**Academia de Studii Economice**

**Bucuresti**

**Calitate și testare software**

* **DOCUMENTATIE -**

**Stud. Anghel Maria Alexandra Georgiana**

**2016CUPRINS**

[Despre aplicatie 3](#_Toc451437098)

[Patternuri 3](#_Toc451437099)

[Testare 3](#_Toc451437100)

[Clase 3](#_Toc451437101)

[Metode: 4](#_Toc451437102)

[Test Case 5](#_Toc451437103)

[TestSuite: 8](#_Toc451437104)

[Bibliografie 9](#_Toc451437105)

# Despre aplicatie

Metodele folosite si testate reprezinta un fragment dintr-o aplicatie implementata cu scopul de a realiza un sistem informatic care să automatizeze activitatea de vânzare si activităţile adiacente acesteia. Este conceput pentru o firmă mică în care aceste activităţi se desfăşoară încă pe baza registrelor, condicilor care sunt incontestabil depăşite de tehnologiile actuale.

# Patternuri

Definirea si justificarea pattern-urilor implementate.

Patternuri implementate :

* Builder - utilizat in crearea de clienti;

-se creaza in mod structurat obiecte compleze;

* Decorator – utilizat in crearea comenzilor;

- acestea sunt extinse din obiectul de baza;

* Simple Factory – utilizat in crearea produselor;

-cestea sunt generate de o “fabrica”, se creaza obiecte din aceeasi familie;

* Singleton –utilizat in crearea administratorului

-in caazul de fata este un singur administrator;

# Testare

Clase::

* ClientBuilder : cu atributele : nume, nrTel, adresa;
* Client: cu atributele: nume, firma; telefon; adresa; data; codClient;
* Comanda: cu atributele: cantitate; produs; pret; nrComanda; tipComanda;
* Operatii : cu atributul: pret;
* Produs : cu atributele :codProdus; denumire; pret; stoc;
* ComandaDecorator : cu atributele: nume;
* ProdusSimpleFactory: cu atributele: codProdus; denumire; pret; stoc;
* AdministratorSingleton: cu atributele nume; telefon; adresa; dataAngajare;

## **Metode**:

Client contine urmatoarele metode:

* pentru citire : getNume, get \_firma, get \_telefon, get \_adresa, get \_data, get \_codCLient;
* pentru scriere: set\_nume, set\_firma, set\_telefon, set\_adresa, set\_data, set\_codCLient;
* constructori cu parametric si fara [arametri;
* pentru verificarea corectitudinii codului, daca acesta este format din 4 caractere;

Comanda prezinta urmatoarele functii:

- pentru citire: get\_cantitate, get\_produs, get\_pret, get\_nrComanda, get\_tipComanda; si pentru lista de comenzi getListaComenzi;

- pentru scriere: set\_cantitate, set\_produs, set\_pret, set\_nrComanda, set\_tipComanda; si pentru lista de comenzi setListaComenzi;

- constructori cu parametric si fara;

- pentru calculare valoare comanda fara TVA:

SE inmulteste cantitatea de produs cu pretul acestuia si aflam, astfel, valoarea fara TVA a comenzii respective.

* Pentru calcului valorii comenzii cu TVA:

Se inmultesc cantatitea produsului si pretul acestuia cu valoarea TVA stabilita de la inceput. (aici este 24%, se inmulteste cu 0,24)

* Pentru inserarea comenzilor efectuate intr-o lista:

Se creeaza o lista cu obiecte de tip comanda in care se insereaza, prin parcurgere, comenzile effectuate pana la momentul respective.

