si pentru lsiat goala sau nula.

TestCalculeazaImpozitul- testeaza metoda de calcul a impozitului din clasa Supermarket. Am testat si pentru lista goala si nula a produselor.

TestCalculeazaTotalulPlatilor-testeaza metoda de calcul a platilor totale ale supermarket-ului .Pe langa testul platilor generale, am testat si in cazul in care lista produselor este goala sau nula.

TestCelMaiMarePret-testeaza metoda celMaiMarePret din clasa Supermarket.Primul test creeaza o lista cu 2 produse .Urmeaza compararea preturilor celor 2 produse si alegerea celui mai mare.Am testat si in cazul listei goale sau nule.

TestCitireDinFisier-testeaza metoda prin care adaugam produse in lista supermarket-ului cu date preuluate dintr-un fisier.Am creat un fisier produse.txt din care am preluat datele .Daca fisierul nu exista se arunca exceptii.

TestEliminareProduse-testeaza metoda de eliminare a unui produs din lista .Am testat si in cazul in care produsul este null dar si in cazul in care lista este nula sau goala.

TestModificaPretulProdusului-testeaza metoda de modificare a pretului produsului .Am testat si pentru cazul in care produsul este null, dar sin in cazul in care lista este nula sau goala.

TestProduseAlimentare-testeaza metoda prin care se returneaza numarul produselor alimentare din supermarket .Am testat si pentru lista goala care returneaza 0 si pentru lista nula care arunca exceptii.

TestProduseDecorative-testeaza metoda prin care se returneaza numarul produselor decorative din supermarket .Am testat si pentru lista goala care returneaza 0 si pentru lista nula care arunca exceptii.

TestProduseDePatiserie-testeaza metoda prin care se returneaza numarul produselor de patiserie din supermarket .Am testat si pentru lista goala care returneaza 0 si pentru lista nula care arunca exceptii.

TestProduseElectronice-testeaza metoda prin care se returneaza numarul produselor electronice din supermarket .Am testat si pentru lista goala care returneaza 0 si pentru lista nula care arunca exceptii.

TestProduseVestimentare-testeaza metoda prin care se returneaza numarul produselor vestimentare din supermarket .Am testat si pentru lista goala care returneaza 0 si pentru lista nula care arunca exceptii.

**Descrierea sumara a functiilor aplicatiei cu referire la pattern-uri si la metodele testate**

Aplicatia gestioneaza produsele din cadrul unui supermarket.Prin Singleton utilizam mai usor lista de produse si exista o singura instanta care sufera modificari.Prin Memento se poate vizualiza istoricul produselor ,iar interfata este notificata usor prin Observer.

Cele 3 design pattern-uri si metodele implementate usureaza functionarea aplicatiei.Cu ajutorul metodelor obtinem informatii cu privire la produse si la supermarket ,ne permite sa realizam modificari asupra produselor , putem gestiona listele cu produse etc.

**Bibliografie**

<http://acs.ase.ro/Media/Default/documents/designpatterns/Curs%20CTS%20%20Design%20Patterns%20v2.pdf>

+seminarii