CALITATE SI TESTARE SOFTWARE

Proiect

Stanculescu Raluca,

Grupa 1068, Seria A

Cuprins

1. [Definirea si justificarea pattern-urilor implementate](#h.ukzvujjp0u2q)………………………….………… 2

[2. Definirea si detalierea metodelor testate prin Unit Testing](#h.v8o0p1qql4rg)…………………………..…… 9

[3. Definirea si descrierea Test Case-urilor](#h.eoo37jgv7r65)………………………………………………..… 10

[4. Definirea si descrierea Test Suite-urilor](#h.ioejyzbk55km)………………………………………………….. 11

[5. Descrierea sumara a functiilor aplicatiei cu referire la pattern-uri si la metodele testate](#h.anndyum7fbz2)... 12

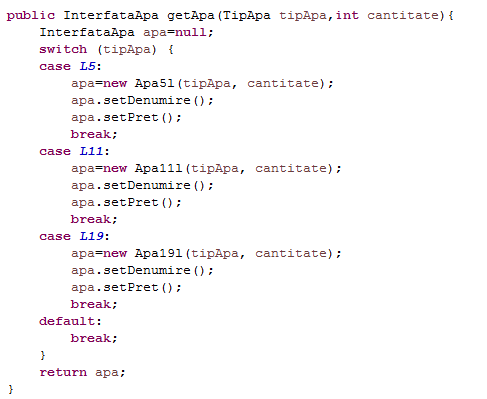
[6. Bibliografie](#h.hq030x8q2itr)………………………………………………………………………………. 12

### Definirea si justificarea pattern-urilor implementate

* Factory Method

Patternul Factory face parte din categoria *Creationale* si ca atare rezolva problema crearii unui obiect fara a specifica exact clasa obiectului ce urmeaza a fi creat. Acest lucru este implementat prin definirea unei metode al carei scop este crearea obiectelor. Metoda va avea specificat ca parametru de intors in antet un obiect de tip parinte, urmand ca, in functie de alegerea programatorului, aceasta sa creeze si sa intoarca obiecte noi de tip copil.

Am uitilizat acest pattern, deoarece aplicatia ofera acces la o gama larga de produse ce fac parte din cadrul aceleiasi familii, diverse sortimente de apa (5L, 11L, 19L).

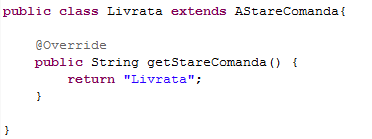


* State

Patternul State face parte din categoria *Comportamentale* si gestioneaza obiecte al caror comportament depind de starea lor, numărul de stări posibile varrind in functie de complexitatea sistemului. Aplicația tratează un anumit eveniment diferit în funcție de starea unui obiect.

Am ales acest pattern deoarece o comanda gestionata de aplicatie se poate afla in mai multe stari (Inregistrata, In desfasurare, Finalizata) in timpul procesarii acesteia.

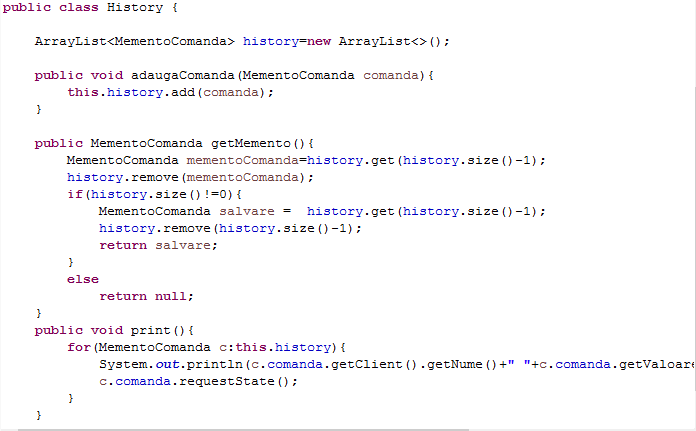




* Memento

Patternul Memento face parte din categoria *Comportamentale* si salveaza si restaureaza starea unui proiect. Aplicația permite salvarea stării unui obiect, imaginile stării obiectului pentru diferite momente fiind gestionate separat si obiectul își poate restaura starea pe baza unei imagini anterioare.

Am utilizat acest pattern, deoarece aplicatia implementeaza optiunea de a gestiona starile anterioare ale unei comenzi.

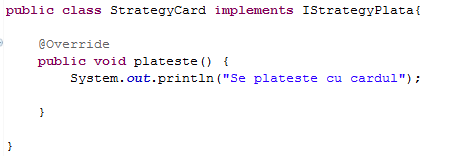


* Strategy

Patternul Strategy face parte din categoria *Comportamentale* si schimba la run-time functia executata. Algoritmul/functia utilizata pentru procesarea unor date se alege la run-time. Nu impune limite privind un numar maxim de functii/algoritmi ve pot fi folositi.

