Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**КУРСОВА РОБОТА**

з дисципліни «Об’єктно-орієнтоване програмування»

на тему

**Шаблони проектування в ООП. Система обліку лікарні**

Виконав студент

ІІ курсу групи КП-61

Щербина Вадим Олегович

Керівник роботи

доц., к.т.н. Заболотня Т.М.

Оцінка

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата, підпис)

Київ-2018

**ЗМІСТ**

**ВСТУП**.........................................................................................................................3

1. **ОПИС СТРУКТУРНО-ЛОГІЧНОЇ ОРГАНІЗАЦІЇ ПРОГРАМИ**.........5
   1. Модульна організація програми............................................................5
   2. Функціональні характеристики.............................................................7
   3. Опис реалізованих класів.......................................................................8
2. **ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ЗА ДОПОМОГОЮ ШАБЛОНІВ ПРОЕКТУВАННЯ**................................................................22
   1. Обґрунтування вибору та опис шаблонів проектування для реалізації програмного забезпечення автомату..................................22
   2. Діаграма класів......................................................................................29
   3. Результати роботи програми................................................................30

**ВИСНОВКИ**..............................................................................................................35

**СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ**......................................................36

**ВСТУП**

Дана курсова робота присвячена розробці програмного забезпечення системи обліку лікарні за допомогою використання шаблонів проектування. Дана програма призначена для зберігання інформації про лікарів, пацієнтів лікарні, і прийоми хворих, з можливістю внесення, вибірки і зміни даних, виведення інформації в необхідному форматі. Лікарня - це така організація, яка працює з дуже великим обсягом інформації про співробітників і про пацієнтів. Лікарі завжди повинні стежити за  своїми пацієнтами, і мати інформацію про курс лікування хворих. А керівництву лікарні необхідно мати інформацію про своїх співробітників для зручнішого управління робочим процесом. Для цього потрібне програмне забезпечення, яке зможе забезпечити доступ до всіх цих даних, які будуть зберігатись в спільній базі даних, що включає всю необхідну інформацію. Дана тематика вибрана для курсової роботи тому, що результати абстрагування об’єктів у даній спроектованій системі дозволяють застосувати всі вивчені методи та принципи об’єктно-орієнтованого програмування для створення програмного забезпечення, зокрема мати змогу правильно організувати код за допомогою шаблонів проектування.

**Об’єктом** курсової роботи є *процес роботи лікарні*.

**Метою** роботи є *розроблення програмного забезпечення* *системи обліку щодо пацієнтів та лікарів* використанням шаблонів проектування.

Для досягнення описаної мети необхідно виконати такі **пункти завдань**:

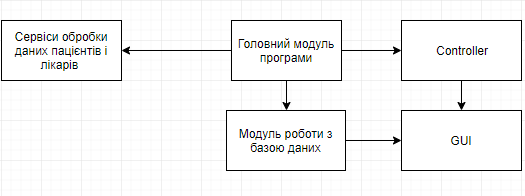
* Абстрагувати об’єкти предметної галузі;
* Розробити структурну організацію програмного забезпечення за допомогою використання основних принципів об’єктно-орієнтованого програмування та шаблонів проектування;
* Визначити та описати функціональні характеристики програми;
* Обґрунтувати вибір шаблонів проектування;
* Розробити графічний інтерфейс користувача;
* Виконати реалізацію програмного забезпечення відповідно до технічного завдання;
* Виконати тестування розробленої програми;
* Оформити документацію з курсової роботи.

Розроблене програмне забезпечення складається з таких логічних частин: інтерфейсу користувача, модуля, що містить інформацію про всіх лікарів, і такого ж модуля, що містить інформацію про всіх пацієнтів, модуля роботи з базою даних.

Використані шаблони проектування: **Стратегія, Компонувальник, Декоратор, Одинак, Фабричний метод, Команда, Спостерігач, Прототип, Заступник.**

Пояснювальна записка складається зі вступу, двох розділів, загальних висновків та списку використаних джерел (*три* найменування). Робота містить 38 рисунків. Загальний обсяг роботи – 30 друкованих сторінок.

1. **ОПИС СТРУКТУРНО-ЛОГІЧНОЇ СХЕМИ ПРОГРАМИ**
   1. **Модульна організація програми**



*Рис. 1.1.1. Модульна організація програми*

**Головний модуль системи** – модуль, який містить у собі основні сутності: лікарів, пацієнтів, сервіси. Встановлює взаємодію з модулями Controller i модулем форм. Цей модуль координує роботу всієї програми. Залежно від того, хто увійшов у систему, головний модуль визначає можливості користувача для роботи із даними.

**Модуль Controller** – модуль відповідає за взаємодію між головним модулем і модулем форм, він за наявністю необхідного запиту з певними параметрами, виконує дії над всіма об’єктами головного модулю.

