МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

„КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. І. Сікорського”

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

## КУРСОВА РОБОТА

з дисципліни «Програмування»   
на тему

## Шаблони проєктування в ООП. Програмне забезпечення ремонту квартир під ключ

Виконала студентка

ІІ курсу групи КП-03   
Сітайло Анна Сергіївна залікова книжка КП-0318

Керівник роботи

доцент, к.т.н. Заболотня Т.М. Оцінка

(дата, підпис)

КИЇВ 2022

## ЗМІСТ

**ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ ТА ТЕРМІНІВ** 3

## ВСТУП 4

## СТРУКТУРНО-АЛГОРИТМІЧНА ОРГАНІЗАЦІЯ ПРОГРАМНОЇ СИСТЕМИ ПІДТРИМКИ РЕМОНТУ КВАРТИР 6

* 1. Модульна організація програми 6
  2. Функціональні характеристики 8
  3. Опис реалізованих класів 10

## ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ЗА ДОПОМОГОЮ ШАБЛОНІВ ПРОЄКТУВАННЯ 29

* 1. Обґрунтування вибору та опис шаблонів проєктування для

програмної реалізації ремонту квартир під ключ 29

* 1. Діаграма класів 39
  2. Опис результатів роботи програми 41

## ВИСНОВКИ 48

## СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 49

## ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ ТА ТЕРМІНІВ

|  |  |
| --- | --- |
| ООП - | модель програмування, яка базується на ствердженні того, що програма – це сукупність об’єктів, які взаємодіють між собою |
| Web - | (англ. World Wide Web) глобальний інформаційний простір, заснований на фізичній інфраструктурі Інтернету і протоколі передачі даних HTTP |
| БД - | база даних - організована структура, яка призначена для зберігання, зміни та обробки взаємозалежної інформації, переважно великих обсягів |
| ПЗ - | програмне забезпечення - набір кодованих інструкцій для керування процесором комп’ютера |

## ВСТУП

Дана курсова робота присвячена розробці програмного забезпечення ремонту приміщень під ключ за допомогою використання шаблонів проєктування. Ремонт приміщень завжди залишається актуальним, адже власники квартир хочуть оновити інтер’єр своїх приміщень, власники нових офісів бажають створити умови, придатні для роботи та ділових зустрічей, школи та дитсадки намагаються покращувати свої приміщення з метою забезпечення комфорту для своїх учнів. Прикладів можна навести безліч. Тим не менш, процес вибору інтер’єру, матеріалів, меблів забирає велику кількість часу, тому власники приміщень потребують певний метод, який допоможе дистанційно вирішувати питання, пов’язані з ремонтом. Саме завдяки створеному програмному забезпеченню користувачі (власники) зможуть зберегти свій час: спостерігати за процесом виконання ремонтних робіт і вирішувати питання щодо дизайну приміщень дистанційно. Дана тематика обрана для виконання курсової роботи тому, що результати абстрагування об’єктів у цій предметній галузі дозволяють застосувати вивчені принципи та методи об’єктно‑орієнтованого програмування для створення програмного забезпечення, зокрема шаблони проєктування.

*Об’єктом* дослідження є відслідковування та керування процесом ремонтних робіт у приміщеннях.

*Метою роботи* є розроблення програмного забезпечення підтримки ремонту приміщень під ключ з використанням шаблонів проєктування.

Для досягнення визначеної мети необхідно виконати такі *завдання*:

* абстрагувати об’єкти предметної галузі;
* розробити структурну організацію ПЗ за допомогою застосування основних принципів ООП та шаблонів проєктування;
* визначити та описати функціональні характеристики програми;
* обґрунтувати вибір шаблонів проєктування, використаних для побудови програми;
* розробити дизайн інтерфейсу користувача;
* виконати реалізацію програмного забезпечення відповідно до вимог технічного завдання;
* виконати тестування розробленої програми;
* оформити документацію з курсової роботи.

Розроблене ПЗ ремонту приміщень під ключ складається з декількох модулів: модуль, який взаємодіє з базою даних, модуль, який представляє собою сторінку клієнта та модуль, який забезпечує взаємодію користувача з даними його квартири.

Реалізовані шаблони проєктування: Заступник, Стан, Будівельник, Ланцюжок обов’язків та Абстрактна фабрика.

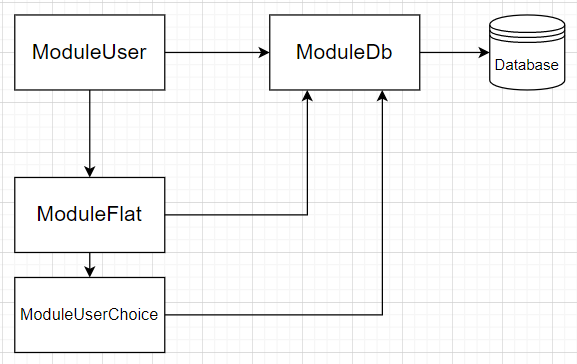
До функціональних можливостей програми належать: реєстрація та вхід у систему, перевірка правильності вхідних даних (при реєстрації, вході в систему, виборі інтер’єру тощо), надання користувачу можливості редагувати дані про своє приміщення (при виборі інтер’єру, підлоги, стін та меблів відповідь користувача записується в базу даних), перегляд загальних даних про приміщення.

Для функціонування розробленої програми необхідно забезпечити наявність на комп’ютері 20 Мб вільного дискового простору.

Розроблене програмне забезпечення може бути використане у компаніях, які займаються ремонтом приміщень під ключ та націлені на потреби своїх клієнтів.

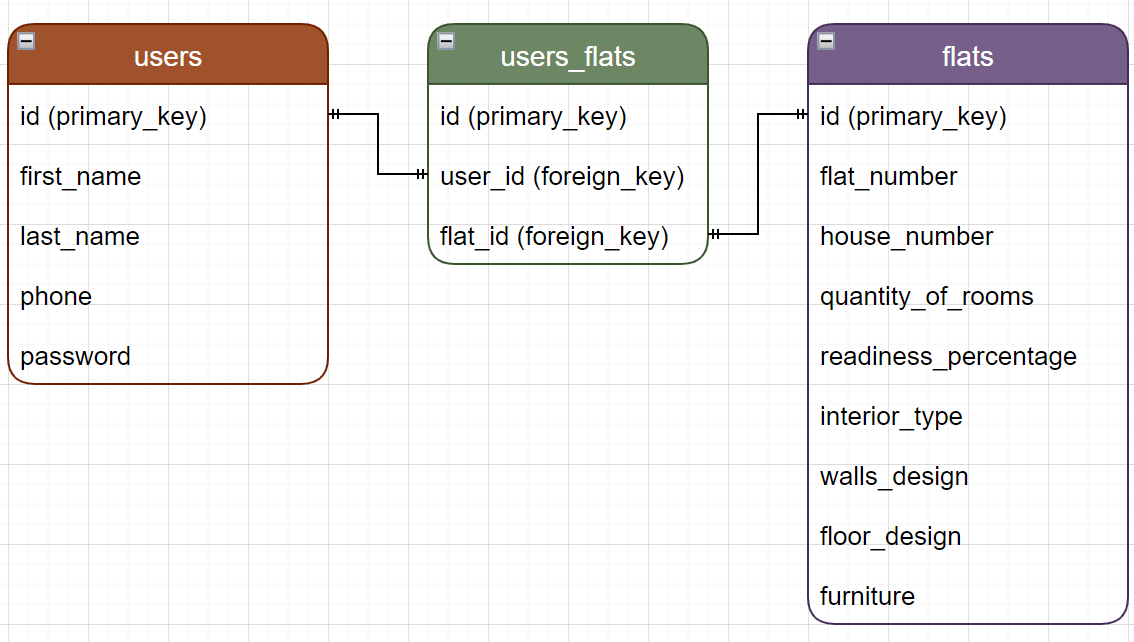
Пояснювальна записка складається зі вступу, трьох розділів, загальних висновків, списку використаних джерел (6-ти найменувань, з них 2 – іноземною мовою). Робота містить 27 рисунків. Загальний обсяг роботи – 49 друкованих сторінок, з них 44 сторінки основного тексту та 1 сторінка списку використаних джерел.

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



*Рис.1.1.1. Модульна організація програми*

Програма має три умовних програмних модулі, які обмінюються між собою певними даними. Серед них ModuleDb, ModuleUser та ModuleFlat (модуль, який взаємодіє з бд, модуль, який представляє собою сторінку клієнта та модуль, який забезпечує взаємодію користувача з даними його квартири). Зв’язок та робота модулів одне з одним відбувається за допомогою виклику тих чи інших класів та методів.



*Рис.1.1.2. Структура БД*

Одним із головних модулів є ModuleDb, який взаємодіє з базою даних. У перелік його функцій входить збереження, зміна та пошук записів у таблицях бази даних. Модуль отримує запити на збереження нових записів, зміни й пошук вже існуючих даних та видає результат виконання цих запитів. Він взаємодіє з іншими двома модулями, згаданими вище.

Модуль ModuleUser відповідає за вхід та реєстрацію користувача. Він містить у собі перевірку введених даних та надає користувачу доступ до основного функціоналу програми. ModuleUser пов’язвний з ModuleDb, оскільки для реєстрації користувача або ж перевірки його наявності в системі, модуль бази даних створює запит до бази даних для додавання або ж пошуку відповідного запису.

Модуль ModuleFlat забезпечує взаємодію користувача з даними його квартири. Користувач має змогу переходити до вибору інтер’єру, дизайну стін, підлоги, виробника меблів. ModuleFlat має підмодуль ModuleUserChoice, який вже надає користувачу можливість робити вибір тієї чи іншої характеристики квартири. Клас, що міститься в ModuleFlat оперує класами та методами ModuleUserChoice за допомогою посилань на необхідні елементи підмодуля та отримує відповідь користувача як результат виконання викликаних методів. Окрім цього, отримані результати ModuleFlat надсилає в ModuleDb. Там створюється запит до бази даних для зміни записів відповідної квартири на ті, які були обрані користувачем.

* 1. **Функціональні характеристики**

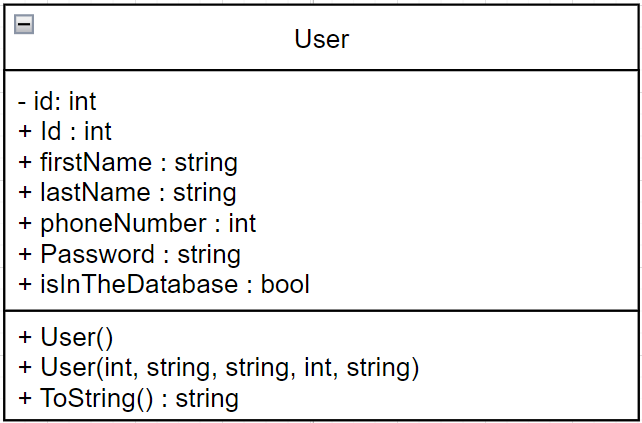
Перш за все користувачу (власнику квартири) необхідно ввійти в систему. Для цього користувач вводить прізвище, ім’я та пароль, які обробляються та перевіряються на коректність. У випадку неправильності вводу, у консолі з’являється повідомлення про помилку. Якщо формат даних правильний, то виконується перевірка на наявність такого користувача в системі. Якщо користувача за введеною інформацією знайдено, то в консолі відображаються його дані та надається коротка інструкція переходу до даних про квартиру. У тому випадку, коли користувач не міститься в базі даних, у консоль виводиться повідомлення про помилку. Аналогічно відбувається і реєстрація в системі, але тоді при наявності запису із введеною інформацією в базі даних в консоль виводиться помилка, а при відсутності запису – повідомлення про успішну реєстрацію.

Коли користувач зміг потрапити на свою «сторінку» (в консолі зображені його дані), він має можливість перейти на «сторінку» своєї квартири (в консолі зображені дані про його квартиру та інструкція для взаємодії з цими даними). Залежно від відсотків готовності квартири власник повинен робити вибір інтер’єру/стін/підлоги/виробника меблів. На кожному етапі користувачу надається коротка інструкція як перейти до певної сторінки, зробити вибір або повернутися назад.