Produs are urmatoarele metode:

-get si set pentru citire si scriere, pentru atributele codProdus, denumire, pret, stoc;

-to strig pentru afisarea detaliilor unui produs;

Client Builder:

* Metode cityire, scriere, afisare string;
* Constructor pt creare de obiecte;
* Metode pt creare builder;

Comanda decorator:

-metode pentru trimitere, returnare comanda;

-afisare sir de stringuri;

-in comanda decorator: functia pentru decorarea comenzii;

Administrator Mock:

-constructor pentru creare obiecte;

-metode pentru citire si scriere attribute;

-functie pt indentificarea diferentelor de varsta;

-metoda pentru identificarea administrstilor mai invarsta si apropiati de pensie;

Produs Simple Factory:

-creare obiecte initiale;

-metode de craere obiecte factorii;

-functii pentru identificare tipuri de produe;

Adminstrator Singleton:

-metode pentru initializare obiect;

-functie pt creare instant unica;

## **Test Case**

* Test\_Administraor;

-testare creare corecta de instante:

public void test\_instanta() -

public void test\_instanta2()

-testare instanta nenula:

public void test\_instanta\_NotNull()

-returnare corecta attribute in cazul incercarii crearii mai multor instante

public void test\_instantaNUme()

public void test\_instantaTelefon()

public void test\_instantaAdresa()

public void test\_instantaDataAngajare()

-returnare corecta de stringuri:

public void test\_toString()

-verificare numar de caractere nr telefon:

public void test\_nrcaratereTelefon()

-introducere corecta de nr telefon:

public void test\_telefonCorrect()

-verificare aplicare oferta, notNull:

public void test\_notNullAplicareOferta()

-testrae limite:

public void test\_limite()

public void test\_limite2()

* Test\_Client;

-testrae introducere corecta date, constructor, get:

public void test\_constructor()

-introducree corecta atribute:

public void test\_setNume()

public void test\_setAdresa()

public void test\_setNrTel()

public void test\_getNUme()

public void test\_getNrTel()

public void test\_getAdresa()

-creare corecta de builder

public void test\_builderNume();

public void test\_builderAdresa()

-sir caractere correct builder:

public void test\_builderToSTring()

public void test\_ToSTring()

* Test\_Comanda;

-Creare corecta in urma executiei decoratorului:

public void test\_getDenumire()

* Test\_Operatii;

-Calculare pret correct:

public void test\_pretCorect()

* Test\_Produs;

-creare corecta factory:

public void test\_factory

* Test\_PizzaDecorator;

-creare corecta decorator:

public void test\_decorator()

* testClient;

public void testCod\_numarCaractere()

public void test\_getNume()

public void test\_getFirma()

public void test\_getTelefon()

public void test\_getAdresa()

public void test\_getData()

public void test\_getCod()

public void test\_analiz\_Client()

* testProdus;

-se verifica citirea din fisier a unei liste cu produse:

public void testDinFisier() {

* TestMock;

-e irelevant sa se faca teste pe obiecte mock

public void test\_Mock\_varsta()

-testare calcul diferenta cu ajutorul obiectelor mock:

public void test\_diferenta()

public void testChechCUeXceptie()

* testComanda;

public void testCalculValoare()

public void testAdaugareTVA()

public void testTipComanda()

public void test\_getCantitate()

public void test\_getProdus()

public void test\_getpret()

public void test\_getNrComanda()

public void test\_getTipComanda()

public void testListaComezi()

public void test\_crestereCantitate()

public void test\_scadereCantitate()

* + TestProdus2;

public void test\_getCodProdus()

public void test\_getDenumire()

public void test\_getPret() {

public void test\_getStoc ()

public void test\_CrezterePret()

public void test\_ScadePret()

public void test\_AplicaReducere()

## TestSuite:

Suita utilizata pentru rularea tuturor testelor din clase;

# Bibliografie

Site-uri

* <http://stackoverflow.com/questions/15456910/arraylist-equality-junit-testing>
* <http://www.tutorialspoint.com/java/java_date_time.htm>
* Online.ase.ro
* <http://junit.org/apidocs/org/junit>

Seminarii Calitate și testare software

Cursuri Calitate și testare software