**Модуль роботи з базою даних** – база даних взаємодіє безпосередньо з основним модулем, функціями бази даних є зберігання, модифікація даних, видача результатів пошуку в базі даних. Зовні модуль може отримувати запити на пошук, видалення, зміну чи додавання даних та, в свою чергу, видавати результат виконання цих запитів. Цей модуль може переадресовувати запити до тієї чи іншої бази даних.

**Сервіси для роботи лікарні** – модуль, який містить у собі багато реалізації функцій для обробки додавання хвороби пацієнту або створення ліків в залежності від того, якою хворобою хворіє пацієнт.

**GUI** –модуль відповідає за інтерфейс користувача. Відображає певні об’єкти, з якими хоче працювати користувач, та посилає потрібні запити до Controller, коли необхідно доступитись до даних.

* 1. **Функціональні характеристики**

Робота з програмою розпочинається з *Main Menu* (головного меню). Користувач може вибрати зайти в систему в залежності від того, яку роль має користувач коли він авторизується він зайде в систему і може мати такі ролі(головний лікар, лікар, пацієнт), також є можливість реєстрації пацієнтів, перед тим як іти на прийом до лікаря пацієнт повинен обов’язково зареєструватись в системі.

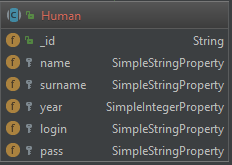
Якщо користувач зайшов, як головний лікар, він має можливість добавляти нових лікарів на роботу, звільняти їх і редагувати інформацію про них.

Якщо користувач зайшов в систему, як лікар він має можливість шукати пацієнтів за їхніми ПІБ або іншими параметрами. Він може добавити пацієнта собі на лікування, після добавлення пацієнта він буде відображатись в таблиці, в якій будуть показанні всі його дані, далі лікар може поставити діагноз пацієнту і в залежності від діагнозу в пацієнта буде зменшуватись здоров’я, також він може виписати ліки, які дозволять пацієнту виздоровіти.

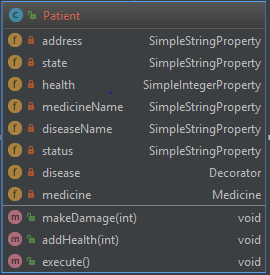
Якщо користувач зайшов систему як пацієнт він може редагувати свої дані, і переглянути вміст своєї картки в якій будуть показанні всі зміни.

* 1. **Опис реалізованих класів**

1. **Клас Human**

Абстрактний клас Human. Є батьківським класом для класів Patient і Doctor. Містить основні поля. . Є класом “Leaf” для шаблону Команда.

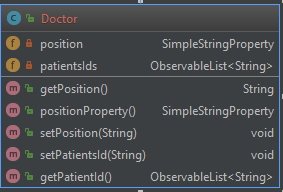
1. **Клас Patient**

Клас, що наслідується від Human представляє собою конкретного пацієнта. Є класом “Invoker” для шаблону Команда.

*-Методи*

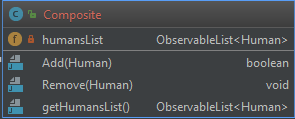
–makeDamage – метод, який викликається коли в пацієнта є якась хвороба і зменшує здоров’я пацієнта до 0, якщо не застосувати ліки.

­­–addHealth – метод, який викликається коли пацієнту назначають певне лікарство, додає здоров’я до 100.

1. **Клас Doctor**

Клас, що наслідується від Human представляє собою конкретного лікаря. Містить в собі список пацієнтів, яких він лікує.

**4) Клас Composite**

Клас наслідується від абстрактного класу Human і від інтерфейсу InterfComposite. Клас є “Composite” для шаблону Компонувальник, містить в собі список об’єктів які належать до типу Human.

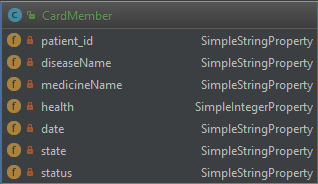
*Методи*

–Add – додає об’єкт типу Human до списку.

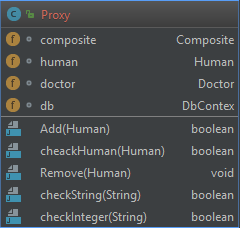
–Remove – видаляє об’єкт з списку.

–getHumansList- повертає список об’єктів типу Human.

**5) Клас CardMember**

Клас містить поля, які є основними полями для перегляду стану пацієнту. Дані, які містить цей клас зберігаються в базу даних і потім відображаються пацієнту.

**6) Proxy**

Клас наслідується від інтерфейсу InterfComposite. Містить в собі поля composite і human. Даний клас є “Proxy” для шаблону Заступник.

*Методи*

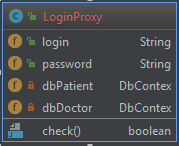
-checkString – перевіряє строку чи вона може бути іменем або прізвище.

-checkInteger – перевіряє строку чи вона є валідною і відповідає року народження.

-checkHuman­ – перевіряє об’єкт типу Human, на те чи є він валідним чи ні.

