**ACADEMIA DE STUDII ECONOMICE DIN BUCUREŞTI**

**PROIECT  
 CALITATE SI TESTARE SOFTWARE**

**BUCUR ANDREEA CRISTINA  
GRUPA 1049  
SERIA A**

**2015**

**CUPRINS**

[**1.** **Capitolul 1: Design Pattern-uri** 3](#_Toc405281320)

[**2.** **Capitolul 2: Definirea si detalierea metodelor testate prin Unit Testing** 4](#_Toc405281321)

[**3.** **Capitolul 3: Definirea si descrierea Test Case-urilor** 5](#_Toc405281322)

[**4.** **Capitolul 4: Definirea si descrierea Test Suite-ului**  8](#_Toc405281323)

1. **Capitolul 1: Design pattern-uri**

In realizarea proiectului am folosit trei design pattern-uri, si anume: Factory, Singleton si Observer.

Aplicatia analizata in cadrul proiectului reprezinta un soft pentru o societate de telefonie mobila. Societatea de telefonie mobila Vodafone prezinta o gama larga de produse pe langa serviciile oferite. Astfel, se vand mai multe tipuri de produse din familia produselor electronice si anume: incarcatoare, telefoane, tablete etc. Clientii vor putea fi notificati atunci cand un produs este sau nu in stoc in momentul solicitarii acestuia. Pentru a putea realiza orice fel de operatiune, utilizatorii trebuie sa se autentifice cu un nume de utilizator si parola unice.

**Factory** este pattern-ul ce defineste o metoda pentru creare de obiecte din aceeasi familie (intefata) in subclase.

Interfata IProduse are doua metode si este implementata de clasele concrete ce reprezinta tipurile de produse si anume: clasa Telefon, Tableta si Incarcator. ProduseFactory reprezinta clasa ce defineste interfata generatorului de obiecte.

**Singleton** reprezinta pattern pentru crearea unei instante unice dar care poate fi referita de mai multe ori.

Utilizator() reprezinta constructorul privat pentru a nu putea fi instantiat. Instance este atributul static, privat, de tipul clasei ce reprezinta instanta unica. GetInstance() reprezinta metoda publica ce da acces la instanta unica ce va fi creata la primul apel al metodei.

**Observer** este pattern-ul ce defineste un handler pentru diferite evenimente.

SubjectObserver reprezinta clasa abstracta ce defineste interfata obiectelor gestioneaza evenimente si care sunt observabile. Observer este interfata ce defineste modalitatea in care sunt notificati observatorii. Clienti reprezinta clase ce extinde interfata Observer si in care gasim functia modifica() declarata in Observer.

Clasa Test reprezinta clasa in care avem functia void main() si in care sunt testate toate functiile.

1. **Capitol II : Definirea si detalierea metodelor testate prin Unit Testing**

Pentru testarea metodelor am realizat JUnit Test Case-uri separate pentru fiecare clasa din care am luat metode pentru testat.

Metoda **setCNP**(String cnp) din clasa Clienti seteaza o anumita valoare pentru atributul cnp din cadrul clasei. Prin intermediul metodei se verifica cnp-ul sa nu aiba o lungime de 13 caractere, sa nu inceapa cu cifra 1 sau cu cifra 2.

Metoda **setVenitLunar**(double venitLunar) din clasa Clienti seteaza valoare pentru venitul lunar al clientului. Prin intermediul metodei se verifica valoarea extrema maxima a venitului si daca venitul este mai mic decat 0.

Metoda **listaNume**(String nume, String prenume) din clasa Clienti verifica daca numele si prenumele clientilor au anumite lungime dupa care ii adauga intr-o lista.

Metoda **calculVenitAnual**(double venitLunar) din clasa Clienti calculeaza venitul anual al clientului si verifica daca este egal cu 0, daca venitul lunar este mai mare decat un venit maxim dat sau daca este numar real. Pentru a putea verifica daca este numar real am folosit functia privata esteValid(double valoare) in care verificam daca un numar este numar real sau nu.

