**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**ІМЕНІ ТАРАСА ШЕВЧЕНКА**

**ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**Кафедра інтелектуальних технологій**

**Лабораторна робота №9**

З дисципліни «Об'єктно-орієнтований аналіз та проєктування»

Тема роботи: «Структурні патерни GoF»

**Варіант №7**

Виконала студентка групи КН-32

Гнатюк Анна Михайлівна

Перевірила:

Красовська Ганна Валеріївна

**Київ – 2024**

**Мета:**

Навчитися застосовувати патери проектування при розробці об’єктно- орієнтованого програмного забезпечення.

**Завдання:**

**Виконання:**

**Інформаційна система міжгалактичного парку атракціонів**

1. **Результати об’єктно-орієнтованого проектування програмної системи.**
   1. **Діаграма класів системи з врахуванням доданих класів, що беруть участь в реалізації структурних патернів.**

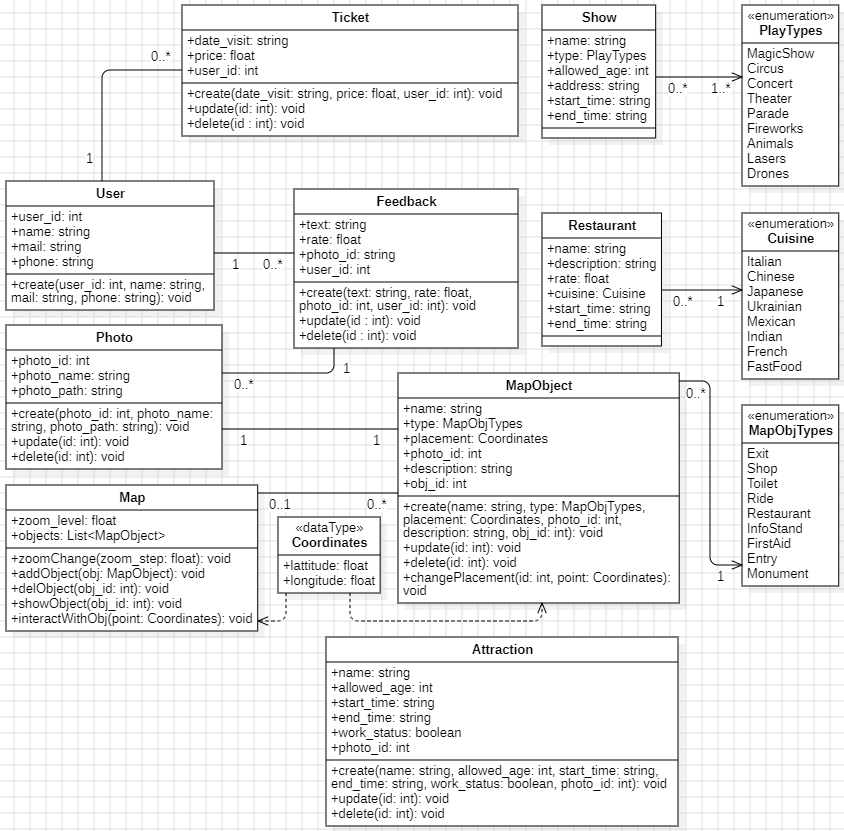


Рис. 1.1.1 – Діаграма класів системи

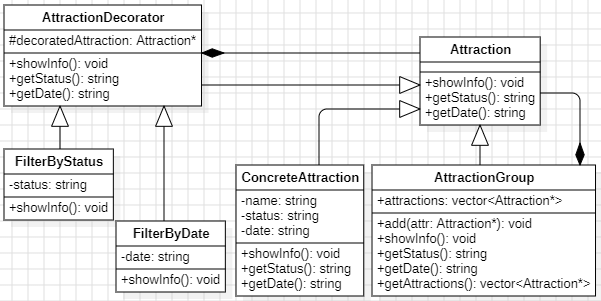


Рис. 1.1.2 – Діаграма класів що беруть участь в реаліз. СП Декоратор + Композит

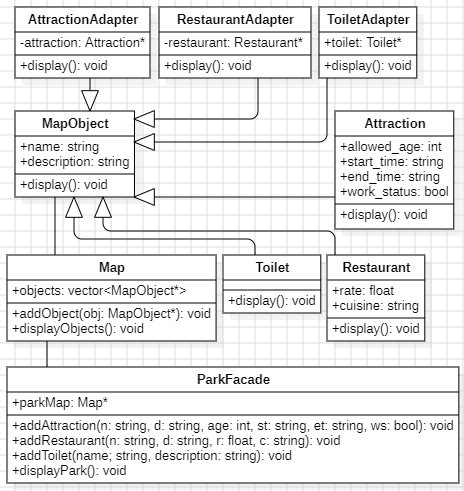


Рис. 1.1.3 – Діаграма класів що беруть участь в реаліз. СП Адаптер + Фасад

* 1. **Діаграма послідовностей для класів, що беруть участь в реалізації структурних паттернів.**