При виборі інтер’єру квартири в консоль виводиться список стилів та їх характеристика. Після чого користувач може обрати один з них або повернутися назад до сторінки квартири. Якщо власник квартири вводить некоректні дані, він отримує повідомлення про помилку. Після кожного вибору відсоток готовності квартири зростає, а користувач переходить до наступного етапу (вибір нової характеристики). Коли власник квартири обирає оформлення стін або ж підлоги, у консоль по черзі виводяться матеріали та колір, які використовуватимуться для ремонту стін та підлоги відповідно. Як тільки користувач робить вибір, він (аналогічно до вибору інтер’єру) переходить на сторінку своєї квартири та переглядає вже оновлені дані про неї. При виборі виробника меблів користувачу надається список меблів: назва, виробник та ціна – для кожного товару. Власник обирає виробника та повертається на сторінку квартири. У випадку, якщо готовність досягає 100 відсотків, у консоль просто виводиться повідомлення, що квартира готова.

* 1. **Опис реалізованих класів**

***Клас User***

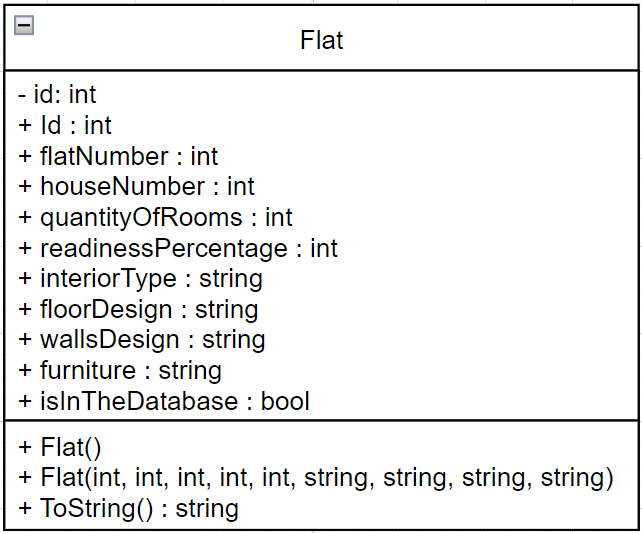
******

Клас, який відповідає за створення користувачів.

*Методи:*

* ToString – описує формат виводу даних про користувача в консоль.

***Клас Flat***

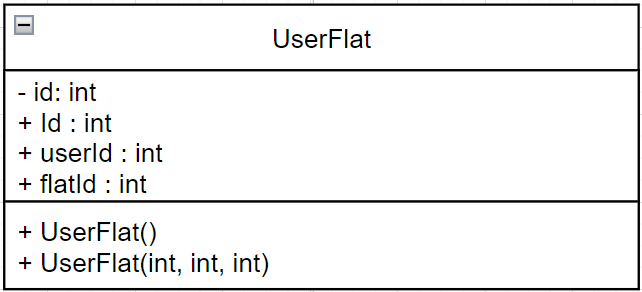
**

Клас, який відповідає за створення квартир.

*Методи:*

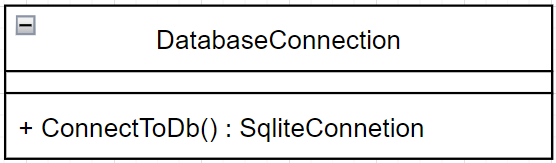
* ToString – описує формат виводу даних про квартиру в консоль.

***Клас Flat***

**

Клас, який відповідає за створення квартир.

***Клас DatabaseConnection***

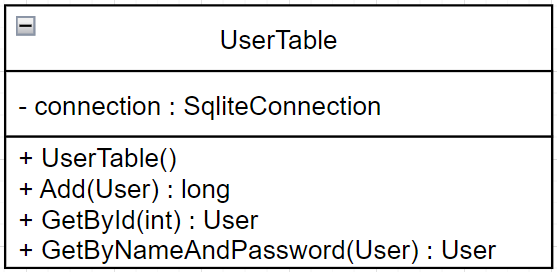
**

Клас підключення до бази даних з таблицями користувачів, квартир та таблиці-зв’язку між користувачами та їх квартирами.

*Методи:*

* ConnectToDb – містить у собі шлях до бази даних та повертає новий зв’язок з нею.

***Клас UserTable***

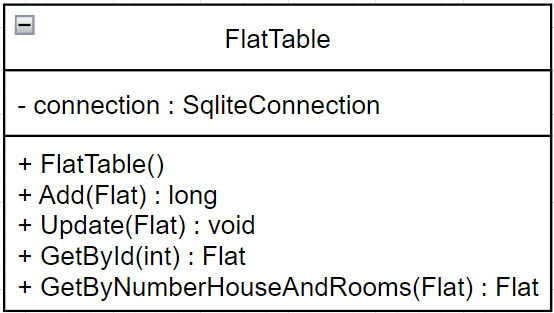
**

Клас взаємодії програми з таблицею користувачів у базі даних. Додає та шукає користувачів за різними полями.

*Методи:*

* Add – приймає дані про користувача, створює sql-запит до бази даних для додавання нового запису, повертає ідентифікатор створеного користувача;
* GetById – приймає ідентифікатор користувача, створює sql-запит до бази даних для пошуку запису в таблиці за ідентифікатором, повертає користувача як результат пошуку за ідентифікатором;
* GetByNameAndPassword – приймає дані про користувача, створює sql-запит до бази даних для пошуку запису в таблиці за повним ім’ям (ім’я та прізвище) та паролем, повертає користувача як результат пошуку за введеними даними.

***Клас FlatTable***

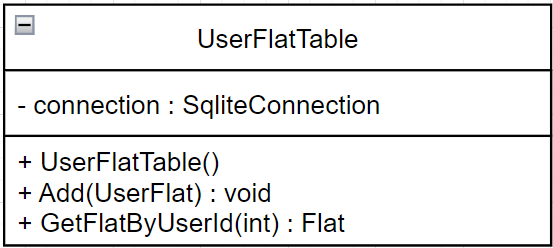
**

Клас взаємодії програми з таблицею квартир у базі даних. Додає, оновлює та шукає квартири за декількома полями.

*Методи:*

* Add – приймає дані про квартиру, створює sql-запит до бази даних для додавання нового запису, повертає ідентифікатор створеної квартири;
* Update – приймає дані про квартиру, створює sql-запит до бази даних для оновлення наявного запису;
* GetById – приймає ідентифікатор квартири, створює sql-запит до бази даних для пошуку запису в таблиці за ідентифікатором, повертає квартиру як результат пошуку за ідентифікатором;
* GetByNumberHouseAndRooms – приймає дані про квартиру, створює sql-запит до бази даних для пошуку запису в таблиці за номером квартири, номером будинку та кількістю кімнат, повертає квартиру як результат пошуку за введеними даними.

***Клас UserFlatTable***

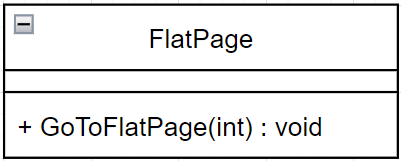
**

Клас взаємодії програми з таблицею-зв’язком між користувачами та квартирами у базі даних. Додає нові записи в таблицю та шукає квартири за ідентифікаторами користувачів (власників відповідних квартир).

*Методи:*

* Add – приймає дані про зв’язок користувача та квартири, створює sql-запит до бази даних для додавання нового запису;
* GetFlatByUserId – приймає ідентифікатор користувача, створює sql-запит до бази даних для пошуку запису в таблиці за ідентифікатором, повертає квартиру як результат пошуку за ідентифікатором її власника.

***Клас FlatPage***

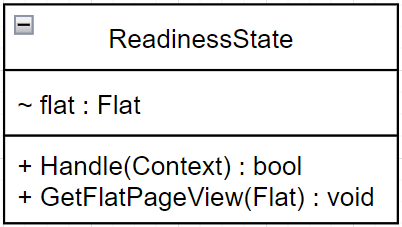
**

Клас, який забезпечує перехід зі сторінки користувача на сторінку квартири, поєднує два модулі програми: ModuleUser та ModuleFlat

*Методи:*

* GoToFlatPage – забезпечує повернення користувача зі сторінки квартири на сторінку її власника.

***Клас*** ***ReadinessState***

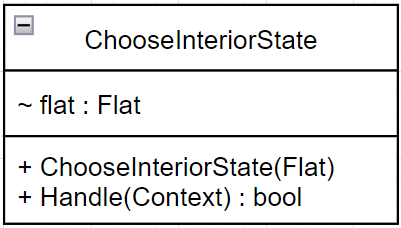
**

Абстрактний клас. Виступає основою дочірніх класів, які у свою чергу обробляють вибір користувача на кожному з етапів готовності квартири.

*Методи:*

* Handle – абстрактний метод, який відповідає за виконання основного функціоналу дочірніх класів, тобто зміни тих чи інших даних про квартиру користувачем (її власником);
* GetFlatPageView – відповідає за відображення даних про квартиру в консолі;

***Клас ChooseInteriorState***

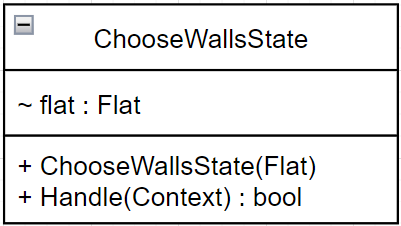
**

Клас, який наслідує ReadinessState, надає користувачу можливість обирати інтер’єр його квартири.

*Методи:*

* Handle – відповідає за перехід користувача на сторінку з вибором інтер’єру його квартири; якщо готовність приміщення більша або дорівнює 25%, то метод одразу направляє користувача на наступний етап вибору.

***Клас ChooseWallsState***

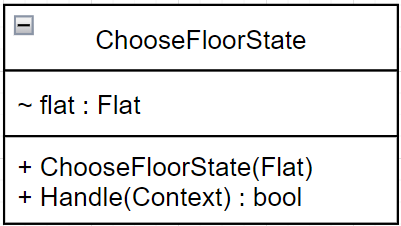
**

Клас, який наслідує ReadinessState, надає користувачу можливість обирати оформлення стін його квартири.

*Методи:*

* Handle – відповідає за перехід користувача на сторінку з вибором дизайну стін його квартири; якщо готовність приміщення більша або дорівнює 50%, то метод одразу направляє користувача на наступний етап вибору.

***Клас ChooseFloorState***

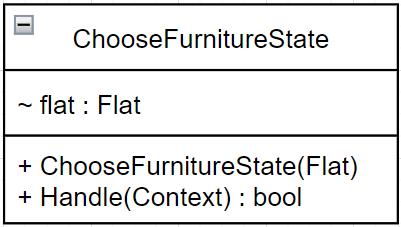
**

Клас, який наслідує ReadinessState, надає користувачу можливість обирати оформлення підлоги його квартири.

*Методи:*

* Handle – відповідає за перехід користувача на сторінку з вибором дизайну підлоги його квартири; якщо готовність приміщення більша або дорівнює 75%, то метод одразу направляє користувача на наступний етап вибору.

***Клас ChooseFurnitureState***

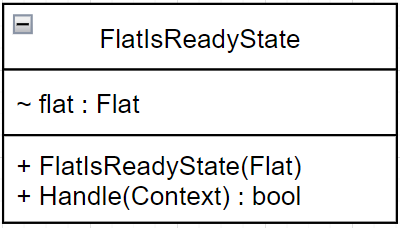
**

Клас, який наслідує ReadinessState, надає користувачу можливість обирати виробника меблів для його квартири.

*Методи:*

* Handle – відповідає за перехід користувача на сторінку з вибором виробників меблів для його квартири; якщо готовність приміщення дорівнює 100%, то метод одразу направляє користувача на наступний етап вибору.

***Клас FlatIsReadyState***

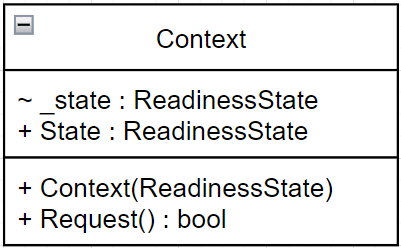
**

Клас, який наслідує ReadinessState, повідомляє про те, що ремонт у кватрирі завершено.

*Методи:*

* Handle – відповідає за вивід повідомлення у консоль про готовність квартири (виконується лише за умови, якщо готовність приміщення дорівнює 100%).

***Клас Context***

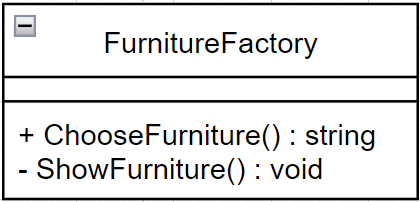
**

Клас, який забезпечує виконання переходу від одного етапу вибору до іншого

*Методи:*

* Request – викликає метод Handle залежно від поточного етапу вибору.