-Add­ – перевіряє чи даний об’єкт є валідним і якщо так, то добавляє його в список, який міститься в composite і записує дані в базу даних.

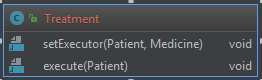
**7) Клас LoginProxy**

Клас містить в собі поля login і dbPatient, dbDoctor. Він потрібен для перевірки логіна користувача.

*Методи*

–check – метод, який перевіряє, чи в базі даних вже є користувач з даним логіном, якщо є то цей метод повертає false , якщо немає то true.

**8) Клас Treatment**

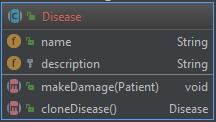
Даний клас є “Receiver” для шаблона Команда.

*Методи*

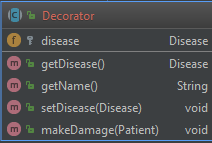
-setExecutor – метод який призначає певному пацієнту ліки.

-execute – метод який викликає метод, який відповадає за збільшення здоров’я пацієнта.

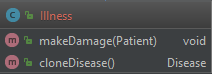
**9) Клас Disease**

Абстрактний клас є батьківським для Decorator. Містить основні дані про конкретну хворобу. Є “Component” для шаблона Декоратор.

**10) Клас Decorator**

Абстрактний клас є батьківським для конкретних декораторів.  Містить в собі певну хворобу. Є “Decorator” для шаблона Декоратор.

**11) Клас Illness**

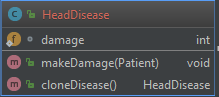
Клас наслідується від абстрактного класу Disease і реалізує метод makeDamage. Є “ConcreteComponent” для шаблона Декоратор і “ConcretePrototype” для шаблона Прототип.

*Методи*

-makeDamage - знімає здоров’я в конкретного пацієнта.(шаблон Декоратор)

-cloneDisease – клонує об’єкт даного класу(шаблон Прототип)

**12) Клас HeadDisease**

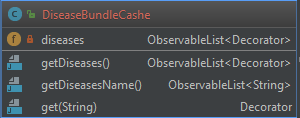
Клас наслідується від абстрактного класу Disease і реалізує метод makeDamage. Є “ConcreteDecorator” для шаблона декоратор і “ConcretePrototype” для шаблона Прототип.

*Методи*

-makeDamage - знімає здоров’я в конкретного пацієнта. шаблон Декоратор)

-cloneDisease – клонує об’єкт даного класу. (шаблон Прототип)

**13) Клас DiseaseBundleCash**

Клас, який містить в собі список хвороб. Він замість створення хвороб клонує їх.

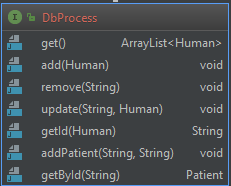
*Методи*

-getDiseases – повертає список всіх хвороб, клонуючи їх.

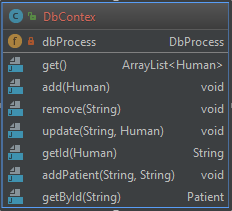
-getDiseasesName – повертає назви всіх хвороб.

-get – повертає хворобу з певною назвою.

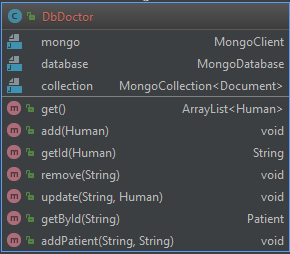
**14) Інтерфейс DbProcess**

Клас задає загальний інтерфейс отримання даних з бази даних. Є “Strategy” для шаблону Стратегія.

**15) Клас DbContex**

Клас, який містить об’єкт конкретної стратегії і містить методи, які визивають методи в конкретної стратегії реалізовані по своєму. Є “Context” для шаблона Стратегія.

**16) Клас DbDoctor**

Даний клас є “ConcreteStrategy” для шаблону Стратегія. Він має доступ до бази даних в якій зберігаються всі дані про лікарів.

*Методи*

- get – повертає список лікарів, які зберігаються в базі даних.

- add ­– добавляє нового лікаря до бази даних

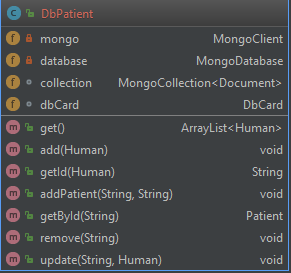
-getId – повертає id(унікальний ідентифікатор) певного лікаря в базі даних

- remove – видаляє певного лікаря з бази даних

-update ­– обновляє інформацію про певного Лікаря в базі даних

-getById – повертає лікаря за його id(унікальний ідентифікатор)

- addPatient – добавляє пацієнта до певного лікаря

**17) Клас DbPatient**

Даний клас є “ConcreteStrategy” для шаблону Стратегія. Він має доступ до бази даних в якій зберігаються всі дані про лікарів.

*Методи*

- get – повертає список лікарів, які зберігаються в базі даних.

