**Universitatea Tehnica a Moldovei**

**Facultatea Calculatoare Informatică și Microelectronică**

**Departamentul ISA**

**Raport**

**Lucrarea de laborator Nr. 2**

**la *Ingineria Produselor Program***

## Tema: Șabloane structurale

A elaborat st. gr. TI-153 Suruceanu V.

A verificat asist.univ. I.Costiuc

**Chișinău 2018**

**Scopul lucrărei:** De studiat și de implementat șabloane structurale.

**Mersul lucrărei:**

**Link-ul spre repozitoriu:** **https://github.com/dumitritag/lab2**

Efectuând lucrarea de laborator am implementat 5 șabloane structurale: Adapter, Bridge, Decorator, Facade, Flyweight.

1. **Adapter Design Pattern**

Adapter design pattern transformă interfața unei clase intr-o altă interfață care este așteptată de client. Șablonul dat permite compatibilitatea claselor care altfel nu ar putea face acest lucru din cauza interfețelor incompatibile, vezi figura 1.

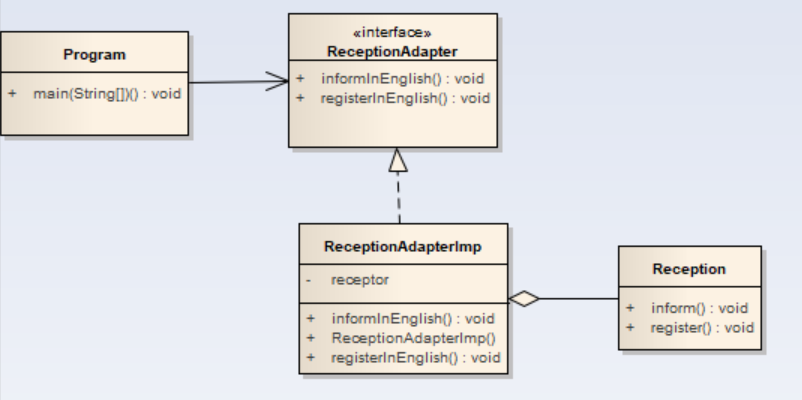


Figura 1. Diagrama UML. Adapter Design Pattern

1. **Bridge Design Pattern**

Bridge design pattern decuplează o abstractizare de implementarea ei, astfel încât cele două să poată varia independent. Bridge pattern pune abstractarea și implementarea în două diferite ierarchii de clase ca ambele să poată fi extinse independent, vezi figura 2.

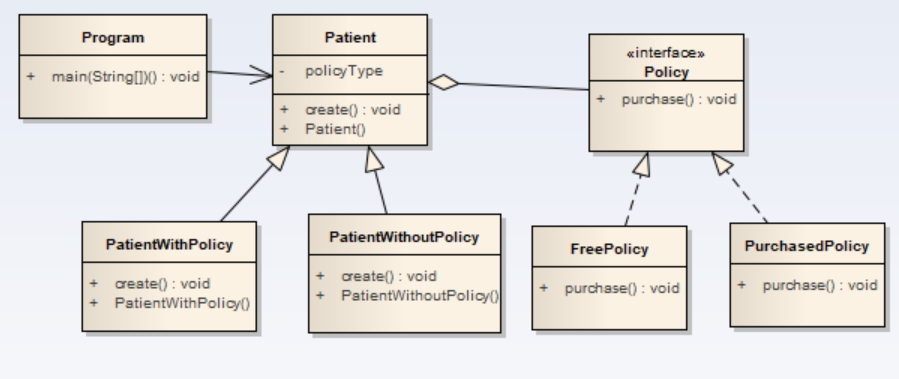


Figura 2. Diagrama UML. Bridge Design Pattern

1. **Decorator Design Pattern**

Decorator design pattern atașează dinamic responsabilități suplimentare la un obiect. Decoratorii oferă o alternativă flexibilă la generarea subclaselor în scopul extinderii funcționalităților, vezi figura 3.