Metoda **calculBonu**s(Clienti detaliiClienti) din clasa Clienti calculeaza bonusul care se adauga la salariul pentru client in functie de categoria in care se incadreaza. Daca salariul este mai mic de 1500 bonusul va fi 50, altfel bonusul va fi 100.

Metoda **setTipProdus**(Produs tipProdus) din clasa Produse seteaza tipul produsului ce poate fi achizitionat. Se verifica daca produsul se incadreaza in categoriile: tableta, telefon sau incarcator, daca nu se genereaza exceptie. Totodata, verificam daca lungimea tipului produsul este mai mare decat lungimea maxima, iar in acest caz se arunca exceptie si daca tip produsului are mai putin de 4 caractere.

Metoda **getYearFromDate**(Date data) din clasa Factura extrage anul din data pe care il comparam cu anul din data facturarii pentru ca acesta sa nu fie diferit de anul 2015.

Metoda **getDayFromDate**(Date data) din clasa Factura extrage ziua din saptamana dintr-o data, iar aici se verifica facture sa nu fie emisa intr-o zi de sambata sau de duminica.

Metoda **totalFactura**() din clasa Factura calculeaza suma total a unei facturi pe baza pretului unui produs. Se face o lista de produse iar numarul de produse din aceasta lista reprezinta cantitatea de produse folosita in cadrul functiei pentru calcularea sumei totale. Se verifica in functie ca lista de produse sa aiba cel putin un produs.

Metoda **setPret**(double pret) din clasa Incarcator seteaza o anumita valoare pentru pretul produsului. In cadrul metodei se verifica valoarea extrema a pretului, daca acesta este mai mic decat 0 sau daca este egal cu 0, cazuri in care se arunca exceptii.

Metoda **setDenumire**(String denumire) din clasa Telefon atribuie valori pentru denumirea obiectelor din clasa.Se verifica daca denumirea are mai putin de 5 caractere, se verifica limita extrema a denumirii, sa nu contina alte caractere in afara de litere si cifre.

1. **Capitolul III : Definirea si descrierea Test Case-urilor**
2. **TestClienti**

In cadrul clasei de test TestClienti am testat metodele din clasa Clienti.

Pentru metoda **setCNP(**String cnp) am realizat urmatoarele teste:

* testCNPScurt() ce testeaza daca CNP-ul introdus are mai putin de 13 caractere
* testCNPLung() ce testeaza daca CNP-ul introdus are mai mult de 13 caractere
* testFirstNumberCNP() testeaza daca CNP-ul introdus incepe cu alta cifra in afara de 1 si 2, caz in care se arunca exceptie.

Pentru metoda **setVenitLunar**(double venitLunar) am realizat urmatoarele teste:

* testClientiSalariuMax() ce testeaza valoarea extrema pozitiva a salariului unui client
* testClientiSalariuValNegativa() ce testeaza daca valorile introduse pentru salariu sunt negative

Pentru metoda **listaNume**(String nume, String prenume) am realizat urmatoarele teste:

* testListaNumeNull() in care se verifica daca numele sau prenumele nu au asignate valori prin intermediul lui assertNull()
* testListaNumeNotNull() in care se verifica daca lista nu este nula prin intermediul lui assertNotNull()

Pentru metoda **calculBonus**(Clienti detaliiClienti) am realizat urmatoarele teste:

* testCalculBonus() ce testeaza corectitudinea metodei prin intermediul lui assertEquals()
* testVenitNull() ce testeaza valoarea bonusului in cazul in care venitul este zero.

Pentru metoda **calculVenitAnual**(double venitLunar) am realizat urmatoarele teste:

* testCalculVenitAnual() ce testeaza corectitudinea metodei prin intermediul lui assertEquals();
* testVenitMaxim() ce testeaza metoda in cazul in care clientul are un venit mai mare decat venitul maxim dat.
* testValoareRealaVenit() teasteaza daca venitul este numar real cu doua zecimale.
* testVenitZero() testeaza metoda in cazul in care venitul clientului este 0.