- add ­– добавляє нового лікаря до бази даних

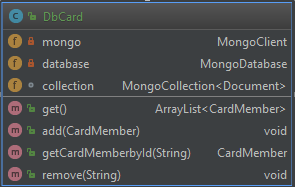
-getId – повертає id(унікальний ідентифікатор) певного лікаря в базі даних

- remove – видаляє певного лікаря з бази даних

-update ­– обновляє інформацію про певного Лікаря в базі даних

-getById – повертає лікаря за його id(унікальний ідентифікатор)

**18) Клас DbCard**

Даний клас має доступ до бази даних в якій зберігаються всі дані про стан пацієнта.

*Методи*

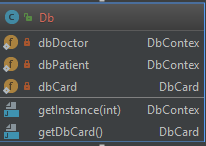
- get – повертає список всіх записів, які зберігаються в базі даних.

- add ­– добавляє новий запис до бази даних

- getCardMemberbyId – повертає останній запис в карткі для певного пацієнта.

- remove – видаляє з бази даних запис в карткі.

**19) Клас Db**

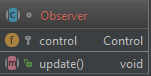
Даний клас реалізовує шаблон “Одинак”. Він надає доступ до різних таблиць бази даних.

*Методи*

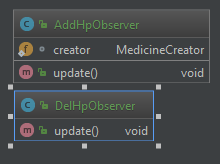
*-*getInstance – в залежності від того, що приймає цей метод він повертає об’єкт, який надає доступ до бази даних пацієнтів або лікарів.

- getDbCard – метод,я кий повертає об’єкт, який надає доступ до бази даних в якй містяться записи в картку всіх пацієнтів.

**20) Observer**

Даний клас є інтерфейсом для створення певних слухачів, які будуть слідкувати за змінами об’єкту. “Subscriber” шаблона Наглядач.

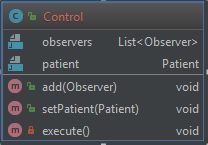
**21) Класи AddHpObserver і DelHpObserver**

Дані класи є реалізацією інтерфейсу Observer вони є певними слухачами, які будуть слідкувати за змінами об’єкту. “ConcreteSubscriber” шаблона Наглядач. 

*Методи*

Методи даних класів добавляють і зменшують здоров’я пацієнта відповідно.

**22) Клас Control**

Даний клас містить список слухачів і коли стається якась подія він повідомляє їх. І кожен слухач оброблює цю подію по своєму.

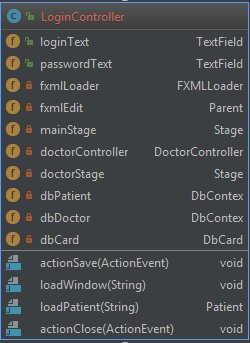
*Методи*

- add – додає певного слухача до списку.

- setPatient – встановлює пацієнта для якого потрібно зробити певну дію.

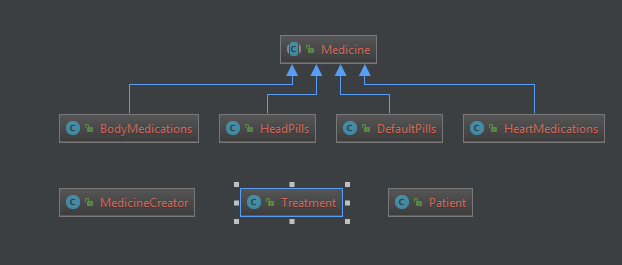
- execute – проходиться по всіх слухачах і викликає їхній метод, який обробить дані.

**23) Класи контролери(DoctorController, PatientController, MainMenuController, AddPatientController, MaindocController, LoginController)**

Дані класи є контролерами, в них відбувається взаємодія GUI і класів, які були описанні вище, вони відповідають за відображення інтерфейсу користувачам.

*Методи*

Методи даних класів визначають поведінку програми при виконані певних дій в інтерфейсі, який надається користувачу.

1. **ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ЗА ДОПОМОГОЮ ШАБЛОНІВ ПРОЕКТУВАННЯ**
   1. **Обґрунтування вибору та опис шаблонів проектування для реалізації програмного забезпечення автомату**
2. **Factory Method і Comand**

*Рис. 2.1.1. UML-діаграма шаблону “****Factory Method і Comand****”*

**Factory Method**

*Породжуючий шаблон*. Визначає інтерфейс для створення об'єктів, але дозволяє підкласам змінювати тип створюваних об'єктів.

*Структура*. **MedicineCreator** – головний інтерфейс даного шаблону. Має у середині один-єдиний метод: створення ліків. Цей інтерфейс реалізовують чотири інших класи(HeadPills, HeartMedications, DefaultPills, BodyMedications). Ці чотири класи і представляють весь спектр наявних ліків.

*Обґрунтування використання даного шаблону*. За допомогою даного шаблона можна організувати гнучку реалізацію створення ліків від певної хвороби, в залежності від того яка хвороба **MedicineCreator** буде створювати потрібні ліки.