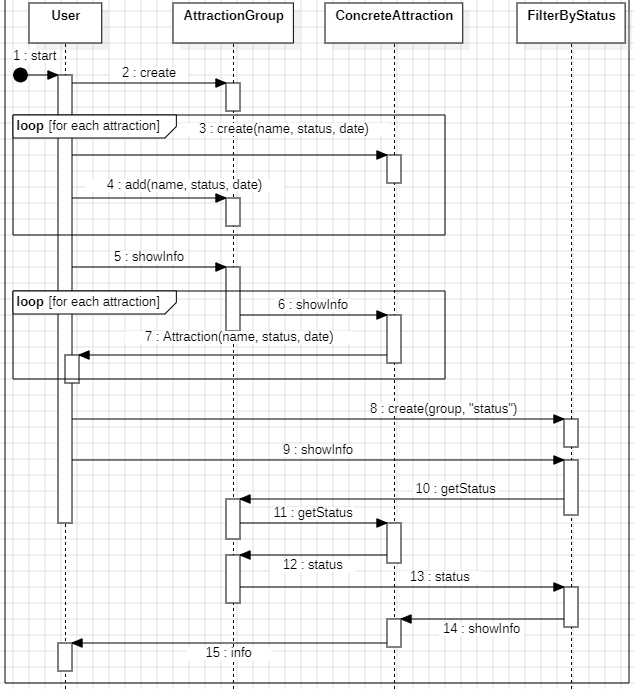


Рис. 1.2.1 – Діаграма послідовностей для класів що беруть участь в реаліз. СП Декоратор + Композит

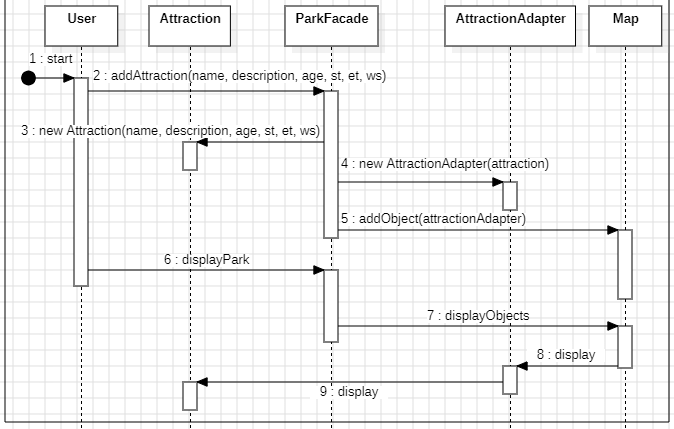


Рис. 1.2.2 – Діаграма послідовностей для класів що беруть участь в реаліз. СП Адаптер + Фасад

* 1. **Діаграма компонентів для розроблюваної системи з описом їх призначення.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Компонент** | **Призначення** | **Інтерфейси які надає** | **Інтерфейси які потрібні** |
| Client GUI | Відповідає за взаємодію з користувачем, відображення даних та отримання введення від користувача. |  | IManageAttractions, IManageTickets, IManageMap, IManageFeedback, IManagePhotos, IManageShows, IManageRestaurants  ISaveData IGetData |
| Attraction Service | Обробляє інформацію про атракціони, включаючи створення, оновлення та видалення атракціонів. | IManageAttractions | ISaveData IGetData |
| Ticket Service | Управляє інформацією про квитки, включаючи створення, оновлення та видалення квитків. | IManageTickets | ISaveData IGetData |
| Map Service | Управляє мапою парку, включаючи додавання, оновлення та видалення об'єктів на мапі. | IManageMap | ISaveData IGetData |
| Feedback Service | Обробляє відгуки користувачів, включаючи створення, оновлення та видалення відгуків. | IManageFeedback | ISaveData IGetData |
| Photo Service | Управляє фотографіями, включаючи завантаження, зберігання та відображення фотографій. | IManagePhotos | ISaveData IGetData |
| Show Service | Обробляє інформацію про шоу, включаючи створення, оновлення та видалення шоу. | IManageShows | ISaveData IGetData |
| Restaurant Service | Управляє інформацією про ресторани, включаючи створення, оновлення та видалення ресторанів. | IManageRestaurants | ISaveData IGetData |
| Database | Центральне сховище для всіх даних системи, включаючи дані про атракціони, квитки, мапу, відгуки, фотографії, шоу та ресторани. | ISaveData IGetData |  |