1. **TestProduse**

Pentru metoda **setTipProdus**(String tipProdus) am realizat testele urmatoare:

* testTipProdusLungimeMax() ce testeaza limita extrema a tipului produsului, in cazul in care tipul produsului depaseste limita maxima data se arunca exceptie
* testTipProdusLungimeMin() ce testeaza ca tipul produsului sa nu aiba o lungime mai mica decat 4 caractere
* testTipProdus() ce testeaza daca tipul produselor se incadreaza in tipurile specificate, si anume:tableta, telefon sau incarcator.

1. **TestProduseIncarcator**

Pentru metoda **setPret**(double pret) am realizat urmatoarele teste:

* testPretMaxim() ce testeaza valoarea extrema a pretului, in cazul in care pretul introdus depaseste valoarea maxima data se arunca exceptie
* testPretZeroValue() ce testeaza daca pretul este 0, caz in care arunca exceptie
* testPretValoareNegativa() ce testeaza daca pretul ia o valoare negative, caz in care se arunca exceptie

1. **TestProduseTelefon**

Pentru metoda setDenumire(String denumire) am realizat urmatoarele teste:

* testLungimeMinimaDenumire() ce testeaza ca lungimea minima a denumirii sa nu fie mai mica de 5 caractere
* testLungimeMaximaDenumire() ce testeaza ca lungimea maxima a denumirii sa nu fie mai mare decat lungimea maxima data, altfel se arunca exceptie
* testFirstCharacter() ce testeaza ca primul character sa nu fie un simbol sau semn de punctuatie
* testNoLetterInDenumire() ce testeaza ca in denumire sa nu se permita sa nu existe litere

1. **TestFactura**

Pentru metoda **getYearFromDate**(Date data) am realizat urmatoarele teste:

* testAnFacturaMaiMic() ce testeaza ca anul extras din data facturarii sa nu fie mai mic de 2015
* testAnFacturaMaiMare() ce testeaza ca anul extras din data facturarii sa nu fie mai mare decat 2015

Pentru metoda **getDayFromDate**(Date data) am realizat urmatoarul test:

* testFacturareSambDum() ce testeaza ca facturarea sa se faca doar in 5 din 7 zile ale saptamanii, iar in functie avem conditia ca zilele in care sa nu se faca facturarea sa fie sambata sau duminica

Pentru metoda **totalFactura**() am realizat urmatoarele teste:

* testTotalFacturaNull() ce verifica daca suma totala a facturii este nula, caz in care se arunca exceptie
* testTotalFacturaNotNull() ce verifica daca suma totala a facturii nu este nula.

1. **TestCitireFisier**

Pentru testele realizate pe citirea din fisier am folosit clasa Clienti si am introdus date corespunzatoare atributelor si metodelor din clasa mentionata.

* testCitireFisier() testeaza corectitudinea datelor introduse in fisier corespunzatoare cu atributele clasei folosite
* testCitireNull() verifica daca venitul anual calculat conform datelor introduse este null
* testCitireNotNull() verifica daca venitul anual calculate conform datelor introduse nu este null

1. **Capitolul IV : Definirea si descrierea Test Suite-ului**

Un Test Suite reprezinta o colectie de Test Case-uri ce sunt destinate pentru a fi utilizate in testarea programelor software pentru a demonstra ca are un anumit set specific de comportamente. Un Test Suite contine adesea instructiuni detaliate sau obiective pentru fiecare colectie de Test Case-uri si informatii cu privire la configuratia sistemului care urmeaza sa fie folosite in timpul testarii.

Pentru incadrarea Test Case-urilor in Test Suite se realizeaza constructorul cu parametrii in fiecare clasa de test.

Se realizeaza un JUnit Test Case nou in care se face o metoda ce va apela celelalte clase de test. Se initializeaza un obiect nou de tipul clasei Test Suite-ului , iar pentru adaugarea partiala de metode se foloseste addTest(new NumeleTestCase-ului(“metodaTestata”). Pentru adaugarea totala de metode se foloseste addTestSuite(NumeleTestCase-ului.class), si astfel vor fi apelate toate metodele testate in clasa respectiva.

**Bibliografie**

1. [**http://acs.ase.ro/software-quality-testing**](http://acs.ase.ro/software-quality-testing)
2. **[www.tutorialspoint.com](http://www.tutorialspoint.com)**
3. [**www.wikipedia.com**](http://www.wikipedia.com)