**Comand**

*Поведінковий шаблон*. Перетворює операцію в об'єкт. Це дозволяє передавати операції як аргументи при викликах методів, ставити операції в чергу, логувати їх, а також підтримувати можливість скасування операцій.

*Структура*.Реалізований для класу **Patient**. Об’єкт цього класу має у середині об’єкт команди(Medicine), і цей об’єкт застосовується для лікування пацієнта, в залежності від того які ліки він має його здоров’я буде збільшуватись по-різному. Кожен об’єкт класів(HeadPills, HeartMedications, DefaultPills, BodyMedications) має метод, який добавляє здоров’я пацієнту по-різному.

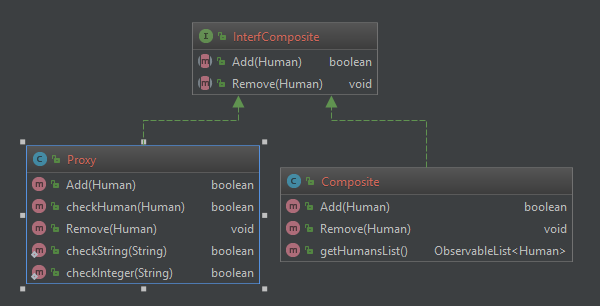
*Обґрунтування використання даного шаблону*. Даний шаблон було використано, щоб параметризувати об’єкт пацієнт дією, яка буде його лікувати в залежності від того, які ліки він приймає.

Учасники:

* Medicine – загальний інтерфейс для створення класів-команд, які є ліками
* HeadPills, HeartMedications, DefaultPills, BodyMedications – конкретні реалізації даного інтерфейсу
* Treatment – клас, який назначає певному пацієнту ліки
* MedicineCreator – клас, який створює ліки в залежності від того, якою хворобою хворіє пацієнт.

Результати використання конструкції :

* Створюються ліки в залежності від того, якою хворобою хворіє пацієнт
* Кожному пацієнту назначаються ліки, які збільшують його здоров’я

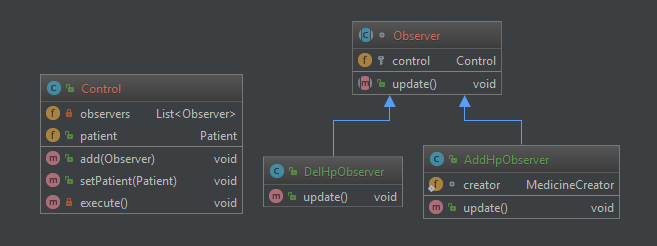
1. **Proxy**

*Рис. 2.1.2. UML-діаграма шаблону “****Proxy****”*

*Структурний шаблон*. Обертає корисний об'єкт або сервіс спеціальним об'єктом-замінником, який “прикидається» оригіналом і перехоплює виклики до нього, а потім, після деякої обробки, направляє їх до обгорнутого об'єкту.

*Структура*. **Proxy** вміщає у собі об’єкти класу **Human і Composite**. Метод *Add* має у середині додаткову перевірку перед виповненням основної логіки, і вона буде виповнена тільки тоді, коли **Proxy** проведе необхідні перевірки(а саме буде перевірено чи валідні дані буде містити об’єкт типу Human перед тим як додати його до списку, який міститься в **Composite**).

*Обґрунтування використання даного шаблону*.Лікарі і пацієнти мають прізвище, фамілію, рік народження, перед тим як додавати дані об’єкти і записувати їх до бази даних доцільно буде перевірити всі дані на валідність, тому слід реалізувати структуру, описану у даному шаблоні.

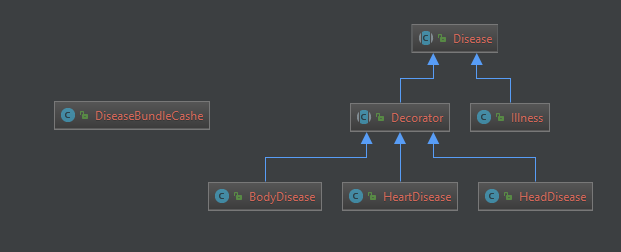
1. **Observer**

*Рис. 2.1.3. UML-діаграма шаблону “****Observer****”*

*Поведінковий шаблон*. Створює механізм підписки, за допомогою якого одні об'єкти можуть підписуватися на оновлення, що відбуваються в інших об'єктах.

*Структура*. **Control** – основний компонент системи, який має у середині структуру даних: підписників на деякі події. Також, у середині реалізовані стандартні методи, визначені даним шаблоном. **DelHpObserver** (віднімає здоров’я в пацієнта)і **AddHpObserver** (добавляє здоров’я пацієнту) – конкретні слухачі подій, кожен з яких реагує на події по-своєму.