Таблиця 1.3.1 – Опис призначення компонентів

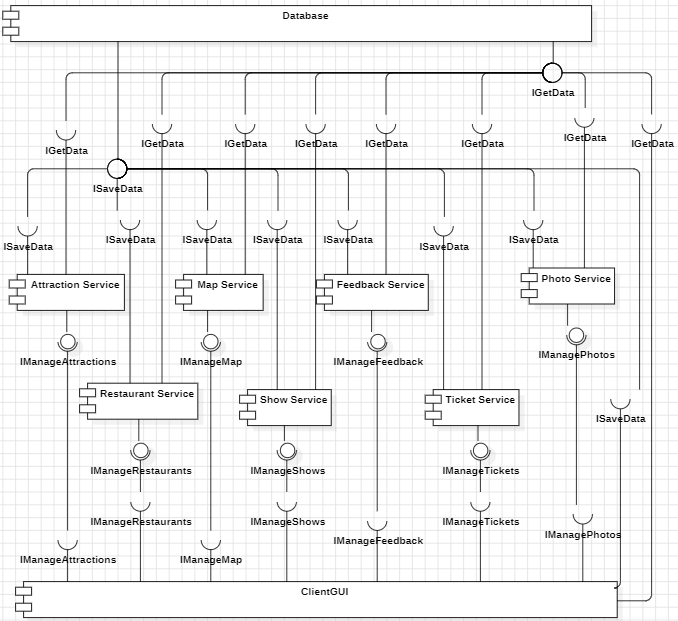


Рис. 1.3 – Діаграма компонентів для розроблюваної системи

1. **Результати тестування програмної реалізації структурних патернів.** 
   1. **Розібрати та порівняти патерни за двома групами, що наведені у таблиці 1. Дати відповідь на запитання про доцільність застосування кожного з цих патернів та можливість їх взаємозв’язку (сумісного використання) чи взаємозамінності.**

|  |  |  |  |
| --- | --- | --- | --- |
| Перша група | | Друга група | |
| Decorator | Composite | Adapter | Bridge |
| Proxy | Facade |
| Adapter | Proxy |

Таблиця 2.1.1 – завдання

|  |  |  |
| --- | --- | --- |
| **Паттерн** | **Опис** | **Доцільність застосування** |
| Decorator | Декоратор дозволяє динамічно додавати нові обов'язки до об'єктів без зміни їх коду. Він “обгортає” об'єкт, розширюючи його поведінку. | Використовується для динамічного додавання поведінки до об'єктів, коли успадкування є занадто статичним. |
| Composite | Композит групує схожі об'єкти в деревовидні структури. Він дозволяє працювати з простими та складеними об'єктами однаково. | Підходить для роботи з ієрархічними структурами, де потрібно однаково обробляти як прості, так і складені об'єкти. |
| Proxy | Проксі є сурогатом або замісником іншого об'єкта, що контролює доступ до нього. | Корисний для контролю доступу до об'єктів, оптимізації використання ресурсів та забезпечення додаткового рівня непрямого доступу. |
| Adapter | Адаптер перетворює інтерфейс класу в інший інтерфейс, що очікується клієнтським кодом. Він дозволяє працювати разом класам з несумісними інтерфейсами. | Застосовується для інтеграції класів з несумісними інтерфейсами без зміни їх коду. |
| Bridge | Брідж відокремлює абстракцію від її реалізації так, що їх можна змінювати незалежно. Це корисно для підтримки різних реалізацій одного інтерфейсу. | Доцільний для систем з необхідністю підтримки кількох реалізацій одного інтерфейсу та їх незалежної модифікації. |
| Facade | Фасад надає уніфікований інтерфейс до набору інтерфейсів в підсистемі. Він спрощує використання складних систем, приховуючи їх складність. | Використовується для спрощення роботи з складними системами, надаючи простий інтерфейс для клієнтів. |