Am implementat acest pattern deoarece aplicatia ofera mai multe modalitati de plata a unei comenzi (Card, Cash).



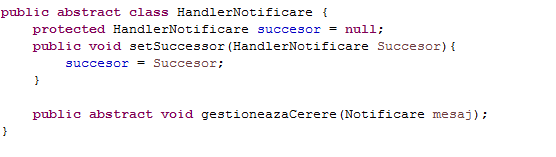


* Chain of responsability

Patternul Chain of responsability face parte din categoria *Comportamentale* si gestionează o secvență de acțiuni ce pot procesa un eveniment sau un obiect, trateaza un eveniment sau un obiect diferit in functie de starea acestuia. Gestiunea tuturor cazurilor ar implica o structură complexă care să verifice toate cazurile particulare.

Am ales acest pattern deoarece aplicatia realizeaza notificarea utilizatorilor atunci cand modulul hartii de vizualizare in timp real a comenzii este activata.

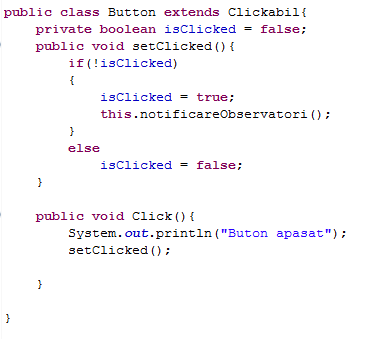


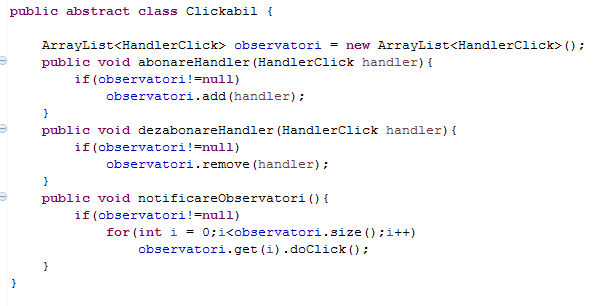


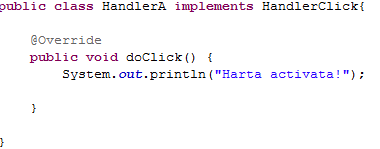
* Observer

Patternul Observer face parte din categoria *Comportamentale* si defineste un handler pentru diferite evenimente. Defineste o relatie de dependenta 1..\* intre obiecte astfel incat cand un obiect isi schimba starea, toti dependentii lui sunt notificati si actualizati automat. Faciliteaza mai buna organizare a comunicatiei dintre clase in functie de rolurile/comportamentul acestora.

Am implementat acest pattern deoarece atunci cand un utilizator da click pe butonul de vizualizare a hartii, mai multe notificari ii sunt trimise utilizatorului pentru a a avea toate detaliile unei comenzi.



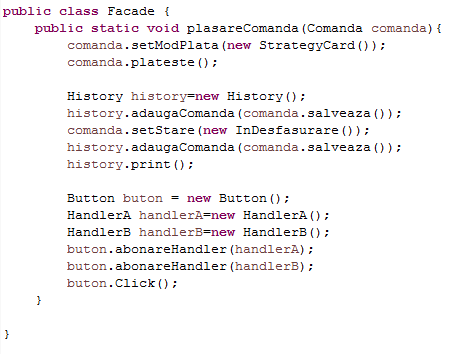




* Facade

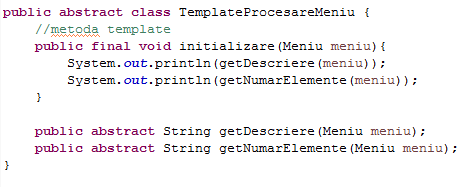
Patternul Facade face parte din categoria *Structurale* si simlifica executia unu scenariu complex.

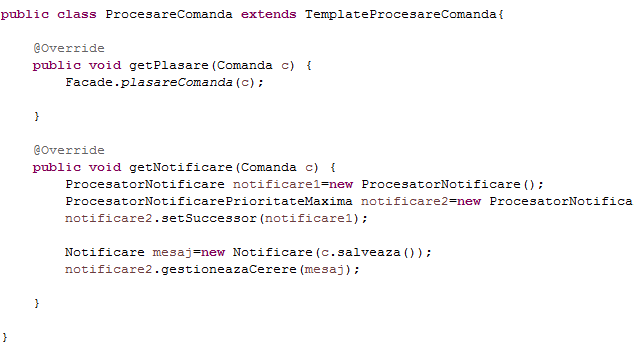
Am utilizat acest pattern deoarece aplicatia contine mai multe clase ce trebuiesc apelate print metodele specifice (vizualizarea comenzii si gestionarea butonului handler.



* Template

Patternul Template face parte din categoria *Comportamentale* si gestionează un șablon fix de acțiuni. Am utilizat acest pattern deoarece algoritmul presupune o secventa predefinita si fiza de pasi.