*Обґрунтування використання даного шаблону*.При виконанні логіки, зв’язаної з назначенням пацієнту певного діагнозу або назначення ліків, відбуваються події, які повинні обробити слухачі, якщо в пацієнта поставлений діагноз і неназначенні ліки його здоров’я повине зменшуватись, якщо ліки назначенні то здоров’я повине збільшуатись. Даний шаблон дозволяє реалізувати таку можливість.

1. **Decorator + Prototype**

*Рис. 2.1.4. UML-діаграма шаблону “****Decorator + Prototype****”*

**Декоратор**

*Структурний шаблон*. Динамічно додає об'єктам нову функціональність, обертаючи їх в корисні “обгортки”.

*Структура*. Для створення хвороб використано шаблон Декоратор, який для кожної хвороби перевизначає метод makeDamage. В залежності від того якого типу хвороба вона буде зменшувати здоров’я пацієнта по-своєму. Ми кожну створену хворобу декоруємо нашими декораторами, і вона буде відноситись до певного типу(Body, Head, Heart).

*Обґрунтування використання даного шаблону*.Після створення класів, хвороб, з’явилася потреба віднести ці хвороби до певних типів, які б по-різному завдавали шкоду пацієнту, але мали один і той же абстрактний клас як фундамент. За допомогою “Декоратора” таке розширення функціоналу вдалося досягти, не змінюючи існуючого коду.

**Прототип**

Прототип – шаблон, що задає види створюваних об’єктів за допомогою екземпляра-прототипу і створює нові об’єкти шляхом копіювання цього прототипу. Є доцільним коли нам часто потрібно створювати однакові об’єкти.

*Структура.* Оскільки, об’єкти-хвороби не змінюють свого стану – нам вигідно застосовути цей шаблон. Він буде містити всі хвороби, які вже є декорованими(BodyDisease, HeadDisease, HeartDisease )і відносяться до певного типу. Коли лікар поставсить діагноз, то хвороба не буде створюватись заново, а скопієються з кешу(DiseaseBundleCashe.)

*Обґрунтування використання даного шаблону*.Даний шаблон дає змогу зекономити на створенні об’єктів типу хвороби, адже лікар за день може поставити безліч діагнозів і нам не потрібне повторне створення багатьох однакових об’єктів.

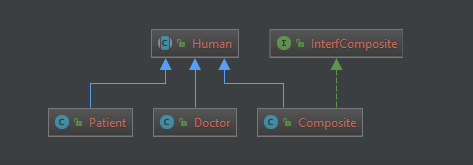
Учасники:

* Disease – оголошує інтерфейс для операції для зменшення здоров’я пацієнта
* Decorator ­– абстрактний клас який містить в собі об’єкт типу Disease і визначає метод, який зменшує здоров’я пацієнта
* Illness – загальний клас хвороби, який ми будемо декорувати, є об’єктом копіювання
* BodyDisease, HeadDisease, HeartDisease – конкретні декоратори, які перевизначають наш метод, вони також є об’єктами для клонування.
* DiseaseBundleCashe – клас, який містить створенні об’єкти наших декорованих хвороб і ми можемо взяти їх звідти клонуванням.

Результати використання конструкції :

* Декоратор відносить хвороби до певних типів
* виконуючи дії з абстрактними об’єктами-хворобами, клієнт

клонує їх використовуючи відповідну операцію.

1. **Компонувальник**

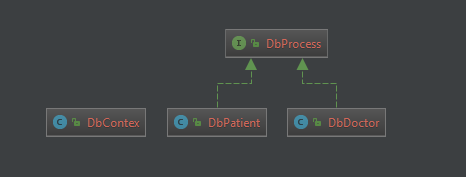
*Рис. 2.1.5. UML-діаграма шаблону “****Компонувальник****”*

*Cтруктурний шаблон* проектування, який призначений для створення деревоподібної структури для подання ієрархії об’єктів, де кожний об’єкт можна розглядати незалежно або як набір вкладених об’єктів через один інтерфейс. Таким чином, з’являється можливість уніфіковано, однаково поводитися з кожним об’єктом.

*Структура*. Клас **Composite** має у собі список об’єктів які наслідуються від абстрактного класу Human (Patient, Doctor). Клас Composite має методи add і remove, які дозволяють добавляти і видаляти об’єкти. Результат ми набагато спрощуємо структуру нашої програми, запобігаючи створенню зайвих класів які б містили список лікарів або пацієнтів, а так в нас є один клас в якому містяться об’єкти типу Human.

*Обґрунтування використання даного шаблону*. Даний шаблон доцільно використати при побудові програми через те, що в нас є ієрархічна структура. Оскільки головний лікар буде містити список лікарів, а кожен лікар буде містити в собі список пацієнтів, що дає змогу легкого і швидкого доступу до залежних об’єктів. Даний шаблон значно спрощує програму, адже нам не потрібно окремо створювати список лікарів або пацієнтів.

