Міністерство освіти і науки України
Національний технічний університет України “Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №4
З дисципліни «Технології розроблення програмного забезпечення»
Тема: **«ШАБЛОНИ «SINGLETON», «ITERATOR», «PROXY», «STATE», «STRATEGY»»**
Office communicator

Виконала:
Студент групи ІА-22
Ільніцький П А

Перевірив:
Мягкий М. Ю.

Київ-2024

Зміст

Тема:
ШАБЛОНИ «SINGLETON», «ITERATOR», «PROXY», «STATE», «STRATEGY»

Мета:
Ознайомитися з основними шаблонами проєктування, такими як «Singleton», «Iterator», «Proxy», «State» та «Strategy», вивчити їхні принципи роботи та навчитись застосовувати для створення гнучкого та масштабованого програмного забезпечення в загальній розробці програмних систем.

Варіант:

..13 Office communicator (strategy, adapter, abstract factory, bridge, composite, client-server)

Мережевий комунікатор для офісу повинен нагадувати функціонал програми Skype з можливостями голосового / відео / конференц-зв'язку, відправки текстових повідомлень і файлів (можливо, оффлайн), веденням організованого списку груп / контактів.

Хід роботи

1. Реалізувати не менше 3-х класів відповідно до обраної теми

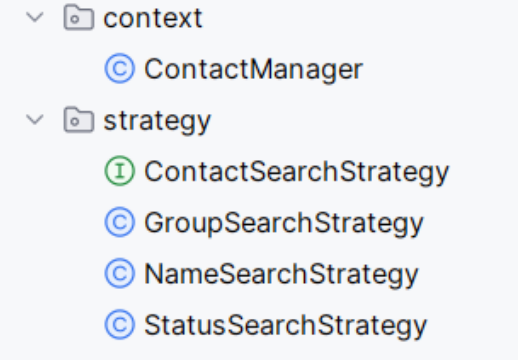


Рис. 1 — Структура проекту

Під час виконання лабораторної роботи було розроблено пять класів, які реалізують функціонал пошуку (рис. 1). Нижче наведено детальний опис кожного з цих класів:

1. **Контекст (Context)** – клас, який зберігає поточну стратегію та надає метод для її зміни. Він використовує стратегію для виконання певної операції, такої як пошук контактів.
2. **Інтерфейс стратегії (Strategy Interface)** – визначає загальний метод для реалізацій стратегій. Він надає контракт, який повинні імплементувати конкретні стратегії.

3. **Реалізація стратегії за ім'ям (NameSearchStrategy)** – конкретна реалізація стратегії пошуку, яка шукає контакти за іменем.
4. **Реалізація стратегії за статусом (StatusSearchStrategy)** – реалізація стратегії пошуку, яка шукає контакти за їх статусом.
5. **Реалізація стратегії за групою (GroupSearchStrategy)** – стратегія пошуку, що базується на групах, до яких належать контакти.

2. Реалізувати один з розглянутих шаблонів за обраною темою

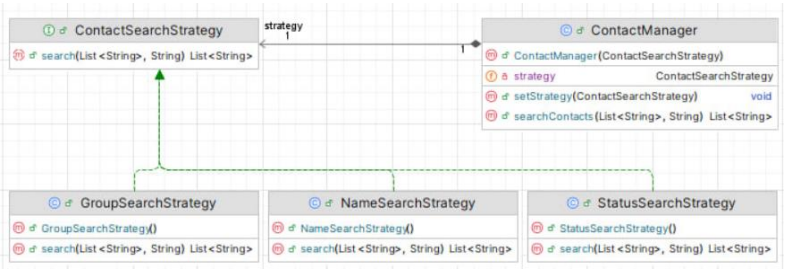


Рис. 2 — Діаграма класів

У проекті ця реалізація патерну Стратегія дозволяє гнучко змінювати методи пошуку контактів залежно від обраного критерію. Клас ContactManager використовує інтерфейс ContactSearchStrategy для визначення стратегії пошуку, яку можна змінювати в реальному часі. Кожен конкретний клас стратегії (GroupSearchStrategy, NameSearchStrategy, StatusSearchStrategy) реалізує пошук за певними параметрами, такими як група, ім'я або статус. Це дозволяє розширювати функціональність пошуку, додаючи нові стратегії без необхідності змінювати основну логіку в класі ContactManager.

Опис патерну

Шаблон Стратегії - це поведінковий шаблон проектування, який дозволяє об'єкту змінювати свою поведінку, інкапсулюючи різні стратегії чи алгоритми в окремі класи. Цей шаблон стає в нагоді, коли є кілька алгоритмів чи стратегій, які можуть використовуватися взаємозамінно, і ви хочете мати можливість вибирати відповідну стратегію під час виконання.

У випадку проекту, показаного на схемі, шаблон Стратегії можна використовувати для обробки різних стратегій пошуку у користувацькому інтерфейсі. На схемі показано кілька класів, таких як `ContactSearchStrategy`, `NameSearchStrategy` і `StatusSearchStrategy`, які представляють різні стратегії пошуку, які можна використовувати для пошуку контактів.

