

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**Лабораторна робота №** 3

з дисципліни “Об'єктно-орієнтоване програмування та шаблони проєктування”

тема “Структурні шаблони проектування”

|  |  |  |
| --- | --- | --- |
| Виконав  студент 2 курсу  групи КП-02  Товстига Максим Анатолійович (*прізвище, ім’я, по батькові*)  варіант №17 |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладач  Заболотня Тетяна Миколаївна  (*прізвище, ім’я, по батькові*) |

Київ 2020

**Мета роботи**

Навчитися на практиці визначати, який структурний шаблон проектування слід використовувати в залежності від умов задачі.

**Постановка завдання**

1. Реалізувати різноманітні методи проведення віртуального медичного огляду лікарями як мінімум трьох спеціалізацій (стоматолог, хірург, ортопед і т.ін.). Забезпечити підтримку віртуального медичного огляду певною множиною лікарів.
2. Розробити модуль програмного забезпечення, що використовується у бібліотеці, який буде дозволяти читачу користуватися послугами електронного каталогу тільки у випадку, якщо у даного читача є власний логін у цьому програмному забезпеченні. У протилежному випадку модуль повинен пропонувати читачу зареєструватися у програмі.

**Тексти коду програми**

|  |
| --- |
| **Program.cs (task1)** |
| **using** System;  **using** **static** System.Console;    **namespace** task1  {  **class** Program  {  **static** **void** Main(**string**[] args)  {  MedicalExamitaion ex = new MedicalExamitaion("Shevchenko", "Petrashenko", "Rebrov", "Myhailichenko", "Lobanovskyi");  ex.To\_Examine();  }  }    **class** MedicalExamitaion  {  Dentist dentist;  Surgeon surgeon;  Traumatologist traumatologist;  Urologist urologist;  Otorhinolaryngologist otorhinolaryngologist;    **public** MedicalExamitaion(**string** surgeon\_name, **string** dentist\_name,  **string** traumatologist\_name, **string** urologist\_name, **string** otorhinolaryngologist\_name)  {  surgeon = new Surgeon(surgeon\_name, 27);  dentist = new Dentist(dentist\_name, 65);  traumatologist = new Traumatologist(traumatologist\_name, 54);  urologist = new Urologist(urologist\_name, 20);  otorhinolaryngologist = new Otorhinolaryngologist(otorhinolaryngologist\_name, 45);  }  **public** **void** To\_Examine()  {  dentist.Examine();  urologist.Examine();  surgeon.Examine();  traumatologist.Examine();  otorhinolaryngologist.Examine();  otorhinolaryngologist.FinishExamination();  traumatologist.FinishExamination();  }  }  **abstract** **class** Doctor  {  **public** **string** name;  **private** **int** age;  **public** **int** Age { **get** { **return** age; } **set** { **if** (**value** >= 20) { age = **value**; } } }  **public** **abstract** **void** Examine();  }  **class** Surgeon : Doctor  {  **public** Surgeon(**string** name, **int** age)  {  **this**.name = name;  Age = age;  }    **public** **override** **void** Examine()  {  WriteLine("Surgeon - U are ok!.");  }  }  **class** Dentist : Doctor  {  **public** Dentist(**string** name, **int** age)  {  **this**.name = name;  Age = age;  }    **public** **override** **void** Examine()  {  WriteLine("Dentist - Your teeth is OK. U can relax!");  }  }  **class** Traumatologist : Doctor  {  **public** Traumatologist(**string** name, **int** age)  {  **this**.name = name;  Age = age;  }    **public** **override** **void** Examine()  {  WriteLine("Traumatologist - Do an X-ray.");  }    **public** **void** FinishExamination()  {  WriteLine("Traumatologist - Your foot is not OK, i will fix it!");  }  }  **class** Urologist : Doctor  {  **public** Urologist(**string** name, **int** age)  {  **this**.name = name;  Age = age;  }    **public** **override** **void** Examine()  {  WriteLine("Urologist - Gratz, it's ok!");  }  }  **class** Otorhinolaryngologist : Doctor  {  **public** Otorhinolaryngologist(**string** name, **int** age)  {  **this**.name = name;  Age = age;  }    **public** **override** **void** Examine()  {  WriteLine("Otorhinolaryngologist - Do cardiogram");  }    **public** **void** FinishExamination()  {  WriteLine("Otorhinolaryngologist - Your heart is OK.");  }  }  } |