1. **Стратегія**

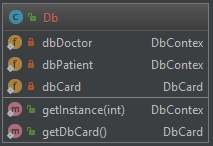


*Рис. 2.1.4. UML-діаграма шаблону “****Стратегія****”*

*Поведінковий шаблон.* Його суть полягає у тому, щоб створити декілька схем поведінки для одного об'єкту та винести в окремий клас. Шаблон Стратегія(Strategy) дозволяє міняти вибраний алгоритм незалежно від [об'єктів-клієнтів,](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)) які його використовують.

*Структура.* Шаблон відповідає за надання користувачу інтерфейсу доступу до бази данних в якій зберігаються об’єкти, які реалізують абстрактний клас Human, тобто для лікаря і пацієнта потрібна різна стратегія отримання даних з бази і спільний інтерфейс який дозволить це зробити(DbConntex).

*Обґрунтування використання даного шаблону*. Даний шаблон було використано, щоб винести реалізацію отримання з бази даних інформації в окремі класи в залежності від того який об’єкт нам потрібно отримувати ми створюємо(DbDoctor або DbPatient). І в нас є клас (DbConntex) який надає інтерфейс для взаємодії з даними класами.

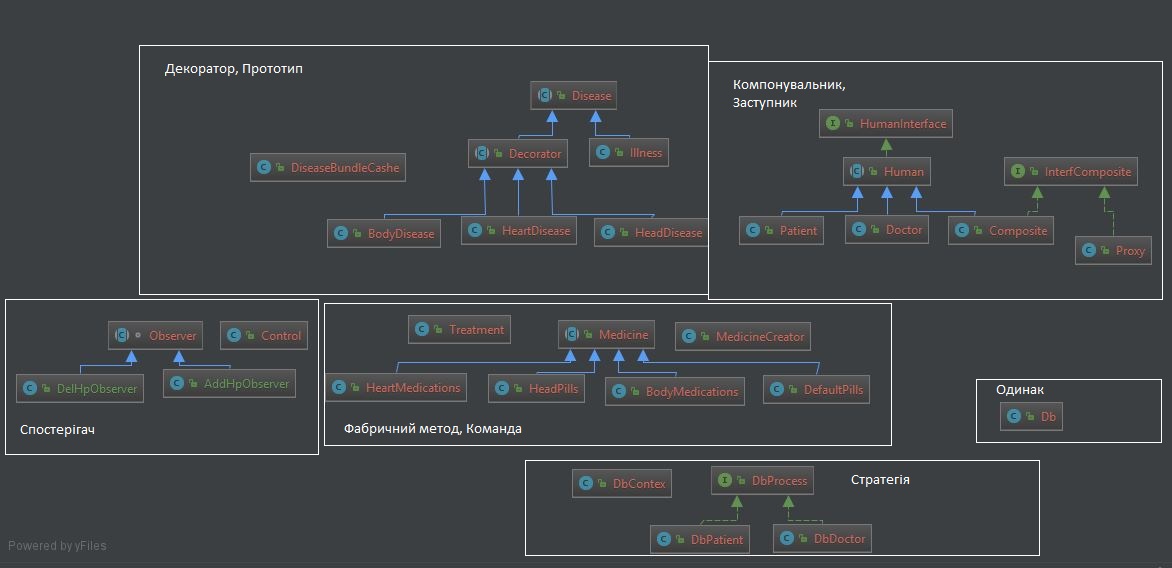
1. **Одинак**

*Породжуючий шаблон*, який гарантує, що існує тільки один екземпляр певного класу і надає до нього глобальну точку доступу

*Структура.* Даний шаблон складається з одного класу в якому містяться об’єкти, які надають доступ до бази даних.

*Обґрунтування використання даного шаблону*. Оскільки ми багато де використовуємо об’єкти доступу до баз даних, то нам вигідно створити такий клас, який містить вже всі ці об’єкти і ми можемо з будь-якої точки нашої програми доступитись до них.

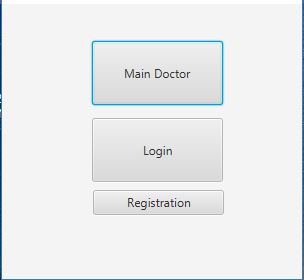
* 1. **Діаграма класів**



*Рис. 2.2.1. Діаграма класів програми*

* 1. **Результати роботи програми**

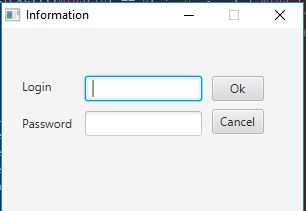
Початок роботи з магазином техніки для дому починається з головного меню:



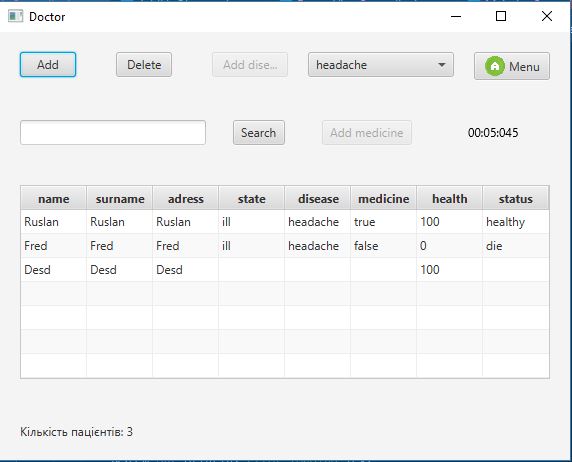
*Рис. 2.3.1. Головне меню*

**Пропонується вибрати роль під якою ви бажаєте зайти в систему(Головний лікар, лікар, пацієнт).Після входу в систему для кожної ролі буде відображатись свій інтерфейс.

