Чернівецький національний університет імені Юрія Федьковича Факультет математики та інформатики Кафедра математичного моделювання

**Лабораторна робота №4**

з навчальної дисципліни: “ Проектування програмних систем”

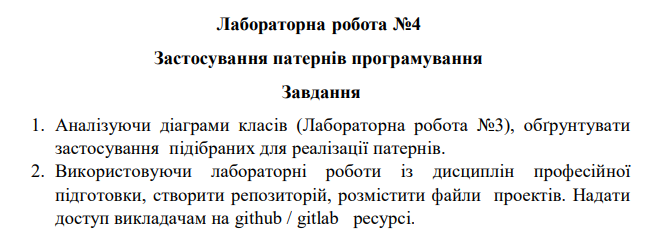
Виконав: студент 3 курсу

307 групи

спеціальності “Системний аналіз”

Граб А.М.

**Чернівці – 2023**

****

**ПОЯСНЕННЯ ВИБОРУ ПАТЕРНІВ ПРОГРАМУВАННЯ ДЛЯ СИСТЕМИ**

**Патерн Singleton**

**Де застосувати:** Клас System

**Контрольований доступ**: Патерн Singleton забезпечує наявність лише одного екземпляра класу System в додатку. Це необхідно, щоб гарантувати, що вся система управляється єдиним центральним об'єктом, який координує роботу всіх інших компонентів.

**Глобальний доступ:** Використання Singleton дозволяє глобальний доступ до об'єкта системи через єдиний статичний метод. Це полегшує управління системними ресурсами та налаштуваннями.

**Запобігання дублювання:** Singleton запобігає створенню додаткових екземплярів класу, що знижує ризик помилок, пов'язаних з некоректним управлінням станом системи.

**Патерн Factory**

**Де застосувати:** Клас Ticket

**Гнучкість:** Патерн Factory дозволяє створювати об'єкти без необхідності знати точний клас, який потрібно створити. Це знижує зв'язність між класами і спрощує внесення змін у систему.

**Масштабованість:** Додаючи нові типи квитків (наприклад, StandardTicket, VIPTicket), можна просто оновити фабрику, не змінюючи клієнтський код, що створює квитки. Це підвищує масштабованість системи.

**Інкапсуляція:** Фабрика інкапсулює логіку створення об'єктів, що спрощує підтримку і розширення системи, оскільки всі зміни в логіці створення об'єктів зосереджені в одному місці.

**Патерн Observer**

**Де застосувати:** Класи User та Email

**Масштабованість і розширюваність**: Патерн Observer дозволяє легко додавати нових спостерігачів або змінювати існуючих, не змінюючи основний код об'єкта, що генерує події (наприклад, коли користувач купує квиток, система повідомляє через електронну пошту).

**Декуплінг:** Використання Observer зменшує зв'язність між об'єктами, що робить систему більш гнучкою і легко підтримуваною. Об'єкти не повинні знати про реалізацію спостерігачів, лише про їхній інтерфейс.

**Динамічне повідомлення**: Observer забезпечує динамічне повідомлення спостерігачів про зміну стану суб'єкта, що дозволяє автоматично оновлювати інтерфейси користувача або виконувати інші дії без прямого втручання в код об'єкта, що генерує події.

**Висновок**

Застосування патернів Singleton, Factory та Observer дозволяє забезпечити гнучкість, масштабованість і підтримуваність системи. Вони допомагають зменшити зв'язність між компонентами, спрощують управління об'єктами і забезпечують централізований контроль стану системи. Це робить систему більш стійкою до змін та легкою в підтримці.

**Посилання на GitHub:**<https://github.com/GarethBale9/Software_systems_design_Hrab_307>