***Клас ChooseFurniture***

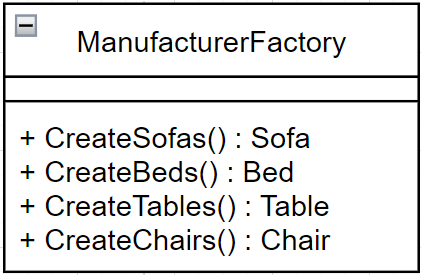
**

Клас, який забезпечує користувачу доступ до перегляду меблів для його квартири. Дозволяє клієнтському коду не прив’язуватися до конкретних класів меблів та працювати з меблями від різних виробників через абстрактні класи.

*Методи:*

* ChooseFurniture – обробляє відповідь користувача та повертає її як результат виконання методу;
* ShowFurniture – виводить у консоль усі меблі, які є в наявності.

***Клас*** ***ManufacturerFactory***

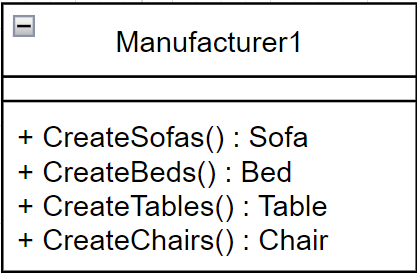
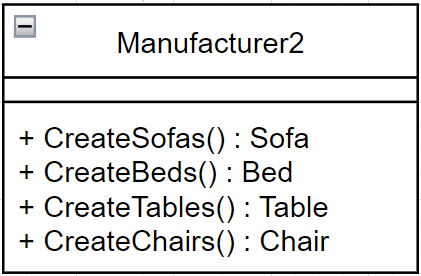
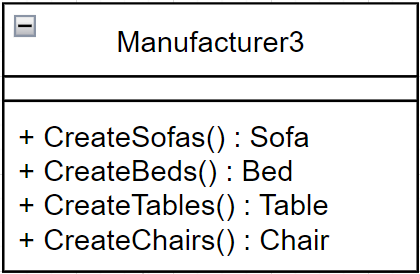
**

Абстрактний клас, який оголошує методи створення таки абстрактних продуктів як диван, ліжко, стіл та стілець.

*Методи:*

* CreateSofas – створює об’єкт типу диван;
* CreateBeds – створює об’єкт типу ліжко;
* CreateTables – створює об’єкт типу стіл;
* CreateChairs – створює об’єкт типу стілець.

***Класи Manufacturer1, Manufacturer2, Manufacturer3***

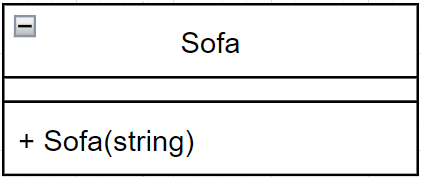
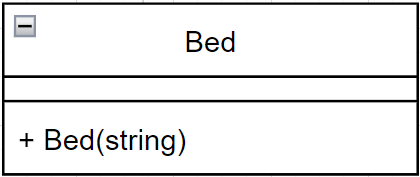
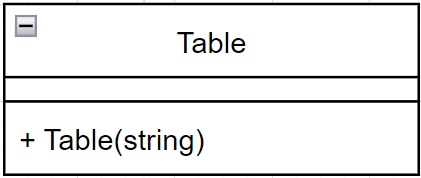
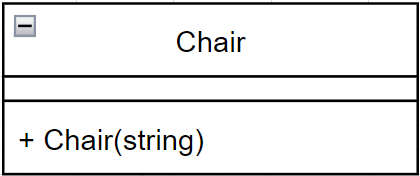
**  

Класи, кожен з яких виступає конкретним виробником (італійським, польським, українським) та реалізує методи абстрактного класу ManufacturerFactory, дозволяючи створювати меблі з конкретними особливостями.

*Методи:*

* CreateSofas – створює опис об’єкту типу диван з властивостями, притаманними конкретно цьому виробнику;
* CreateBeds – створює опис об’єкту типу ліжко з властивостями, притаманними конкретно цьому виробнику;
* CreateTables – створює опис об’єкту типу стіл з властивостями, притаманними конкретно цьому виробнику;
* CreateChairs – створює опис об’єкту типу стілець з властивостями, притаманними конкретно цьому виробнику.

***Класи Sofa, Bed, Table, Chair***

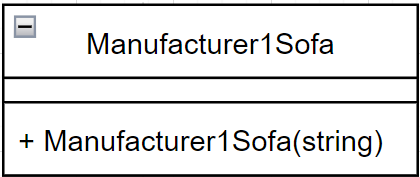
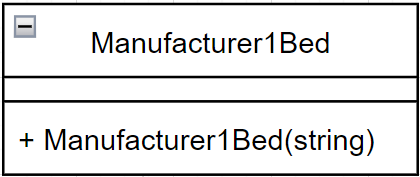
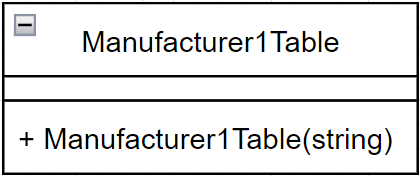
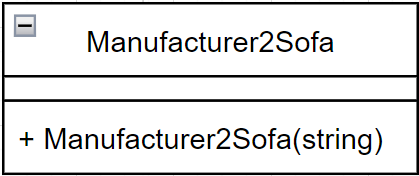
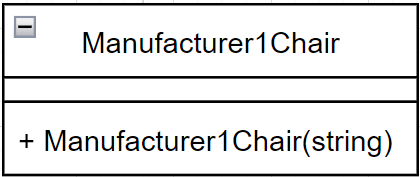
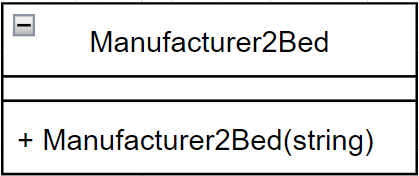
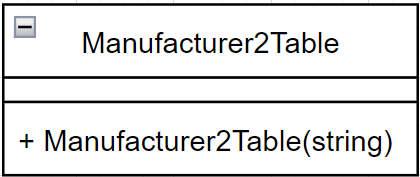
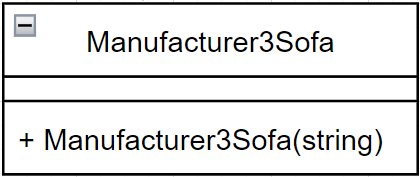
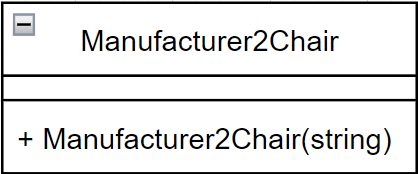
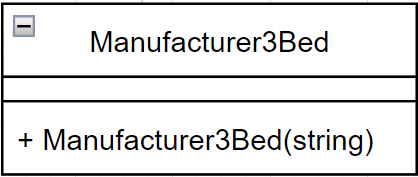
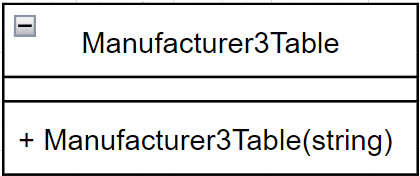
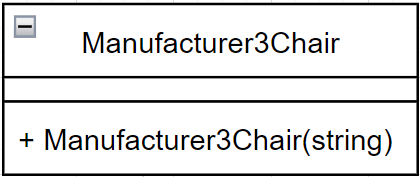
** **  

Абстрактні класи, які оголошують інтерфейси товарів, що пов’язані між собою за сенсом, але виконують різні функції.

*Методи (відповідно стосуються класів Sofa, Bed, Table, Chair):*

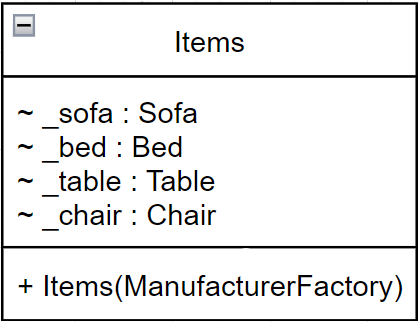
* Sofa – створює об’єкт типу диван;
* Bed – створює об’єкт типу ліжко;
* Table – створює об’єкт типу стіл;
* Chair – створює об’єкт типу стілець.

***Класи*** ***Manufacturer1Sofa, Manufacturer1Bed, Manufacturer1Table, Manufacturer1Chair, Manufacturer2Sofa, Manufacturer2Bed, Manufacturer2Table,*** ***Manufacturer2Chair, Manufacturer3Sofa, Manufacturer3Bed, Manufacturer3Table, Manufacturer3Chair***

**         

Класи, які стосуються різних абстрактних класів (*Sofa/Bed/Table/Chair*) меблів, але мають одні й ті ж властивості, притаманні конкретним виробникам (*Manufacturer1/ Manufacturer2/ Manufacturer3*).

***Клас Items***

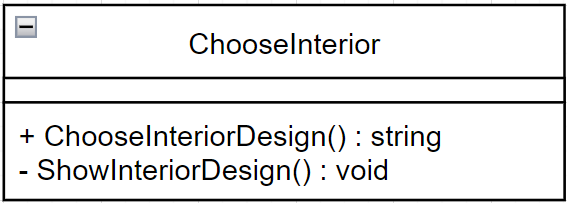
**

Клас, який групує меблі за їх виробником.

*Методи:*

* Items – метод, який приймає виробника та на його основі отримує доступ до тих меблів, які стосуються лише цього конкретного виробника, та збирає їх разом.

***Клас ChooseInterior***

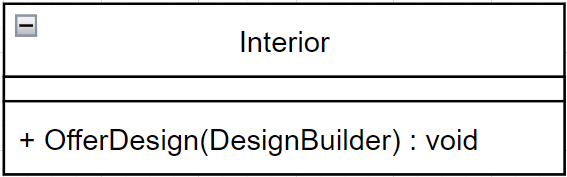
**

Клас, який забезпечує користувачу доступ до перегляду варіантів інтер’єру для його квартири.

*Методи:*

* ChooseInteriorDesign – обробляє відповідь користувача та повертає її як результат виконання методу;
* ShowInteriorDesign – виводить у консоль усі типи інтер’єру, які можна застосувати для квартири користувача.

***Клас Interior***

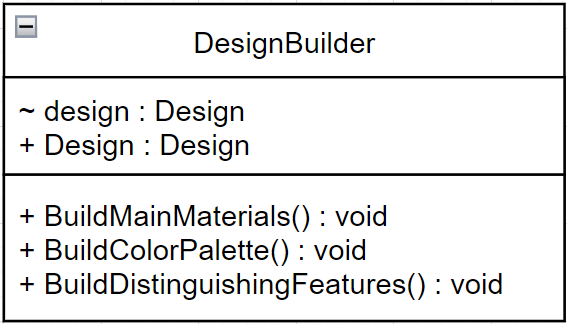
**

Клас, який описує порядок виклику властивостей інтер’єрів та зберігає їх відносно кожного типу інтер’єру.

*Методи:*

* OfferDesign – метод, який приймає конкретний дизайн інтер’єру та викликає його характеристики у заданому порядку.

***Клас DesignBuilder***

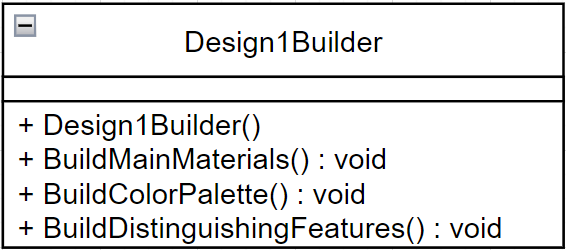
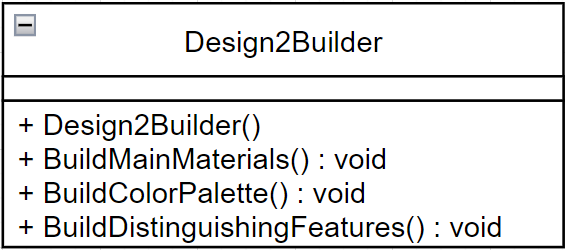
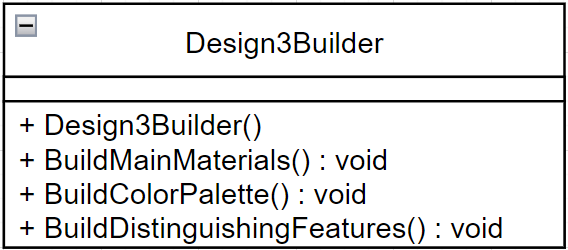
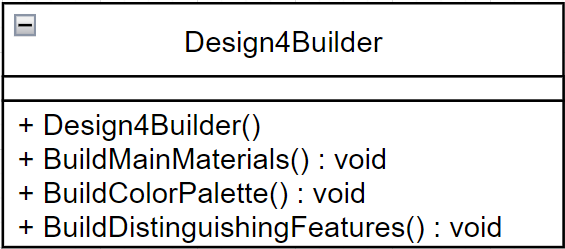
**

Абстрактний клас, який оголошує шляхи конструювання елементів, спільних для всіх наявних типів інтер’єру.