Таблиця 2.1.2 – Розбір паттернів

|  |  |  |  |
| --- | --- | --- | --- |
| **Паттерн 1** | **Паттерн 2** | **Можливість взаємозв’язку** | **Можливість взаємозамінності** |
| Decorator | Composite | Декоратор може додавати нові обов'язки до компонентів Композита, дозволяючи динамічно розширювати їхній функціонал | Вони не можуть бути взаємозамінними, бо мають різні цілі |
| Decorator | Proxy | Декоратор може додавати нові обов'язки до Proxy, розширюючи його функцінал динамічно | Вони не можуть бути взаємозамінними, оскільки Proxy контролює доступ, а Декоратор розширює поведінку |
| Decorator | Adapter | Декоратор може додавати нові обов'язки до адаптованих Адаптером об'єктів | Вони не можуть бути взаємозамінними, оскільки Адаптер змінює інтерфейс, а Декоратор додає нову поведінку |
| Adapter | Bridge | Адаптер може використовуватися для адаптації різних реалізацій, що використовуються в Бріджі | Вони не можуть бути взаємозамінними, оскільки мають різні цілі: Адаптер адаптує інтерфейс, а Брідж розділяє абстракцію і реалізацію |
| Adapter | Facade | Адаптер може використовуватися для адаптації інтерфейсів компонентів, що використовуються Фасадом | Вони не можуть бути взаємозамінними, оскільки Фасад спрощує інтерфейс, а Адаптер адаптує його |
| Adapter | Proxy | Адаптер може адаптувати інтерфейс Proxy для сумісності з іншими компонентами | Вони можуть бути взаємозамінними в деяких випадках, коли Proxy використовується для адаптації інтерфейсів |

Таблиця 2.1.3 – Аналіз взаємозв’язку паттернів

* 1. **Самостійно для свого варіанту завдання придумати та програмно реалізувати по одному прикладу сумісного використання паттернів з двох наведених груп.**

Для виконання завдання було обрано так дві пари паттернів:

Decorator + Composite

Adapter + Proxy

* + 1. **Реалізація сумісного використання паттернів Decorator та Composite**

Декоратор та Композит можна реалізувати для фільтрації атракціонів, для варіанту використання та юзер сторі “Перегляд графіку роботи атракціонів”.

Так як суть декоратора, в розширенні функціональності об’єктів, застосуємо його для фільтрації атракціонів. Декратор буде додавати нові обов’язки фільтру (за новими параметрами).

Так як суть Композита в групуванні схожих об’єктів в деревовидні структури, застосуємо його для групування атракціонів. Композит буде дозволяти нам об’єднувати атракціони в групи, і викликати методи для всіх атракціонів у групі.

Для реалізації Декоратора, додамо опції фільтрації за датою та за статусом роботи:

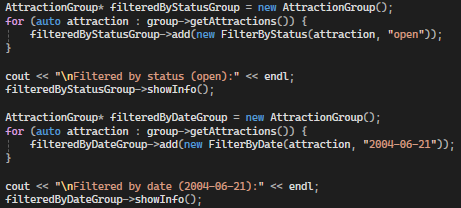


Рис.2.2.1.1 – використання Декоратора

Для реалізації Композита, згрупуємо атракціони:

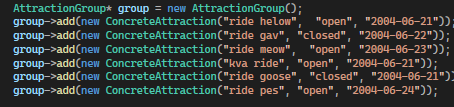


Рис.2.2.1.2 – використання Композита

Перевіримо роботу, ми повинні побачити повний список горок, відфільтрований за статусом роботи (працює) та відфільтрований за датою (21.06.2004)