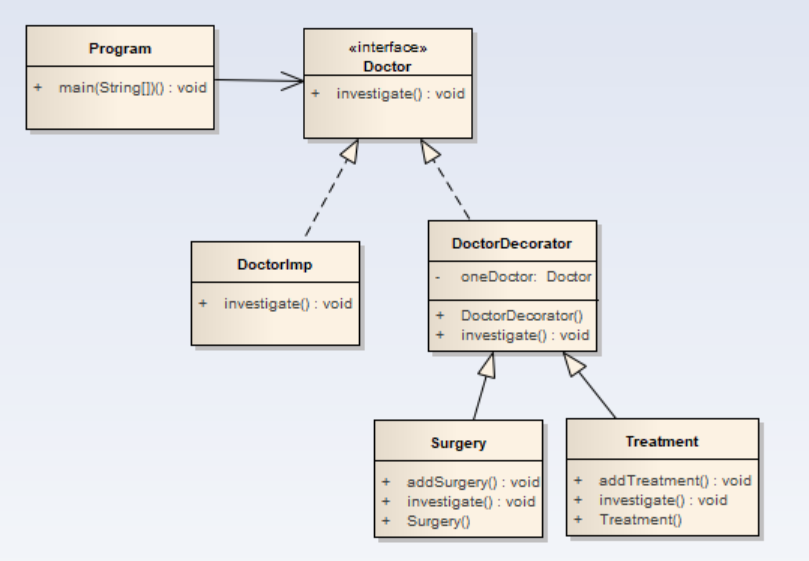


Figura 3. Diagrama UML. Decorator Design Pattern

1. **Facade Design Pattern**

Facade design pattern oferă o interfață unificată pentru un set de de clase dintr-un subsistem. Șablonul dat definește o intefață de nivel înalt care facilitează utilizarea unui subsistem. Facade pattern simplifică interacțiunea clientului cu un sistem complex, prin localizarea interacțiunilor cu sistemul într-o singură interfață. Ca rezultat clientul va interacționa cu un singur obiect, în loc să fie impus să interacționeze direct cu obiectele complicate care formează subsistemul (fig. 4).

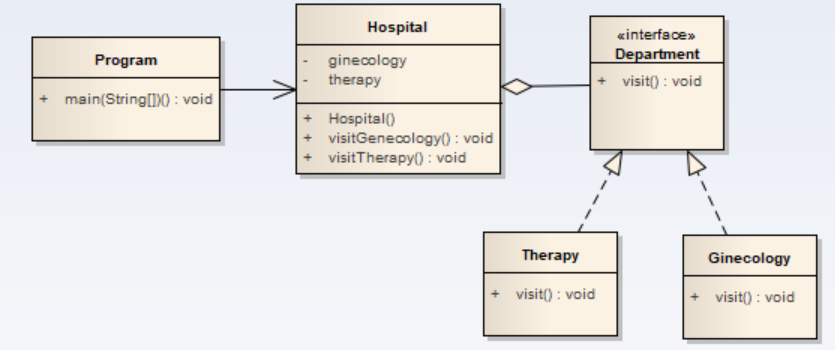


Figura 4. Diagrama UML. Facade Design Pattern

1. **Flyweight Design Pattern**

Flyweight design pattern se utilizează pentru a crea o mulțime de obiecte noi, dar în loc să fie create obiecte noi de fiecare dată, Flyweight pattern permite reutilizarea obiectelor deja existente. Șablonul dat poate fi folosit pentru a reduce memoria necesară, timpul de instanțiere și costurile relatate la crearea obiectelor noi (fig. 5).

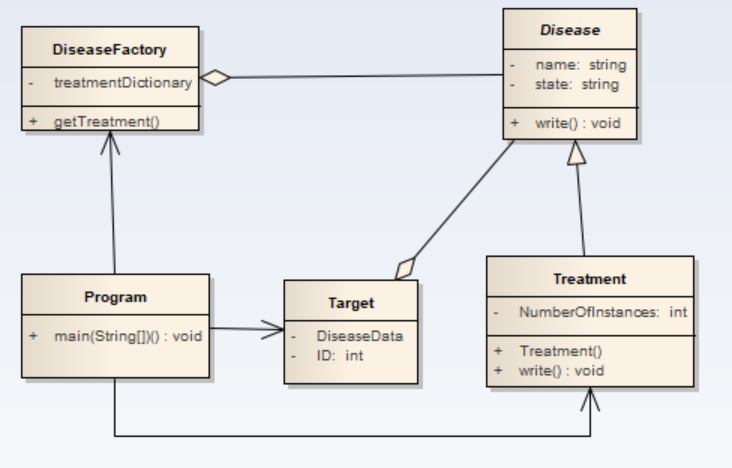


Figura 5. Diagrama UML. Flyweight Design Pattern

**Concluzii**

În această lucrare de laborator mi-am dezvoltat abilitățile practice în implementarea șabloanelor structurale, ele reprezentând modalități de structurare a obiectelor într-un limbaj de programare orientat pe obiect. Șabloanele studiate vor fi utile pentru utilizarea corectă a obiectelor, pentru testarea programelor deja create, căci șabloanele cum ar fi Facade, Bridge, Decorator pot simplifica cuplarea claselor, permițând să se modifice proiectele deja create. Șablonul Adapter permite unor clase cu interfeţe diferite să lucreze împreună (prin “traducerea” interfeţelor).