*Методи:*

* BuildMainMaterials – метод, в якому описуються матеріали, присутні в інтер’єрі;
* BuildColorPalette – метод, в якому описується палітра кольорів, присутніх в інтер’єрі;
* BuildDistinguishingFeatures – метод, в якому описуються особливості інтер’єру, властиві саме йому.

***Класи Design1Builder, Design2Builder, Design3Builder, Design4Builder***

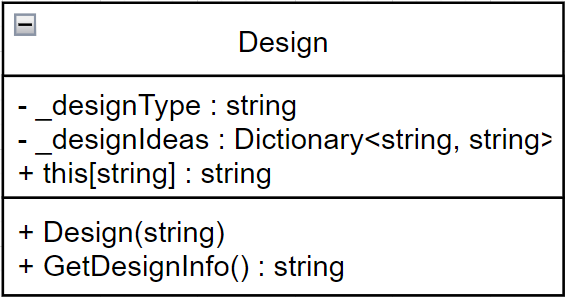
** **  

Класи, які реалізують кроки формування опису конкретного типу інтер’єру по-своєму (з тими особливостями, які приманні конкретно одному з представлених інтер’єрів).

*Методи:*

* BuildMainMaterials – метод, в якому описуються матеріали, присутні в конкретному інтер’єрі;
* BuildColorPalette – метод, в якому описується палітра кольорів, присутніх в конкретному інтер’єрі;
* BuildDistinguishingFeatures – метод, в якому описуються особливості конкретного інтер’єру, властиві саме йому;

***Клас Design***

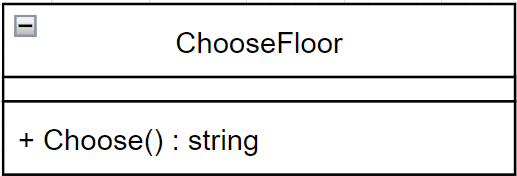
**

Клас, який групує дизайн інтер’єру залежно від його типу.

*Методи:*

* GetDesignInfo – метод, в якому створюється формат виводу характеристик конкретного інтер’єру в консоль.

***Клас ChooseFloor***

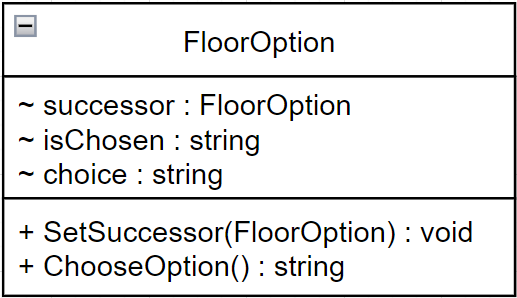
**

Клас, який забезпечує користувачу доступ до перегляду варіантів підлоги для його квартири.

*Методи:*

* Choose – обробляє відповідь користувача та повертає її як результат виконання методу;

***Клас FloorOption***

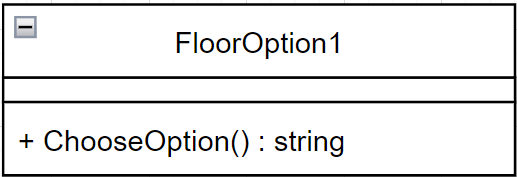
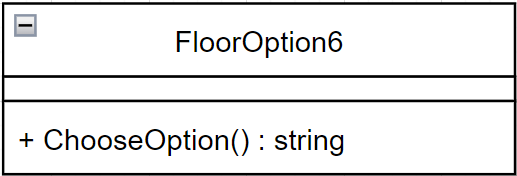
**

Абстрактний клас, який описує метод вибору варіантів підлоги.

*Методи:*

* SetSuccessor – визначає наступний варіант вибору;
* ChooseOption – надає варіант підлоги для квартири, обробляє відповідь користувача та на основі цього або повертає її як результат виконання методу, або направляє до іншого варіанту.

***Класи FloorOption1… FloorOption6***

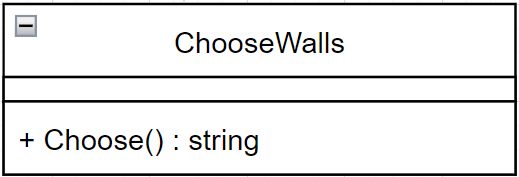
** … **

Класи, які описують методи вибору варіантів підлоги користувачем для своєї квартири.

*Методи:*

* ChooseOption – надає варіант підлоги для квартири, обробляє відповідь користувача та на основі цього або повертає її як результат виконання методу, або направляє до іншого варіанту.

***Клас ChooseWalls***

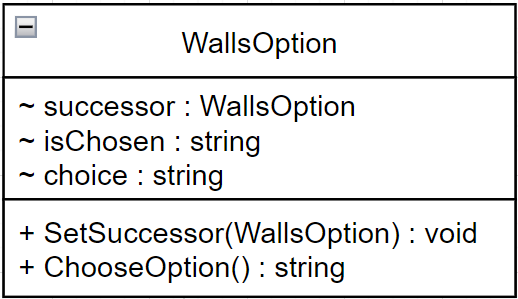
**

Клас, який забезпечує користувачу доступ до перегляду варіантів стін для його квартири.

*Методи:*

* Choose – обробляє відповідь користувача та повертає її як результат виконання методу;

***Клас WallsOption***

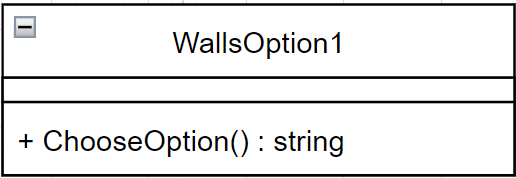
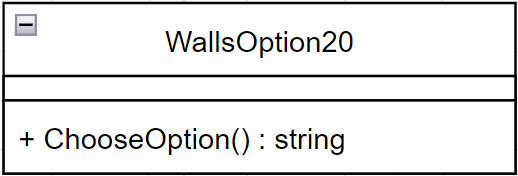
**

Абстрактний клас, який описує метод вибору варіантів стін.

*Методи:*

* SetSuccessor – визначає наступний варіант вибору;
* ChooseOption – надає варіант оформлення стін для квартири, обробляє відповідь користувача та на основі цього або повертає її як результат виконання методу, або направляє до іншого варіанту.

***Класи WallsOption1… WallsOption20***

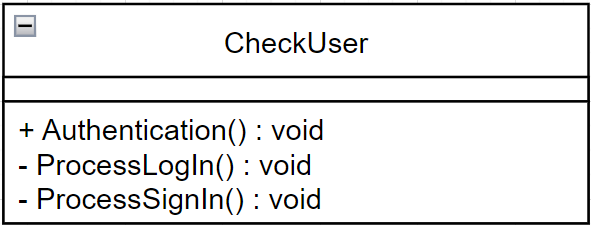
** … **

Класи, які описують методи вибору варіантів стін користувачем для своєї квартири.

*Методи:*

* ChooseOption – надає варіант оформлення стін для квартири, обробляє відповідь користувача та на основі цього або повертає її як результат виконання методу, або направляє до іншого варіанту.

***Клас CheckUser***

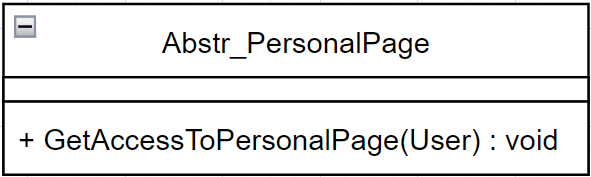
**

Клас, який приймає дані користувача для входу або реєстрації.

*Методи:*

* Authentication – містить коротку інструкцію щодо того, як користувачу зайти у свій профіль або зареєструватися;
* ProcessLogIn – приймає та перевіряє дані, введені користувачем для входу у свій профіль;
* ProcessSignIn – приймає та перевіряє дані, введені користувачем для реєстрації в системі;

***Клас Abstr\_PersonalPage***

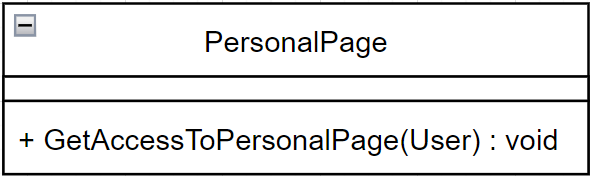
**

Абстрактний клас, який описує логіку для використання класу перевірки там, де очікується клас переходу на сторінку користувача.

*Методи:*

* GetAccessToPersonalPage – описує основну логіку дочірніх класів.

***Клас*** ***PersonalPage***

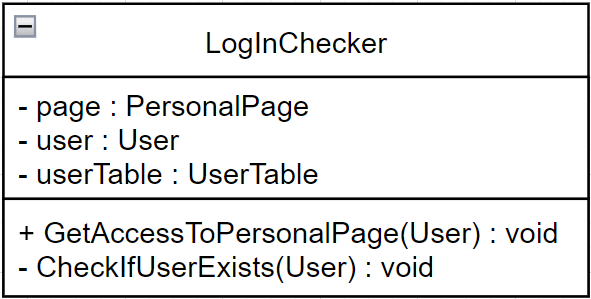
**

Клас, який постає сторінкою користувача, поєднує модулі програми ModuleFlat та ModuleUser.

*Методи:*

* GetAccessToPersonalPage – відповідає за можливість переходу користувача на сторінку його квартири, надає коротку інструкцію щодо можливих подальших дій користувача на сторінці свого профілю.

***Клас LogInChecker***

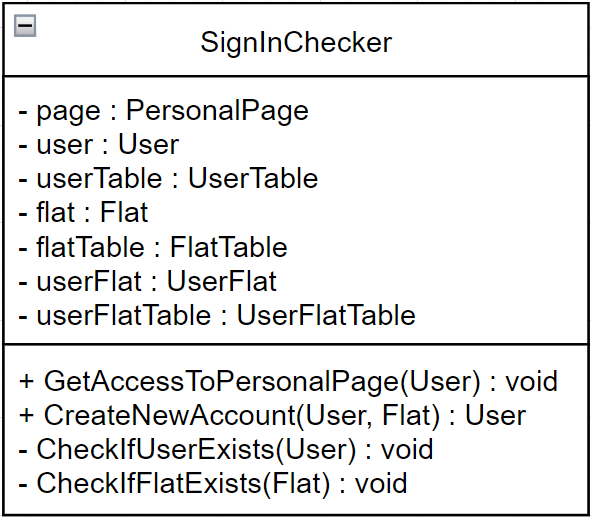
**

Клас, який постає сторінкою користувача, поєднує модулі програми ModuleFlat та ModuleUser.

*Методи:*

* GetAccessToPersonalPage – за тієї умови, що користувач наявний у системі, викликає цей же метод класу PersonalPage; якщо користувача в системі не існує, то в консоль виводиться повідомлення про помилку входу;
* CheckIfUserExists – перевіряє чи міститься користувач у системі та присвоює статус наявності цьому користувачу.

***Клас SignInChecker***

**

Клас, який постає сторінкою користувача, поєднує модулі програми ModuleFlat та ModuleUser.