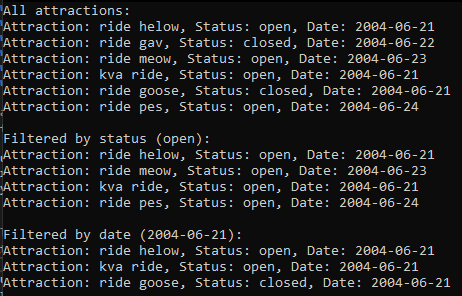


Рис.2.2.1.3 – результат роботи паттернів Decorator + Composit

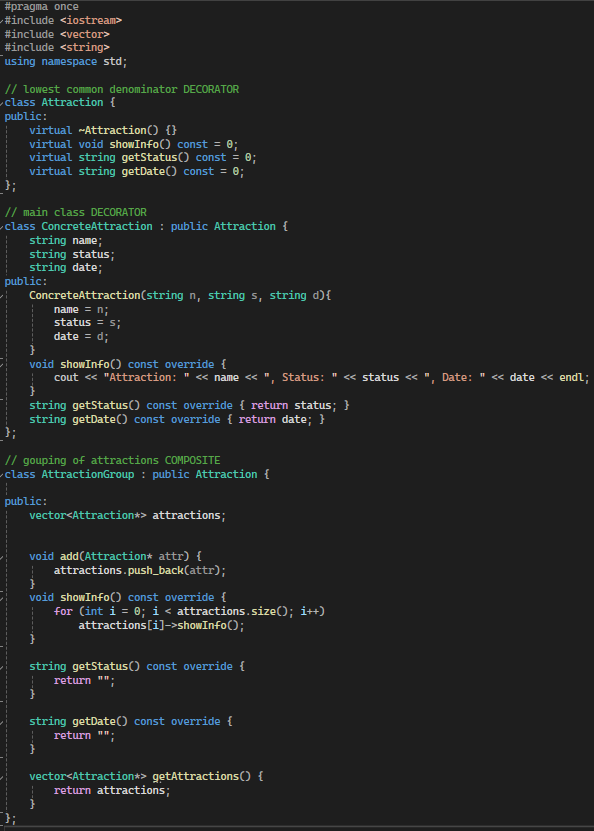
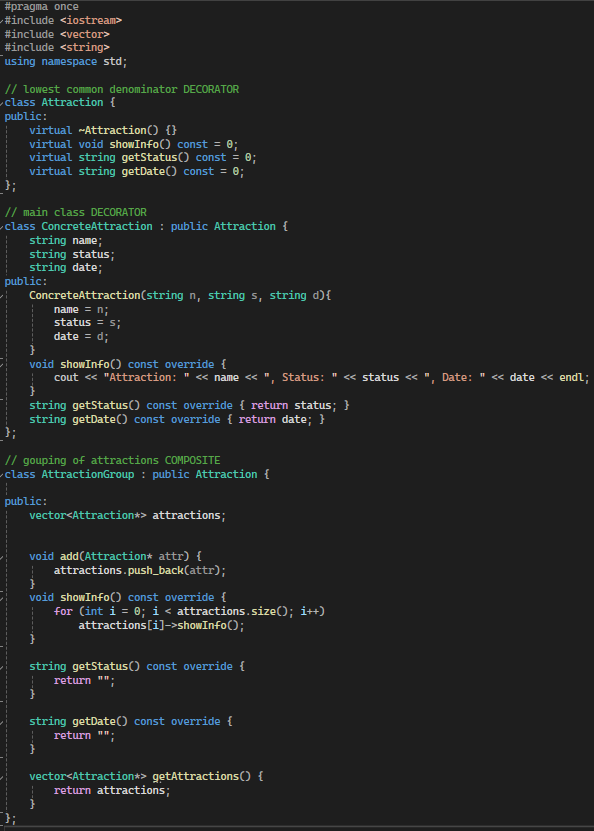