|  |
| --- |
| **Program.cs (2задача)** |
| **using** System;  **using** **static** System.Console;  **using** System.Collections.Generic;    **namespace** task2  {  **class** Program  {  **static** **void** Main(**string**[] args)  {  Abstr\_Library library = new Checker();  Reader reader1 = new Reader(0, "Marchelino Brynkovian", 16, **true**);  Reader reader2 = new Reader(1, "Alexander Kostyliev", 45, **false**);  Reader reader3 = new Reader(2, "Ekaterina Drobotenko", 18, **true**);  library.GetLibraryBooksList(reader1);  library.GetLibraryBooksList(reader2);  library.GetLibraryBooksList(reader3);  }  }    **class** Reader  {  **private** **int** id;  **public** **int** Id { **get** { **return** id; } **set** { **if** (**value** > 0) { id = **value**; } } }  **public** **string** name;  **private** **int** age;  **public** **int** Age { **get** { **return** age; } **set** { **if** (**value** > 0) { age = **value**; } } }  **public** **bool** hasLogin = **false**;  **public** Reader(**int** id , **string** name, **int** age, **bool** hasLogin)  {  Id = id;  **this**.name = name;  Age = age;  **this**.hasLogin = hasLogin;  }  }    **abstract** **class** Abstr\_Library  {  **public** **abstract** **void** GetLibraryBooksList(Reader reader);  }    **class** Book  {  **public** **int** id;  **public** **string** name;  **public** **string** author;  **public** **int** publishedDate;    **public** Book(**int** id, **string** name, **string** author, **int** publishedDate)  {  **this**.id = id;  **this**.name = name;  **this**.author = author;  **this**.publishedDate = publishedDate;  }    **public** **override** **string** ToString()  {  **return** $"({id}) - '{name}', {author} - {publishedDate}";  }  }    **class** BooksList  {  **public** List<Book> CreateBooksList()  {  List<Book> books = new List<Book>();  books.**Add**(new Book(1, "The Lord of Rings", "J.R.R Tolkien", 1937));  books.**Add**(new Book(2, "Witch 2", "Tom Cruise", 2021));  books.**Add**(new Book(3, "Kateryna", "Taras Shevchenko", 1865));  books.**Add**(new Book(4, "The Great Gatsby", "F. Scott Fitzerald", 1925));  books.**Add**(new Book(5, "The Godfather", "Mario Puzo", 1995));  books.**Add**(new Book(6, "The Black Swan", "Nassim Nicholas Taleb", 2000));  books.**Add**(new Book(7, "Ivanhoe", "Walter Scott", 1819));  books.**Add**(new Book(8, "Im Westen nichts Neues", "Erich Maria Remarque", 1937));  books.**Add**(new Book(9, "Human, All Too Human", "Friedrich Nietzsche", 1887));  books.**Add**(new Book(10, "1984", "George Orwell", 1847));  **return** books;  }  }    **class** Library : Abstr\_Library  {  **public** **override** **void** GetLibraryBooksList(Reader reader)  {  WriteLine($"Hi, {reader.name}!**\n**Welcome to our library :)");  WriteLine("Enter 'search' to choose a book.");  List<Book> books = new BooksList().CreateBooksList();  **if** (ReadLine() == "search")  {  **foreach** (Book book **in** books)  {  WriteLine(book.ToString());  }  WriteLine("Which one would you like to read?");  **int** id = 0;  **if** (**int**.TryParse(ReadLine(), **out** id) && id >= 1 && id <= books.Count)  {  WriteLine($"Excellent choise! Have a nice day");  }  **else**  {  WriteLine($"Oops, we don't have any books with this number! **\n**Please, try again.");  }  }  **else**  {  WriteLine("**\n**Ooops, unknown command!**\n**Please, try again.");  }  }  }    **class** Checker : Abstr\_Library  {  Library library = new Library();  **public** **override** **void** GetLibraryBooksList(Reader reader)  {  **if** (reader.hasLogin)  {  library.GetLibraryBooksList(reader);  }  **else**  {  WriteLine($"**\n**Oops, user {reader.name}, you are not logged in!**\n**Please, log in and try again.**\n**");  }  }  }  } |