*Методи:*

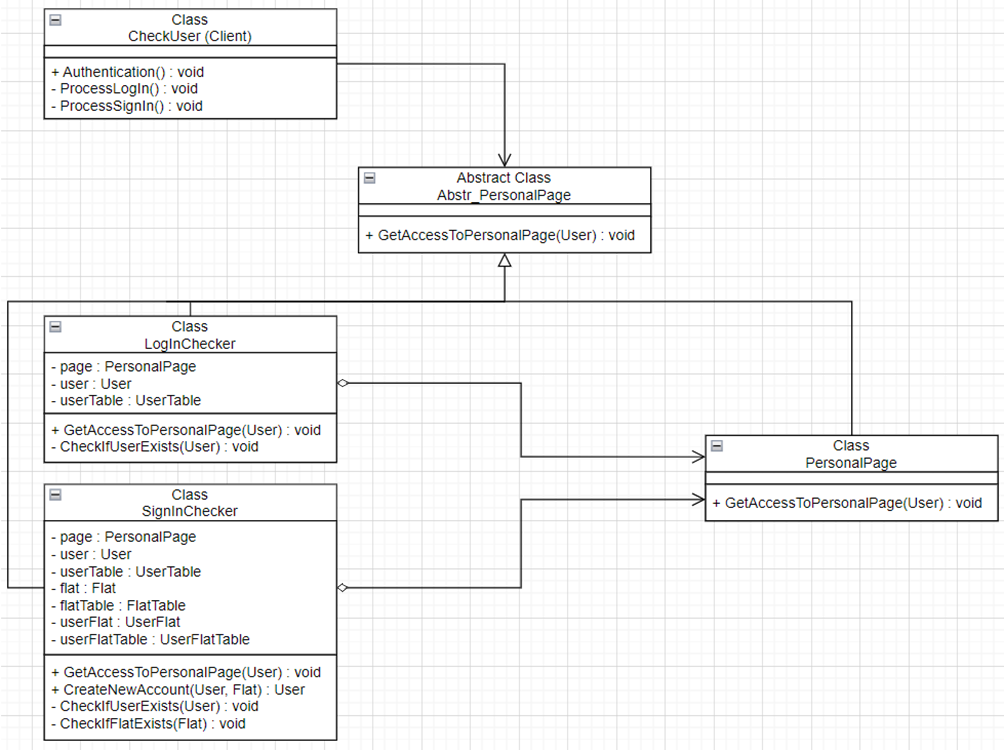
* GetAccessToPersonalPage – за тієї умови, що нового користувача записано в систему, викликає цей же метод класу PersonalPage; якщо користувача в системі не створено, то в консоль виводиться повідомлення про помилку реєстрації;
* CreateNewAccount – відповідає за створення нового запису про користувача та його квартиру в системі (для цього викликається відповідний метод модуля ModuleDb);
* CheckIfUserExists – перевіряє чи міститься користувач у системі та присвоює статус наявності цьому користувачу;
* CheckIfFlatExists – перевіряє чи міститься запис про квартиру в системі та присвоює статус наявності цій квартирі.

1. **ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ЗА ДОПОМОГОЮ ШАБЛОНІВ ПРОЄКТУВАННЯ** 
   1. **Обґрунтування вибору та опис шаблонів проєктування** **для програмної реалізації ремонту квартир під ключ**
2. **Заступник**

Заступник – структурний шаблон проєктування, який забезпечує створення заступника об’єкта для контролю доступу до останнього через перехоплення всіх викликів. Надає об’єкт-заступник або об’єкт-замінник. Обгортаючи доступ до реального компонента, Заступник зменшує складність роботи з ним.

*Учасники шаблону:*

* **Інтерфейс сервісу** (Abstr\_PersonalPage) визначає загальний інтерфейс для сервісу і заступника; завдяки цьому об’єкт заступника можна використовувати там, де очікується об’єкт; визначає ті методи, які описуються в LogInChecker, SignInChecker та PersonalPage;
* **Сервіс** (PersonalPage) – містить корисну бізнес-логіку; PersonalPage надає доступ користувачу до його профілю та можливість взаємодіяти з даними його квартири**;**
* **Заступник** (LogInChecker, SignInChecker) містить посилання на об’єкт сервісу; після того, як заступник закінчує свою роботу, він передає виклики вкладеному сервісу; у нашому випадку обидва заступники перевіряють наявність введених користувачем даних у системі, а потім (якщо необхідні умови виконуються) ніби передає свої повноваження сервісу.



*Рис. 2.1.1. Діаграма класів, які входять до шаблону «Заступник»*

*Результати використання конструкції:*

* Користувач при вході в систему (або реєстрації) вводить свої дані, які перевіряються на коректність та передаються заступнику LogInChecker (або SignInChecker при реєстрації);
* Заступник перевіряє отриману інформацію на наявність у системі; у випадку, що користувач існує, передає дані вкладеному сервісу (при реєстрації якщо записів про користувача і його квартиру не знайдено, то заступник передає дані вкладеному сервісу);
* Сервіс PersonalPage виступає особистою сторінкою користувача і надає йому доступ до основного функціоналу програми.

*Обґрунтування використання шаблону:*

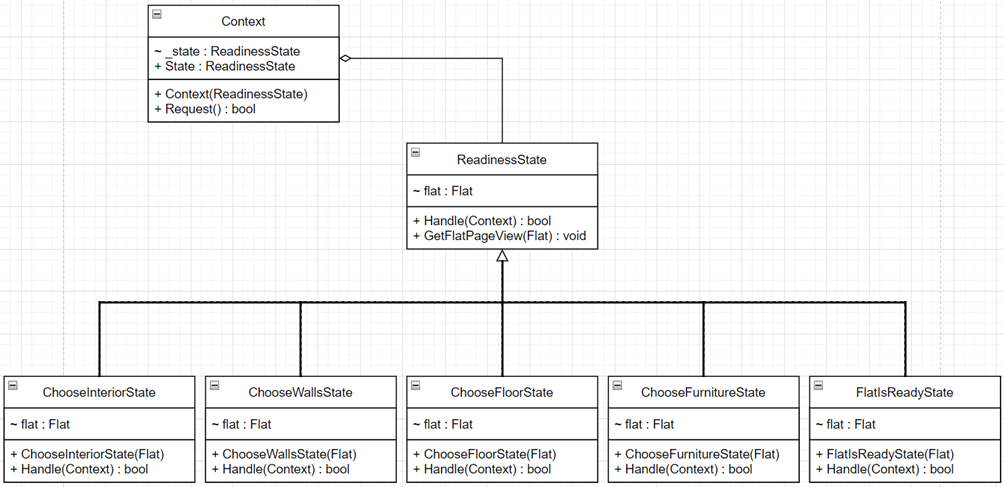
Даний шаблон було використано для того, щоб забезпечити безпечний доступ до особистих даних користувача та його приватної власності. При вході в систему дані користувача перевіряються, щоб надати користувачу доступ саме до його сторінки та даних про квартиру. При реєстрації користувач перевіряється для того, щоб запобігти створенню однакових профілів та одразу надавати новому користувачу доступ до його сторінки. Якби даний шаблон проєктування не використовувався, будь-хто міг би не лише переглядати, а й змінювати дані про квартири та їх власників.

1. **Стан**

Стан – поведінковий шаблон, який дозволяє об’єкту варіювати свою поведінку залежно від внутрішнього стану. Передбачає реалізацію поведінки, асоційованої з певним станом об’єкту, а також забезпечення зміни поведінки відповідно до зміни внутрішнього стану.

*Учасники шаблону:*

* **Контекст** (Context) – зберігає посилання на об’єкт стану і делегує йому частину роботи, яка залежить від стану; контекст працює з цим об’єктом через спільний інтерфейс станів (ReadinessState); має метод для присвоювання йому нового об’єкту-стану;
* **Стан** (ReadinessState) – описує загальний інтерфейс стану; у нашому випадку описує загальні властивості кожного етапу ремонту квартири;
* **Конкретні стани** (ChooseInteriorState, ChooseWallsState, ChooseFloorState, ChooseFurnitureState, FlatIsReadyState) – реалізують поведінку, пов’язану з певним станом контексту; відповідають за обробку вибору користувача та зміну даних про квартиру залежно від конкретного етапу ремонту квартири.

**

*Рис. 2.1.2. Діаграма класів, які входять до шаблону «Стан»*

*Результати використання конструкції:*

* Користувач на сторінці своєї квартири обирає інтер’єр; після того, як він приймає остаточне рішення, його вибір записується до бази даних;
* Коли перший етап ремонту завершено (інтер’єр квартири обрано) і готовність квартири збільшилася на 25%, власник переходить на наступний етап – вибір оформлення стін; аналогічно до першого етапу, користувач переходить да наступного етапу щоразу, як робить вибір;
* Коли готовність ремонту приміщення досягає 100% користувач залишається на одному й тому ж етапі готовності.

*Обґрунтування використання шаблону:*

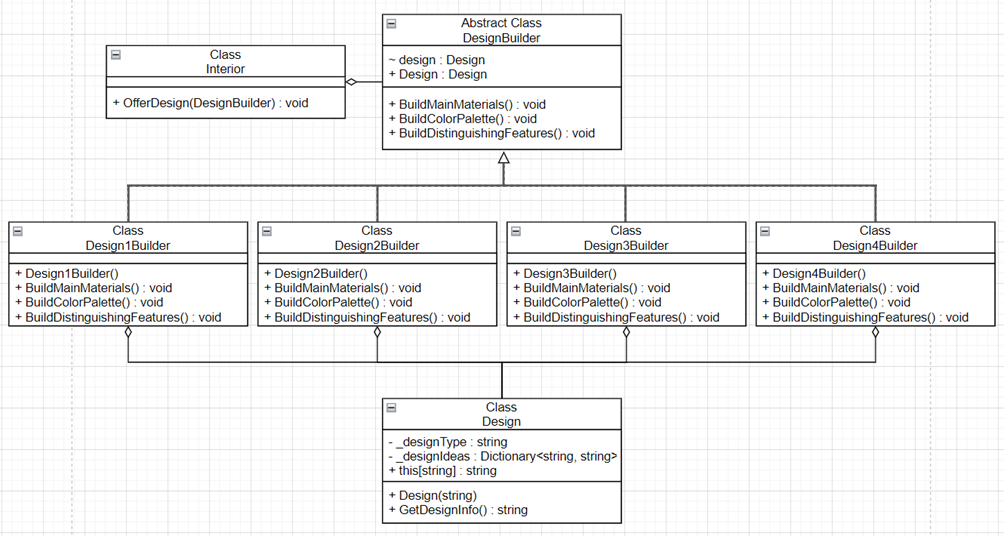
Даний шаблон було використано для того, щоб відслідковувати стан користувача і надавати йому відповідні можливості доступу до даних його квартири. Стан, на якому знаходиться власник квартири, залежить безпосередньо від готовності його квартири, що дозволяє без яких-небуть перешкод переходити саме на потрібний етап ремонту навіть за умови, що користувач вийшов із системи.

1. **Будівельник**

Будівельник – це породжуючий шаблон проєктування, який відокремлює конструювання складного об’єкта від його подання, тому в результаті одного і того ж процесу конструювання можуть утворюватися різні подання. Будівельник дозволяє створювати складні об’єкти покроково, що надає можливість використовувати один і той самий код для отримання різних представлень об’єктів.

*Учасники шаблону*:

* **Клієнт** (ChooseInterior) – створює об’єкт-розпорядник і конфігурує його потрібним об’єктом-будівельником; в нашому випадку ChooseInterior створює об’єкти DesignBuilder та Interior, завдяки яким надалі викликає класи-будівельники, які власне створюють типи інтер’єру;
* **Інтерфейс будівельника** (DesignBuilder) – оголошує кроки конструювання продуктів, загальні для всіх видів будівельників; DesignBuilder містить у собі порядок групування деталей про тип інтер’єру;
* **Конкретний будівельник** (Design1Builder–Design4Builder) – реалізує кроки будівництва (таких класів може бути безліч при чому кожен з них створює об’єкт по-своєму, дотримуючись оголошених кроків); конкретні будівельники можуть виконувати різноманітні об’єкти, які не мають спільного інтерфейсу; класи Design1Builder–Design4Builder містять у собі реалізовані модулі з інформацією про тип інтер’єру;
* **Продукт** (Design) – створюваний об’єкт; продукти, створені різними будівельниками, не зобов’язані мати спільний інтерфейс; в нашому випадку продукти мають спільний інтерфейс Design, який формує дані про кожен створений тип інтер’єру;
* **Директор** (Interior) – визначає порядок виклику кроків будівництва для виготовлення тієї чи іншої конфігурації продуктів; задає структуру, в якій подається інформація про інтер’єр.



*Рис. 2.1.3. Діаграма класів, які входять до шаблону «Будівельник»*

*Результати використання конструкції:*

* При виконанні програмного коду створюється 4 об’єкти (типи інтер’єру) однакових за структурою та різних за наявною в них інформацією;
* Дані, представлені у зручному для перегляду форматі, надаються користувачу для ознайомлення і вибору одного з них.