Рис.2.2.1.4 – Attraction.h

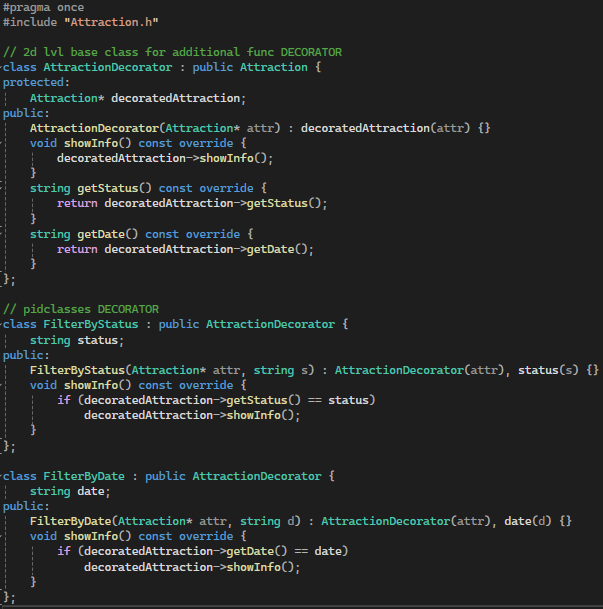


Рис.2.2.1.5 – AttractionDecorator.h



Рис.2.2.1.6 – main.cpp

* + 1. **Реалізація сумісного використання паттернів Adapter та Facade**

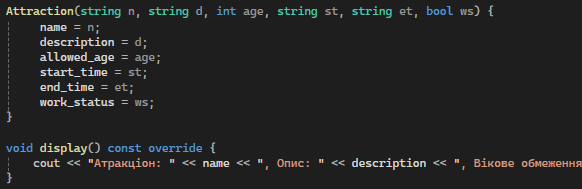
Паттерни Адаптер та Фасад можна реалізувати для керування об’єктами для мапи, для варіанту використання та юзер сторі “Перегляд мапи парку”.

Так як суть Адаптера, в тому щоб дозволити класам з несумісними інтерфейсами працювати разом, ми можемо використати його для стандартизації (адаптації) різних типів об’єктів що можуть бути на мапі, щоб вони відповідали єдиному інтерфейсу.

Так як суть Фасаду в забезпеченні спрощеного інтерфейсу для складних систем, ми можемо використати його для спрощення управління мапою та об’єктами. Фасад буде надавати зручні(спрощені) методи для додавання, видалення та відображення об’єктів на мапі.

Нехай, буде створюватись сутність Атракціон, в якого є крім назви та опису, ще вікове обмеження, час роботи, статус роботи і функція для виведення всіх даних. Також буде створюватись сутність Туалет в якого крім назви та опису ще є рейтинг та кухня, і відповідна функція для виведення власної інформації. На мапі повинні відображатись лише назва та опис точки, тому нам потрібно адаптувати класи.

Адаптуємо Атракціон, стандартизуємо метод виведення інформації.



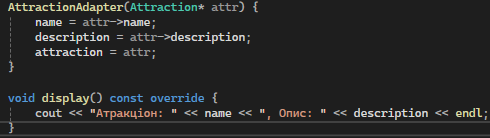


Рис.2.2.2.1 – застосування паттерну Адаптер

Для зручного управління мапою, створимо фасад

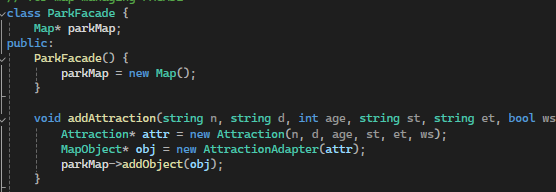


Рис.2.2.2.2 – застосування паттерну Фасад

Тепер потрібно протестувати роботу. Додамо точки різних класів.