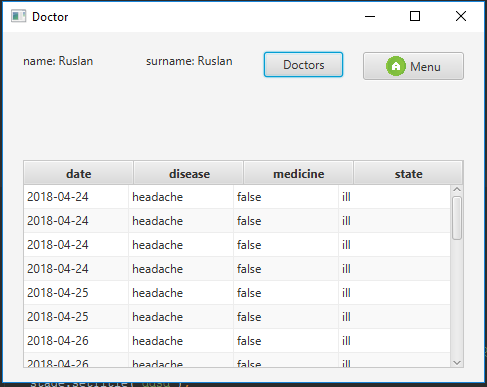
*Рис 2.3.2 Інтерфейс форми роботи з даними головного лікаря*

 *Рис 2.3.3 Форма для входу*

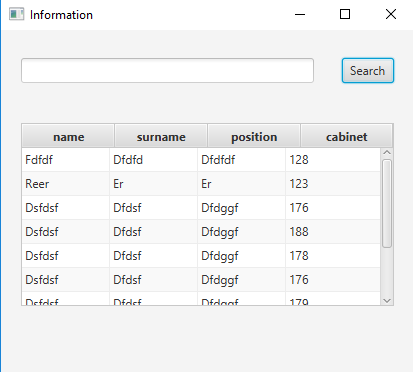
Ввівши логін і пароль в форму користувач увійде в систему в залежності від того яку роль він має(лікар або пацієнт).

*Рис 2.3.4 Інтерфейс форми роботи з даними лікаря*

Якщо користувач зайшов в систему, як лікар він має можливість шукати пацієнтів за їхніми ПІБ або іншими параметрами. Він може добавити пацієнта собі на лікування нажавши на кнопку Add, після добавлення пацієнта він буде відображатись в таблиці, в якій будуть показанні всі його дані(name, surname, address, state, disease, medicine, health, status), далі лікар може поставити діагноз пацієнту вибравши з списку хворобу нажавши на кнопку addDisease і в залежності від діагнозу в пацієнта буде зменшуватись здоров’я, також він може виписати ліки нажавши на кнопку Add medicine, які дозволять пацієнту виздоровіти, кожних 5 секунд здоров’я пацієнта буде обновлятись.



*Рис 2.3.4 Інтерфейс форми роботи з даними пацієнта*

Якщо користувач зайшов систему як пацієнт він може редагувати свої дані, і переглянути вміст своєї картки в якій будуть показанні всі зміни, може знайти потрібного йому лікаря.

*Рис 2.3.4 Форма пошуку пацієнта*

Кожен лікар може шукати пацієнтів і додавати їх собі на лікування, пошук здійснюється за певними критеріями.

**ВИСНОВКИ**

Метою даної курсової роботи було розроблення програмного забезпечення для системи обліку лікарні з використанням шаблонів проектування. Підставою для розроблення стало завдання на виконання курсової роботи з дисципліни «Об’єктно-орієнтоване програмування» студентами ІІ курсу кафедри ПЗКС НТУУ «КПІ».

Для досягнення поставленої мети у повному обсязі виконано завдання, визначені у аркуші завдання на курсову роботу; розроблено графічні матеріали; реалізовано всі вимоги до програмного продукту, наведені у технічному завданні; створено відповідну документацію. Розроблене програмне забезпечення дозволяє будь-якому лікареві вести облік своїх пацієнтів, ставити їм діагнози, переглядати поточний стан їхнього здоров’я, пацієнти ж можуть переглядати історію своєї картки, а головний лікар може приймати лікарів на роботу.

Програму створено на основі використання шаблонів проектування. Зокрема, до структури програмного забезпечення входить реалізація семи шаблонів, які належать до різних груп.

Для розроблення програмного забезпечення була використана мова програмування Java.

**СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ**

1. Язык шаблонов. Города. Здания. Строительство. / Кристофер Вольфганг Александер. – 1977. – 1096 с.
2. Приёмы объектно-ориентированного проектирования. Паттерны проектирования / Эрих Гамма, Ричард Хелм, Ральф Джонсон, Джон Влиссидес. – 1994. – 395 с.
3. Руководство Microsoft по проектированию архитектуры приложений. / С. Сомасегар, Скотт Гатри, Девид Хилл. – 2009. – 529 с.