*Обґрунтування використання шаблону:*

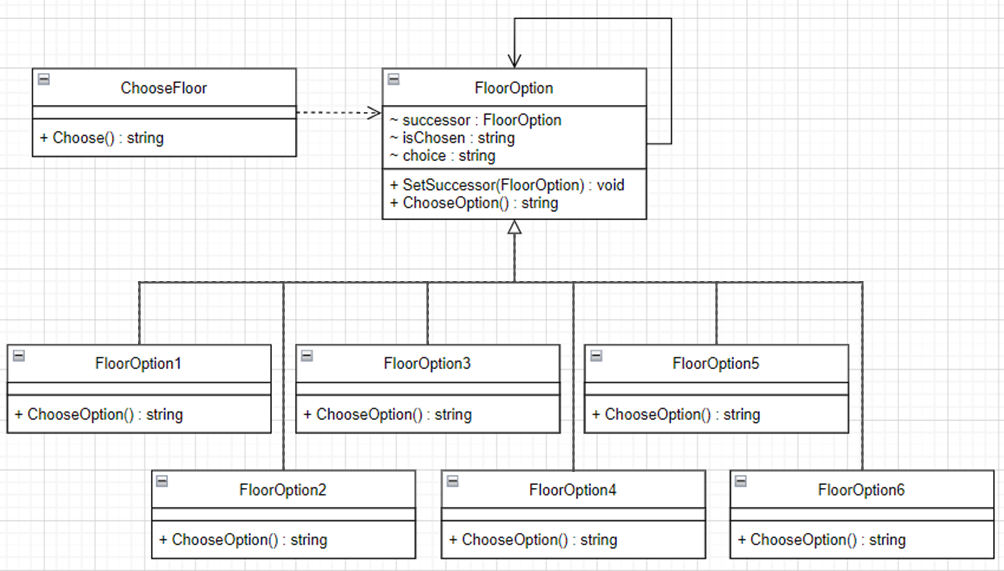
Даний шаблон був застосований з метою структурувати наявну інформацію. Кожен тип інтер’єру має свої характеристики, але при цьому система неначе конструктор покроково складає їх: назва, основні матеріали, палітра кольорів та риси, притаманні саме цьому інтер’єру. Так записи про кожен можливий дизайн квартири легко порівнювати між собою. Якби інформація надавалася без використання шаблону, то дані були нечитабельні та важкі для розуміння.

1. **Ланцюжок обов’язків**

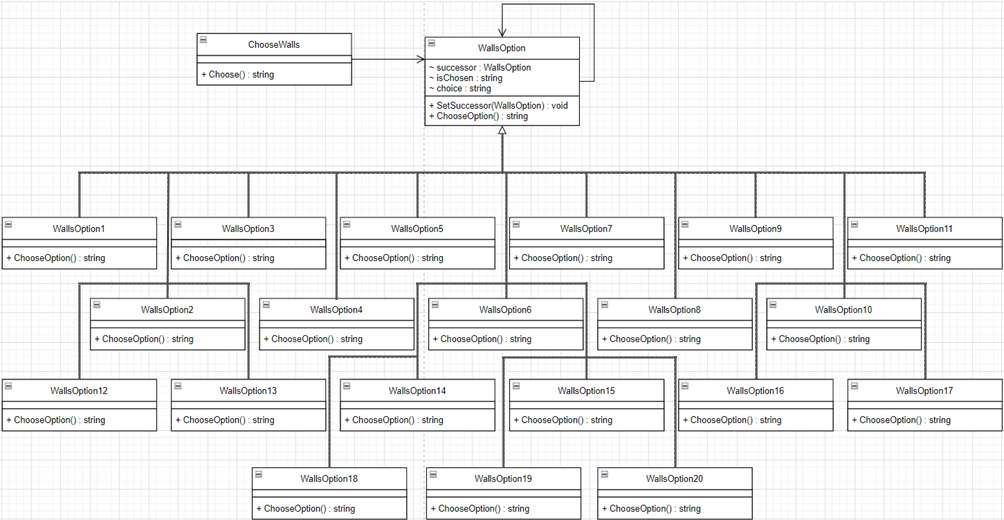
Ланцюжок обов’язків – поведінковий шаблон, призначений для організації в системі рівнів відповідальності. Даний шаблон будує об’єкти складених частин програми зв’язаними між собою ланцюжком для передачі запиту на обробку від більш низьких, деталізованих прошарків до більш глобальних, дозволяє уникнути зв’язування відправника запиту та його отримувача.

*Учасники шаблону*:

* **Обробник** (WallsOption, FloorOption) – визначає спільний для всіх конкретних обробників інтерфейс;
* **Конкретні обробники** (WallsOption1–WallsOption20, FloorOption1–FloorOption6) – містять у собі код обробки запитів (варіант оформлення стін або підлоги відповідно); якщо користувач робить вибір, то ланцюжок переривається і відповідь записується в базу даних, якщо ні, то ланцюжок циклічно виконується;
* **Клієнт** (ChooseFloor) – містить у собі ланцюжок обробників, які викликаються завжди в однаковому порядку.



*Рис. 2.1.4.1. Діаграма класів, які входять до шаблону «Ланцюжок обов’язків»*

**

*Рис. 2.1.4.2. Діаграма класів, які входять до шаблону «Ланцюжок обов’язків»*

*Результати використання конструкції:*

* Коли користувач переглядає список запропонованих варіантів, він може декілька разів пройтися по запропонованим позиціям і при цьому система не перенавантажується, оскільки зв’язок між об’єктами (запропонованими варіантами) спрощений;
* Користувач може вийти з ланцюжка на будь-якому етапі і при цьому його відповідь буде оброблена та збережена системою.

*Обґрунтування використання шаблону:*

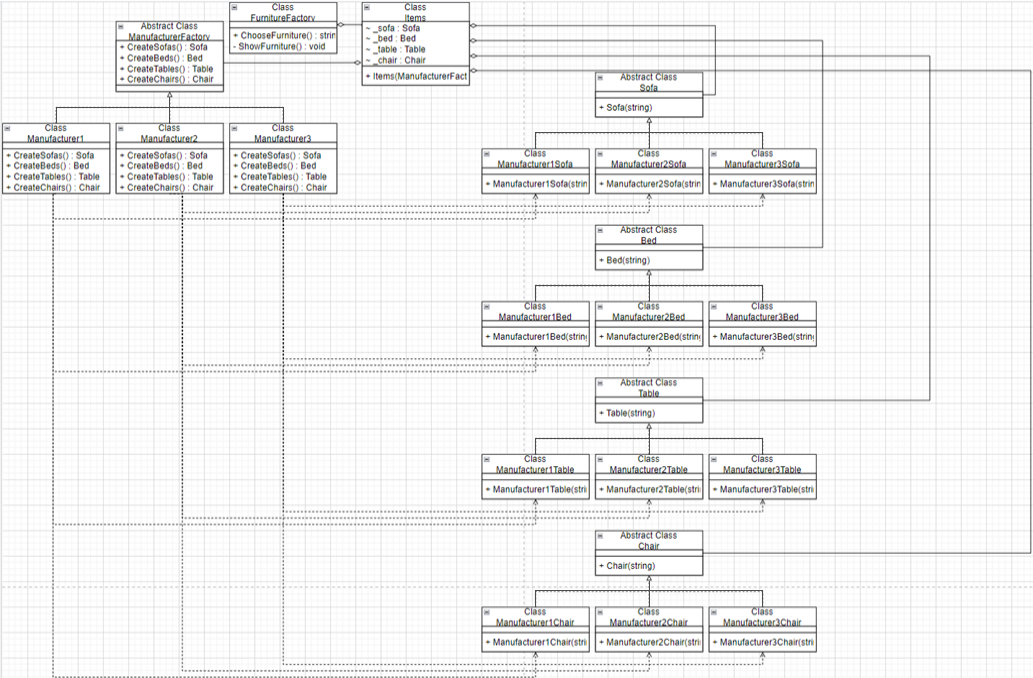
Даний шаблон було використано з метою полегшення зв’язків між об’єктами, що обробляють запити. Оскільки заздалегідь невідомо який саме варіант оформлення стін або підлоги обере власник, то об’єкт, який оброблятиме запит користувача буде визначатися динамічно. «Ланцюжок обов’язків» дозволяє створити таку архітектуру, яка буде достатньо гнучкою і при цьому правильно визначатиме порядок виконання об’єктів (варіантів оформлення).

1. **Абстрактна фабрика**

Це породжуючий шаблон проєктування, який дозволяє змінювати поведінку системи, варіюючи створювані об’єкти, але при цьому залишаючи незмінними інтерфейси. Надає інтерфейс для створення сімейств взаємопов’язаних або незалежних об’єктів, не визначаючи при цьому їх конкретних класів.

*Учасники шаблону*:

* **Абстрактні продукти** (Sofa, Bed, Table, Chair) – виступають інтерфейсами меблів, пов’язаних між собою за типом;
* **Конкретні продукти** (Manufacturer1Sofa, Manufacturer1Bed, Manufacturer1Table, Manufacturer1Chair, Manufacturer2Sofa, Manufacturer2Bed, Manufacturer2Table, Manufacturer2Chair, Manufacturer3Sofa, Manufacturer3Bed, Manufacturer3Table, Manufacturer3Chair) – класи, які відносяться до різних абстрактних продуктів, але мають однакових виробників;
* **Абстрактна фабрика** (ManudacturerFactory) – оголошує виробника, який створює різні абстрактні продукти;
* **Конкретна фабрика** (Manudacturer1, Manudacturer2, Manudacturer3) – класи, кожен з яких стосується свого виробника та реалізує методи абстрактної фабрики, дозволяючи створювати всі види меблів.



*Рис. 2.1.5. Діаграма класів, які входять до шаблону «Абстрактна фабрика»*

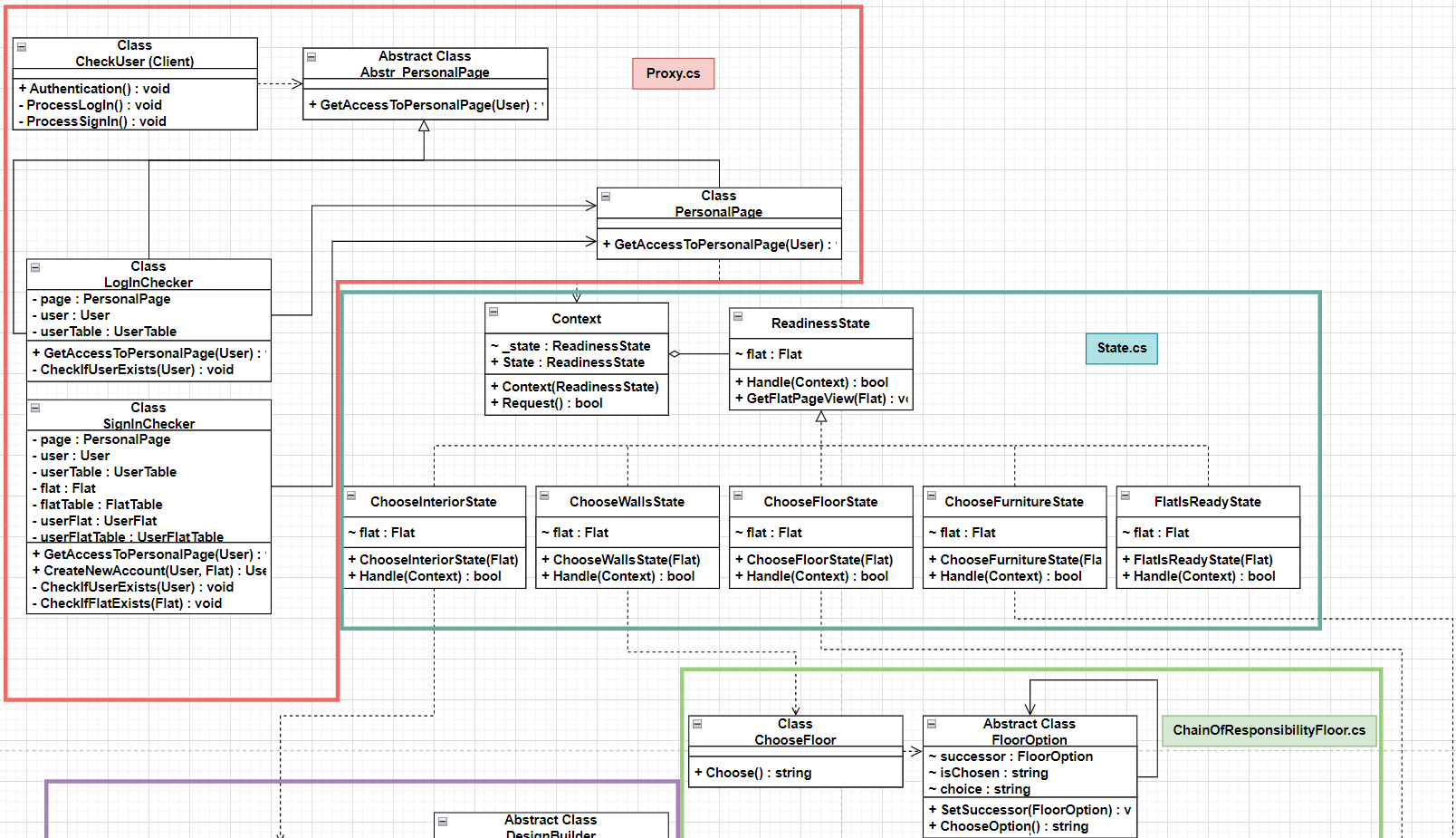
*Результати використання конструкції:*

* Створюється цілий список меблів (кожен типу у декілької екземплярах): диванів, ліжок, столів та стільців – які відрізняються своїм виробником;
* Кожна група об’єктів відображається користувачу окремо, що дозволяє взаємодіяти з кожним виробником індивідуально.

