Proiect Calitate si Testare Software

Prof. coordonator: DR. CODRIN-FLORENTIN NISIOIU

Student: Alexandru Mănescu, grupa 1070

# Definirea si justificarea pattern-urilor implementate

## Builder

Pentru crearea clientilor am folosit Builder. Acest design pattern separa constructia unui obiect complex de reprezentarea sa cu scopul de a crea reprezentari diferite urmand acelasi proces de constructie.



## Simple Factory

Implementeaza un mecanism centralizat prin care crearea obiectelor este transparenta pentru client; prin interfața publică clientul știe cum să creeze obiecte însă nu știe cum este implementat acest lucru.



## Singleton

Modelul Singleton se bazează pe existența unei singure instanțe ce poate fi creata o singură dată dar care poate fi referită de mai multe ori.



## Chain of responsability

Pentru secventialitatea terapiilor am folosit Chain of responsability, care gestioneaza tratarea unui eveniment de catre mai multi furnizori de solutii. Fiecare handler este „constient” doar de urmatorul handler, deci ordinea conteaza.



# Definirea si detalierea metodelor testate prin Unit Testing (minim 10 metode)

1. ExistaNume
2. VerificaNume
3. ExistaSingleton
4. ExistaMasaj
5. ExistaDrenaj
6. ExistaClient
7. ExistaOraInceput
8. ExistaOraSfarsit
9. ExistaInterval
10. ExistaTratament1
11. ExistaTratament2

# Definirea si descrierea Test Case-urilor

1. ExistaNume

Verificam ca s-a alocat memorie pentru nume.

1. VerificaNume

Validam ca numele este cel introdus.

1. ExistaSingleton

Verificam unicitatea instantei si ca s-a alocat memorie pentru aceasta.

1. ExistaMasaj

Verificam ca s-a alocat memorie pentru tratamentul masaj.

1. ExistaDrenaj

Verificam ca s-a alocat memorie pentru tratamentul masaj.

1. ExistaClient

Verificam ca s-a alocat memorie pentru clientul.

1. ExistaOraInceput

Verificam ca s-a alocat memorie pentru ora initiala.

1. ExistaOraSfarsit

Verificam ca s-a alocat memorie pentru ora finala.

1. ExistaInterval

Verificam ca s-a intervalul nu este degenerat.

1. ExistaTratament1

Verificam ca tratamentul a fost efectuat corect.

# Definirea si descrierea Test Suite-ului

Test suite-ul grupeaza Test Case-urile care verifica functionalitatea modulului ce asociaza clientii si terapiile dorite.

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

Aplicația va genera aceste variante de planificare cu ajutorul unor algoritmi ce vor avea la bază teoria din domeniile Programarii Evolutive si a Algoritmilor Genetici.

Aplicația va fi folosită pe masura ce trece timpul si se efectueaza terapiile pentru a centraliza feedback-ul clientilor. Pentru a avea o vedere de ansmblu a sugestiilor si reclamatiilor se doreste implementarea elementelor grafice, precum diagrama pie chart, pentru a vizualiza proportile dintre cuvintele cheie regasite in cadrul celor impartasite de clienti. Este esential ca solutia software sa aiba o interfata bine gandita pentru a fi utilizata cat mai usor si rapid. Elementele din cadrul fiecarui modul trebuie sa fie intuitive, folosind iconite deja consacrate si dispuse cat mai ordonat. Denumirile folosite in cadrul aplicatiei (sectiuni, meniuri, liste, butoane etc.) trebuie sa fie sugestive pentru o utilizare optima.