### 2. Definirea si detalierea metodelor testate prin Unit Testing.

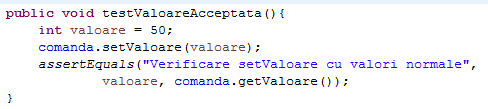
* Metoda testareNumeNull() testeaza pentru clasa Client, daca obiectul este instantiat cu valoare nul pentru atributul nume.
* Metoda testareNumar() verifica daca numarul de telefon al clasei Client contine lietere,
* Metoda testareNumeCifre() realizeaza un test asupra numelui clientului, acesta neputand fi format din cifre.
* Metoda testConstructorValoriNormale() realizeaza un test pentru valorile normale ale constructorului din clasa Comanda.
* Metoda testConstructorProduseNull() verifica daca lista de produse asignata unei comenzi este nula sau nu.
* Metoda testConstructorClientNull() realizeaza un test pentru a verifica daca o comanda este realizata de un client, adica constructorul contine o valoare nenula pentru campul Client.
* Metoda testModalitatePlata() face un test pentru a vedea daca a fost aleasa o modalitate de plata a comenzii, in caz contrar testul va pica.
* Metoda testValoareAcceptata() realizeaza un test pe o valoare normala acceptata a unei comenzi.
* Metoda testValoareAcceptataMinima() realizeaza un test pe o valoare minima acceptata a unei comenzi.
* Metoda testValoareNeacceptata() realizeaza un test pe o valoare neacceptata a unei comenzi, aceasta fiind mai mica sau mai mare de o valoare impusa.

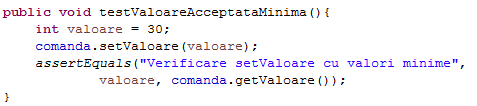
### 

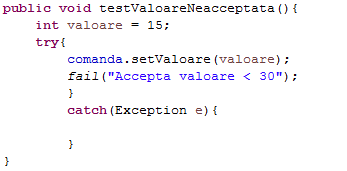
### 3. Definirea si descrierea Test Case-urilor

Un Test Case reprezinta o clasa ce defineste setul de obiecte pentru a rula mai multe teste.

* TestClient reprezinta clasa de obiecte ce ruleaza mai multe teste pe atributele clasei Client. Este formata din testele: testareNumeNull(), testareNumar(), testareNumeCifre.
* TestComandaConstructor reprezinta o clasa ce defineste setul de obiecte pentru a rula mai multe teste asupra constructorului din clasa Comanda. Este formata din metodele: testConstructorValoriNormale(), testConstructorProduseNull(), testConstructorClientNull(),testModalitatePlata().
* TestComandaValoare repezinta o clasa ce defineste setul de obiecte pentru a rula mai multe teste asupra valorii totale a comenzii inregistrate de catre un client. Este formata din metodele: testValoareAcceptata(), testValoareAcceptataMinima(), testValoareNeacceptata().



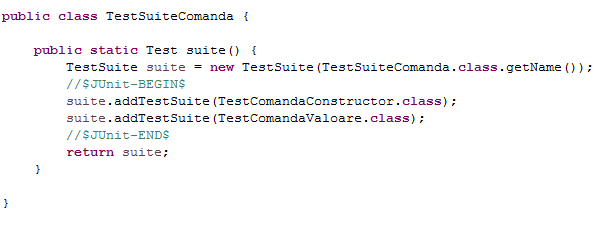




### 4. Definirea si descrierea Test Suite-urilor

Un Test Suite reprezinta o colectie de cazuri de testare (Test Case-uri).

* TestSuiteComanda reprezinta o colectie de cazuri de testare ale clasei Comanda., acesta fiind format din TestComandaConstructor si TestComandaValoare.



### 

### 5. Descrierea sumara a functiilor aplicatiei cu referire la pattern-uri si la metodele testate

Aplicatia este utilizata pentru inregistrarea comenzilor unei firme de apa, pentru livrarea lor ulterioara. Astfel avem mai multe produse ce fac parte din aceeasi categorie, pentru care putem implementa Factory.

In momemtul inregistrarii unei comenzi clientul trebuie sa aleaga o modalitate de plata: card sau cash (Strategy), iar acesta poate vizualiza starile anterioare ale comenzii (Memento). O comanda se poate afla in mai multe stari: inregistrata, in desfasurare si finalizata (State).

Clientul, prin intermediul aplicatiei, poate vizualiza o harta cu traseul comenzii pana la destinatie printr-un buton ce activeaza harta (Observer).

Aplicatia va notifica clientul cand rezervere de apa sunt pe sfarsite pentru a putea da o comanda in timp util(Chain). Toate operatiunile sunt facute printr-o interfata usor de utilizat, clientul nefiind nevoit sa realizeze multe operatiuni (Facade & Template).

### 6. Bibliografie

<http://acs.ase.ro/software-quality-testing>

<http://cursuri.cs.pub.ro/~poo/wiki/index.php/Design_Patterns_Basics>