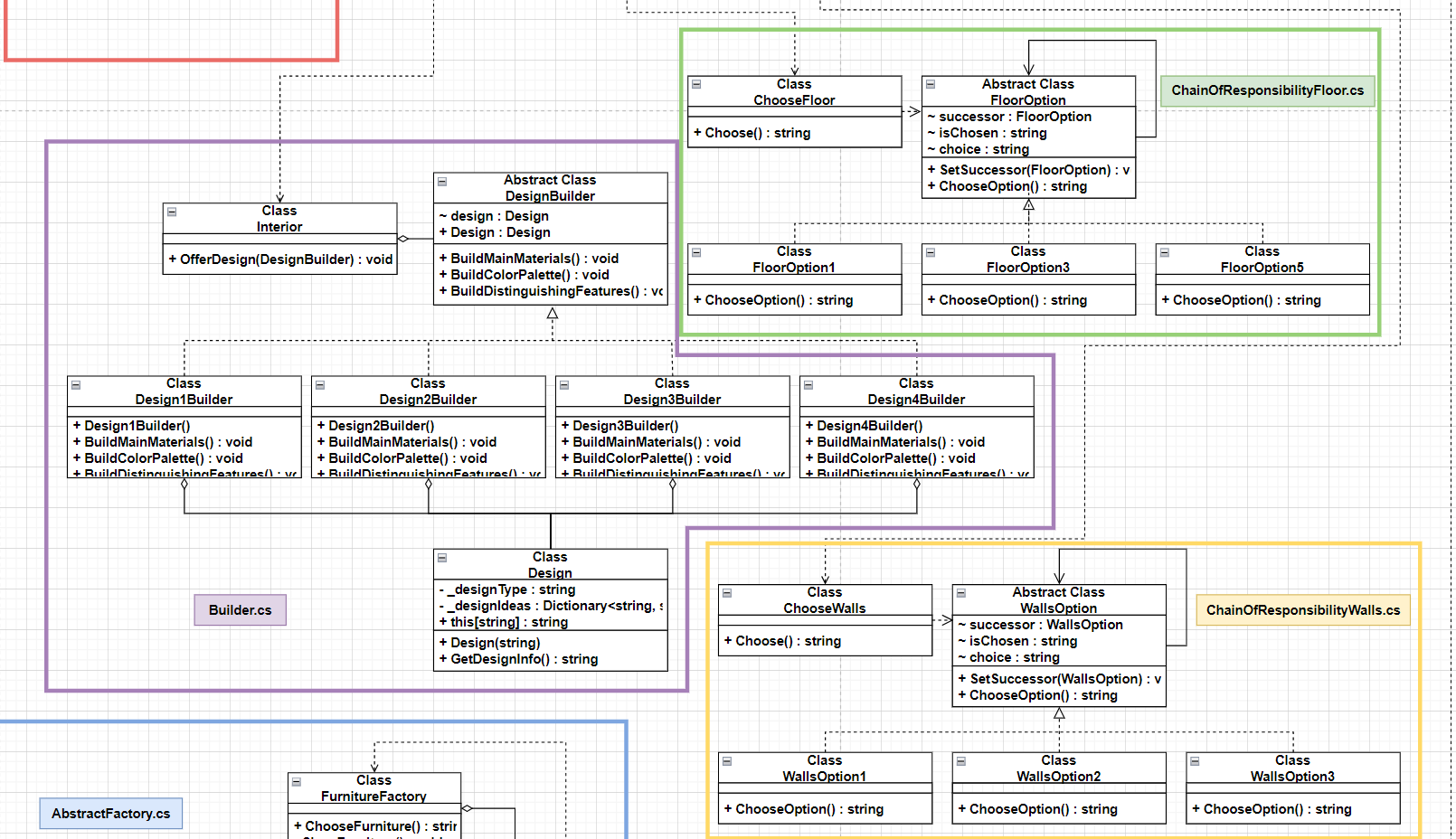
*Обґрунтування використання шаблону:*

Даний шаблон було використано з метою створення сімейств пов’язаних між собою об’єктів. Він дозволяє створити таку архітектуру, завдяки якій специфікація породжуваного об’єкту та його сімейства визначалася тільки в нащадках класу меблів. Завдяки «Абстрактній фабриці» в коді програми використовуються не специфічні класи, а абстрактні об’єкти, маніпуляція якими відбувається на більш високому рівні.

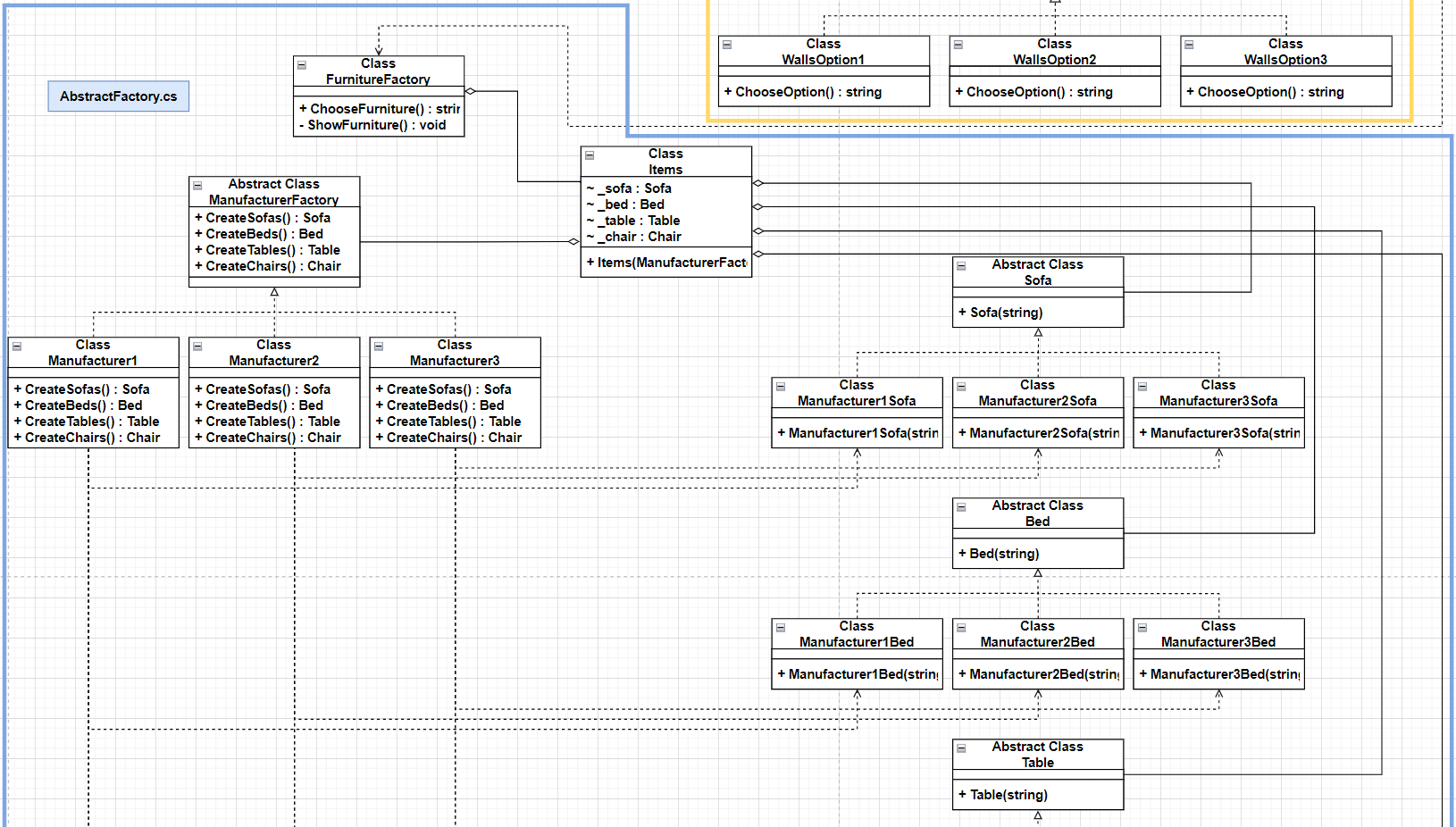
* 1. **Діаграма класів**



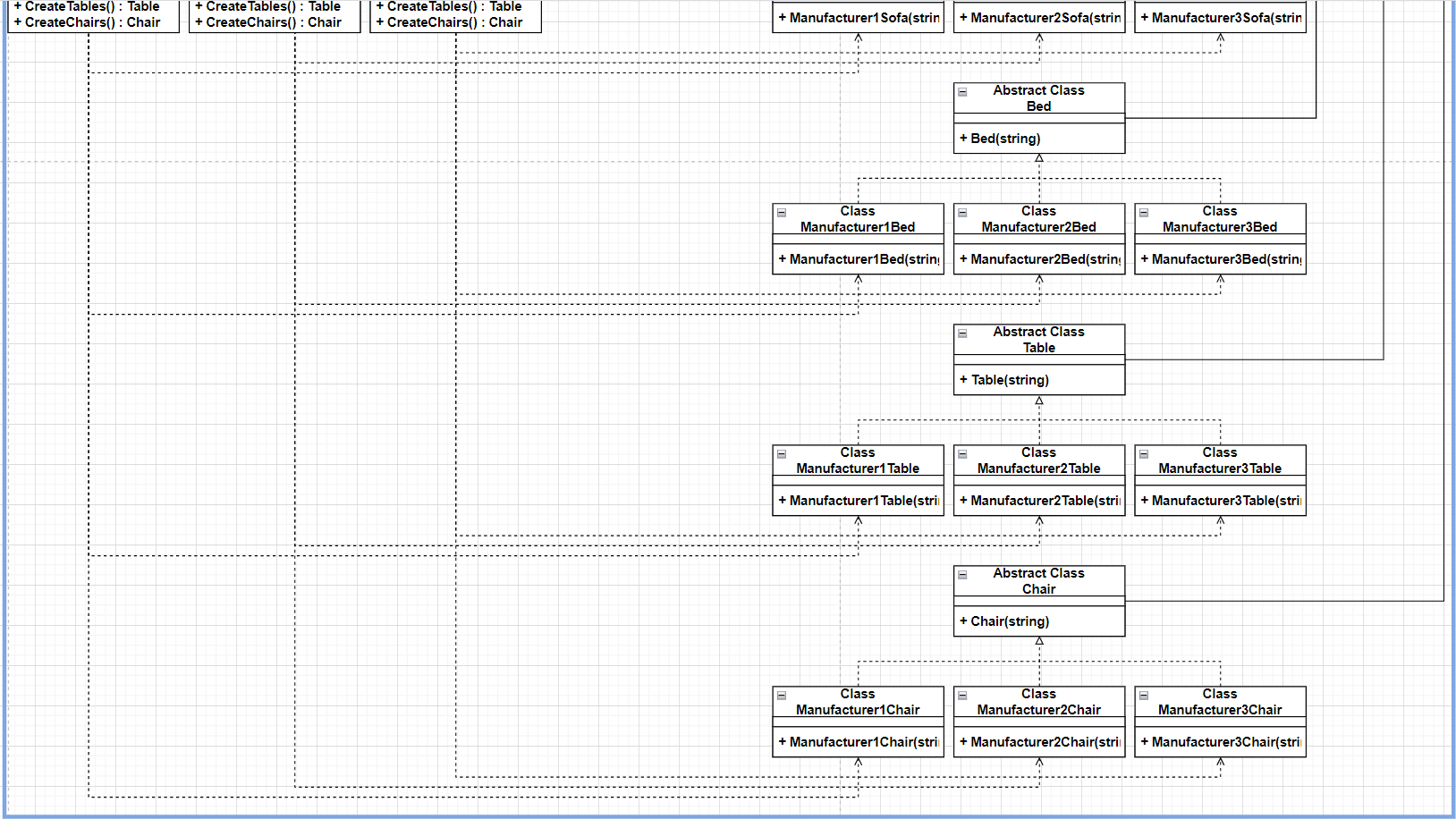
*Рис. 2.2, аркуш 1. Діаграма класів*



*Рис. 2.2, аркуш 2. Діаграма класів*

**

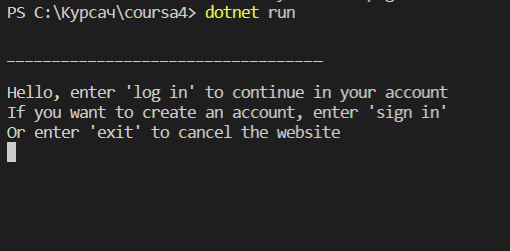
*Рис. 2.2, аркуш 3. Діаграма класів*

**

*Рис. 2.2, аркуш 3. Діаграма класів*

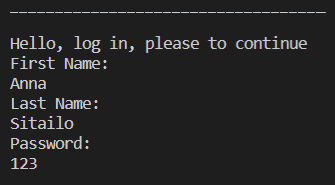
* 1. **Опис результатів роботи програми**

При запуску програми в консолі з’являється повідомлення про можливість входу в систему або реєстрацію.