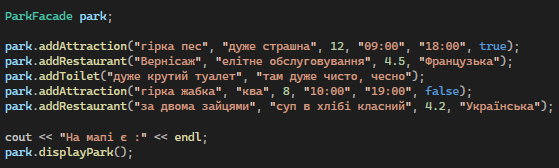


Рис.2.2.2.3 – застосування обох паттернів

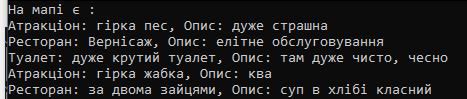


Рис.2.2.2.4 – результат роботи паттернів Adapter + Façade

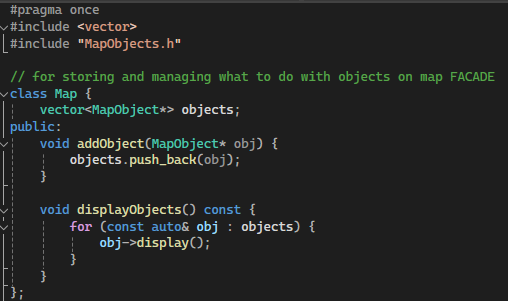


Рис.2.2.2.5 Map.h

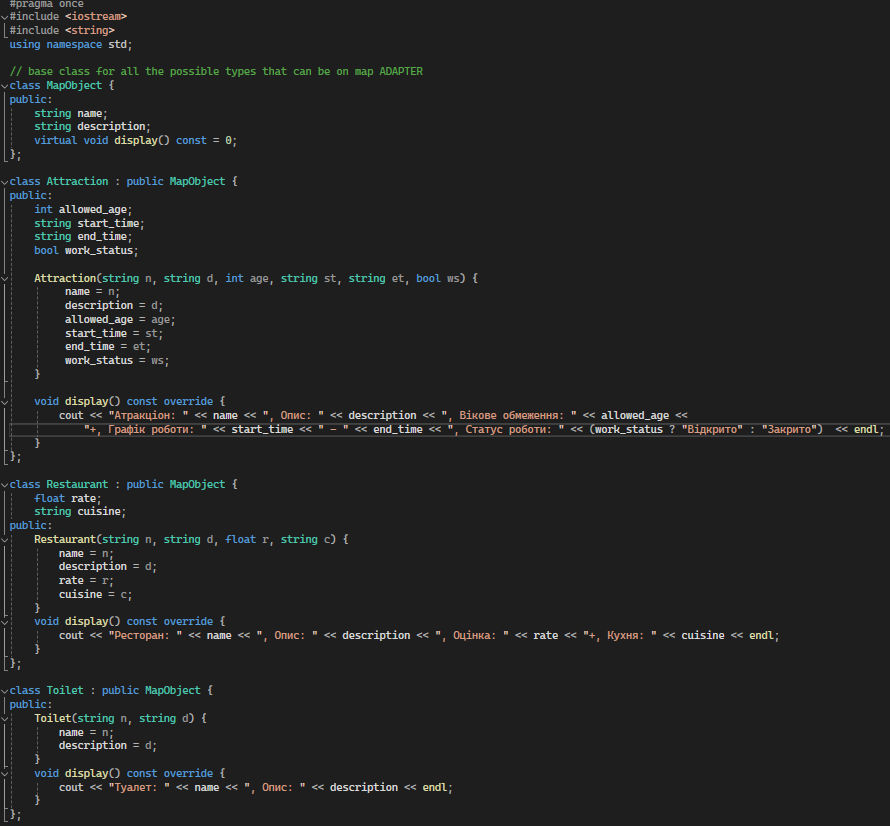


Рис.2.2.2.6 – – MapObject.h



Рис.2.2.2.7 – Adapters.h

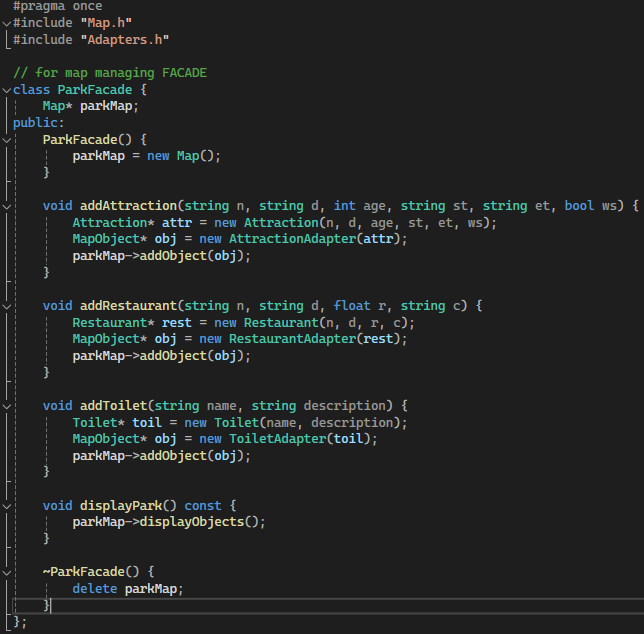


Рис.2.2.2.8 – ParkFacade.h

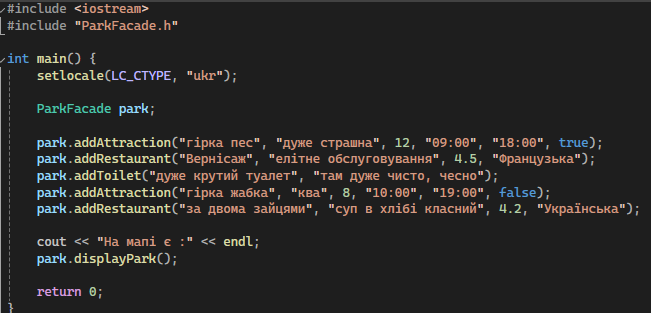


Рис.2.2.2.9 – main.cpp

1. **Програмний код.**

Програмний код наведено на рисунках.

Посилання на гітхаб:  
<https://github.com/ffructose/Lab9_OOAP>

**Висновки:**

В ході виконання даної лабораторної роботи було засвоєно застосовування паттернів проектування при розробці об’єктно- орієнтованого програмного забезпечення.