**Приклади роботи програм**

|  |
| --- |
| **Приклади роботи першої задачі** |
|  |

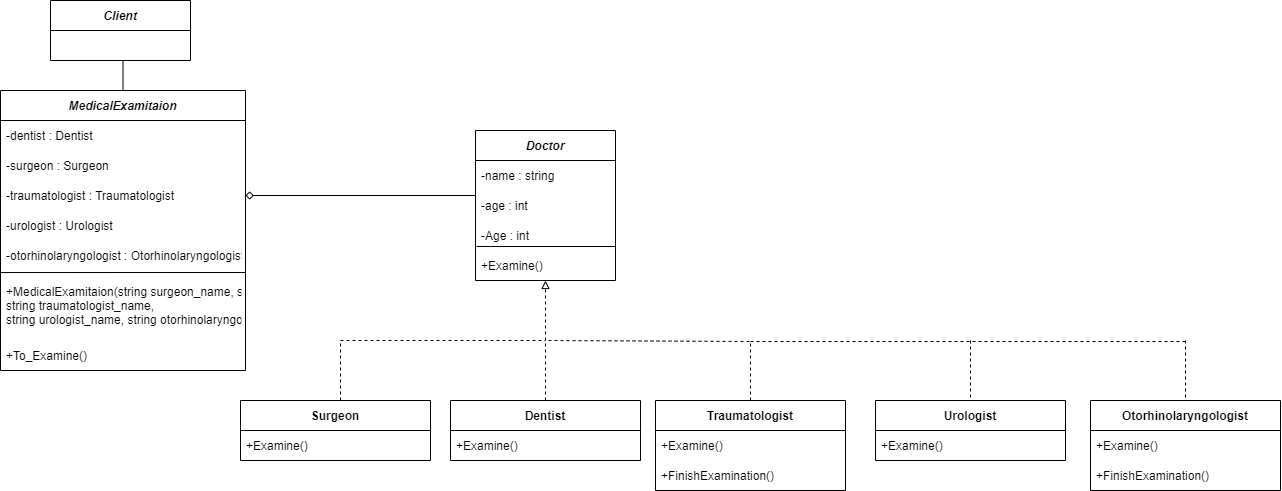
|  |
| --- |
| **Приклади роботи другої задачі** |
|  |

**Обгрунтування використання шаблонів**

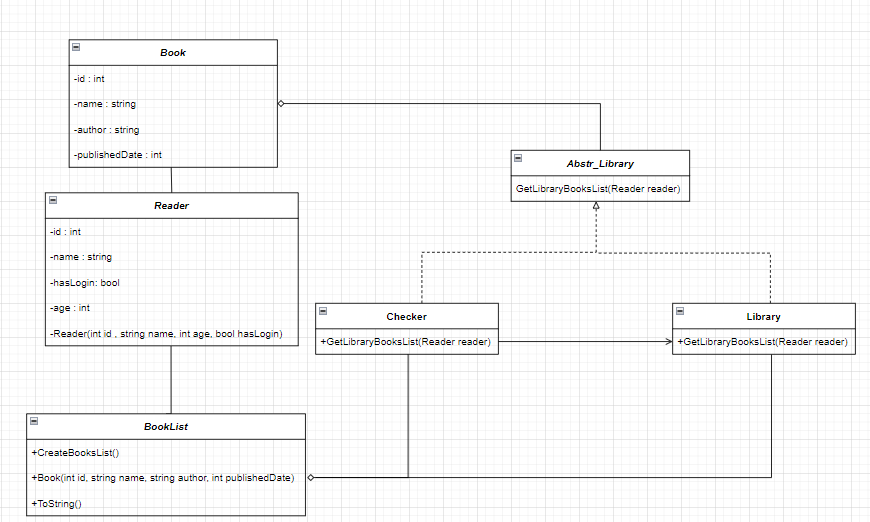
1. Вибір шаблону Facade обумовлюється тим, що він дозволяє приховати складність системи шляхом зведення всіх можливих зовнішніх викликів одного об'єкту, делегує їх відповідним об'єктам системи.

2. Вибір шаблону Proxy обумовлюється тим, що він надає можливість створити такий об'єкт, який контролює доступ до іншого об'єкта, перехоплюючи всі виклики

**UML-діаграми до задач**

**Задача 1:** ****

**Задача 2:**



**Висновки**

В ході лабораторної роботи я покращив знання про структурні шаблони проектування. Навчився використовувати шаблони проектування в залежності від умов задачі.