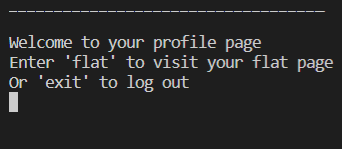
*Рис.2.3.1. Ілюстрація головної сторінки (початку роботи з програмою)*

Якщо користувач хоче увійти в систему, він вводить «log in», після чого йому потрібно ввести дані для входу.



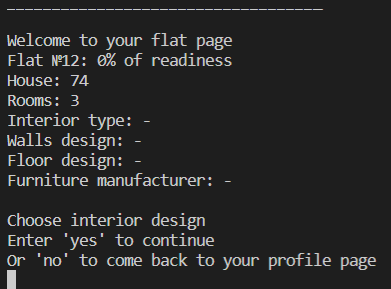
*Рис.2.3.2. Ілюстрація форм для входу в систему*

Якщо введені дані коректні та користувач міститься в системі, він заходить на сторінку свого профілю.



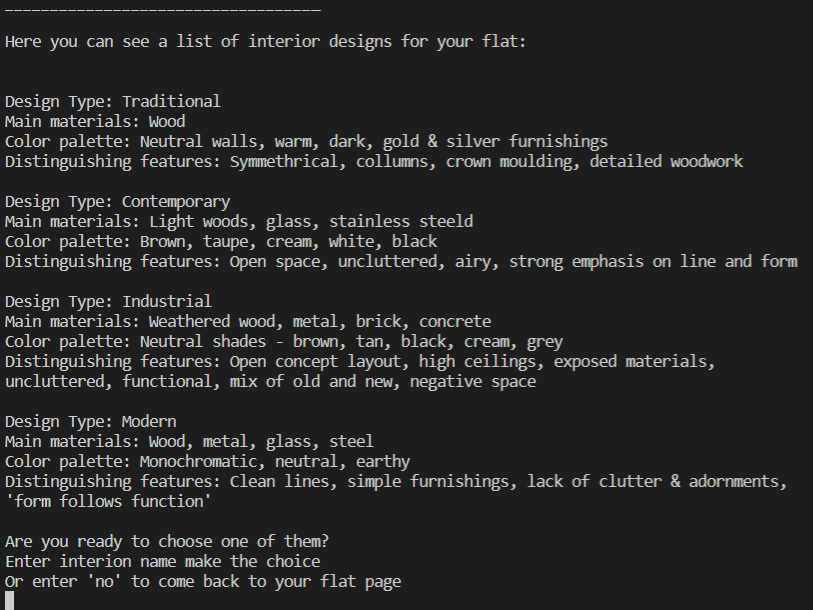
*Рис.2.3.3. Ілюстрація профілю користувача, який увійшов у систему*

Якщо користувач хоче перейти на сторінку своєї квартири, він вводить «flat» і переходить на нову сторінку.



*Рис.2.3.4. Ілюстрація даних про квартиру*

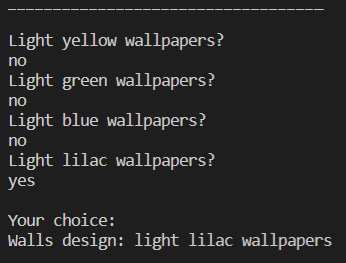
Під даними щоразу зображена коротка інструкція щодо того чи іншого етапу ремонту. Якщо користувач натискає «yes», то переходить на сторінку з вирішенням кожного з етапів ремонту.



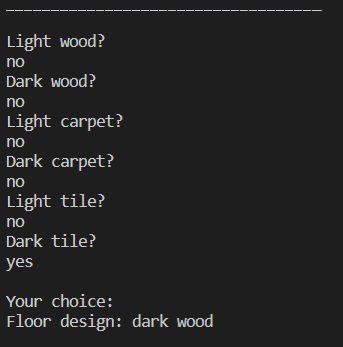
*Рис.2.3.5. Ілюстрація списку типів інтер’єру*

Власник вводить тип інтер’єру, який він хоче обрати для своєї квартири. Якщо дані коректні, то він повертається до даних своєї квартири (Рис.2.3.4.), під якими зображена інструкція для переходу до нового етапу ремонту. Якщо ж ні, то користувач також перегодить на сторінку з Рис.2.3.4., але при цьому данні про квартиру залишаються незмінними.

Процеси вибору стін та підлоги аналогічні, тому можемо одразу розглядати обидва. Користувачу пропонують певні варіанти у вигляді запитань. Якщо користувач хоче обрати якийсь із варіантів, йому просто потрібно після нього ввести «yes».



*Рис.2.3.6. Ілюстрація вибору стін*



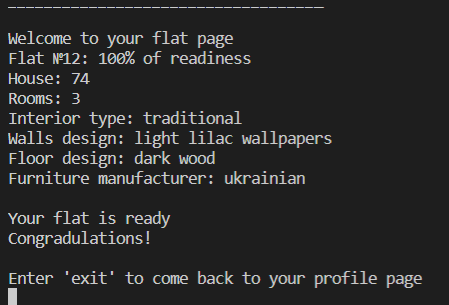
*Рис.2.3.7. Ілюстрація вибору підлоги*

Під час вибору виробника меблів користувачу потрібно ввести одного з них під списком всіх меблів.



*Рис.2.3.8. Ілюстрація вибору виробника меблів*

Коли всі етапи пройдено, під даними квартири зображується повідомлення про готовність квартири.



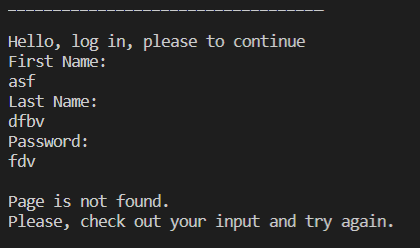
*Рис.2.3.9. Ілюстрація повідомлення про готовність квартири*

При некоректно введених даних у консоль виводиться повідомлення про помилку.



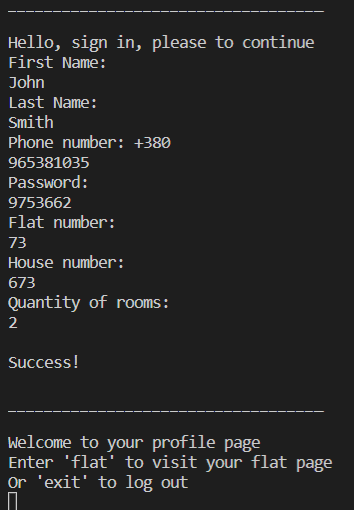
*Рис.2.3.10. Ілюстрація повідомлення про неправильно введені дані*

Якщо під час входу дані, введені користувачем, не містяться в системі, у консоль виводиться повідомлення про помилку.



*Рис.2.3.11. Ілюстрація повідомлення про відсутність користувача у системі*

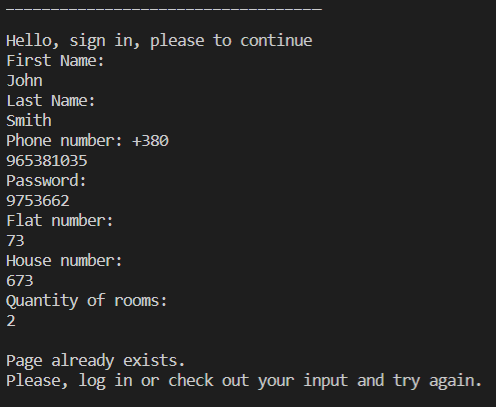
Якщо користувач хоче зареєструватися, він вводить «sign in», після чого йому потрібно ввести дані для реєстрації.



*Рис.2.3.12. Ілюстрація реєстрації користувача у системі*

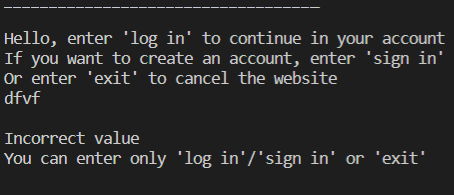
Одразу після реєстрації користувач переходить на сторінку свого профілю (Рис. 2.3.3.).

Якщо профіль з даними, введеними при спробі зареєструватися, вже існують в системі, то виводиться повідомлення про наявність користувача в системі.



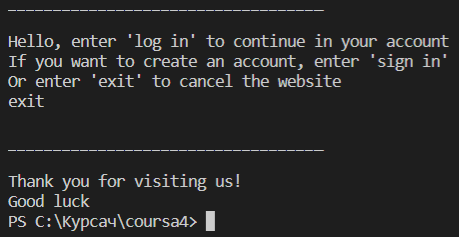
*Рис.2.3.13. Ілюстрація повідомлення про наявність такого користувача в системі*

Якщо користувач вводить неправильні дані, він отримує повідомлення про помилку.



*Рис.2.3.14. Ілюстрація повідомлення про некоректність введених даних*

Якщо користувач хоче припинити роботу програми, він вводить «exit», після чого в консоль виводиться повідомлення і програма завершує свою роботу.



*Рис.2.3.15. Ілюстрація завершення роботи програми*

## ВИСНОВКИ

Метою даної курсової роботи було розроблення програмного забезпечення для організації процесу ремонту квартир з використанням шаблонів проєктування. Підставою для розроблення стало завдання на виконання курсової роботи з дисципліни «Програмування» студентами ІІ курсу кафедри ПЗКС НТУУ «КПІ ім. І.Сікорського».

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, програмного та апаратного забезпечення, наведені у технічному завданні; створено відповідну документацію.

Розроблене програмне забезпечення дозволяє пришвидшувати процес вирішення організаційних питань під час проведення ремонту в квартирах. Дозволяє власникам контролювати та корегувати процес ремонту в їх квартирах дистанційно, без необхідності регулярного перебування в квартирі.

Програму створено на основі використання шаблонів проєктування: зокрема до структури розробленого програмного забезпечення входить реалізація п’яти шаблонів, які належать до різних груп шаблонів проєктування (структурних, породжувальних та поведінкових).

Для розроблення програмного забезпечення використано мову програмування C++ та платформа Windows.

Перспективним напрямком подальшого дослідження даної тематики є розширення функціоналу розробленої програми та створення web-версії спроектованої системи.

**СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ**

1. Приёмы объектно-ориентированного проектирования. Паттерны проектирования / Эрих Гамма, Ричард Хелм, Ральф Джонсон, Джон Влиссидес. – 1994. – 395 с.
2. Шаблоны проектирования. Новый подход к объектно-ориентированному анализу и проектированию / Алан Шаллоуей, Джеймс Р. Тротт. — М. : «Вильямс», 2002. — 288 с.
3. Крэг Ларман. Применение UML 2.0 и шаблонов проектирования. — М.: «Вильямс», 2006. — С. 736.
4. Пышкин, Е.В. Основные концепции и механизмы объектно-ориентированного программирования [Текст] / Е.В. Пышкин – СПб. : БХВ-Петербург, 2005. – 640 с.
5. Design Patterns [Електронний ресурс]: [Веб-сайт]. - Режим доступу: https://sourcemaking.com/design\_patterns (дата звернення 26.04.2022) - Назва з екрана
6. Software Design Patterns [Електронний ресурс]: [Веб-сайт]. - Режим доступу: https://www.geeksforgeeks.org/software-design-patterns (дата звернення 26.04.2022) - Назва з екрана