Клас `ContactManager` ймовірно буде основною точкою входу для користувацького інтерфейсу і використовуватиме один із класів стратегії пошуку для виконання фактичного пошуку. Клас `ContactManager` не повинен знати деталей того, як працює кожна стратегія пошуку; йому потрібно лише знати, що вони всі реалізують загальний інтерфейс чи абстрактний клас, який визначає функціональність пошуку.

Такий підхід має кілька переваг:

1. Гнучкість: Інкапсулюючи стратегії пошуку в окремі класи, легко додавати, видаляти або змінювати доступні стратегії пошуку без впливу на клас `ContactManager`.

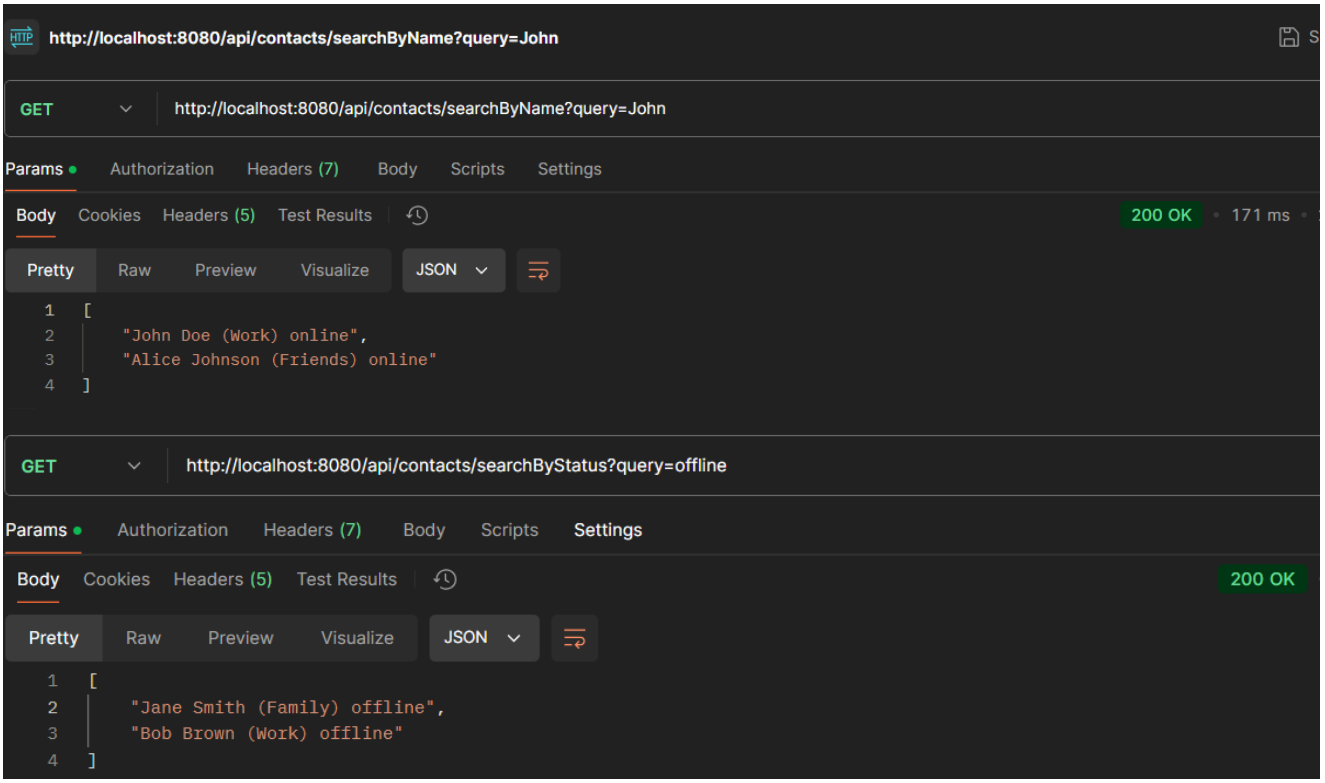
2. Можливість тестування: Кожну стратегію пошуку можна тестувати незалежно, що полегшує забезпечення правильності всієї системи.

3. Можливість підтримки: Код стає більш модульним і зрозумілим, оскільки логіка пошуку ізольована в окремих класах.

4. Можливість повторного використання: Класи стратегії пошуку можуть бути потенційно повторно використані в інших частинах програми чи навіть в інших проектах.

Загалом, шаблон Стратегії дозволяє проекту мати більш гнучку та підтримувану архітектуру, де функціональність пошуку можна легко розширювати чи змінювати без впливу на основну логіку класу `ContactManager`.

Перевірка роботи



Висновки:

У ході лабораторної роботи було реалізовано патерн Strategy для гнучкої зміни поведінки обробки даних у проекті менеджера завантажень. Патерн дозволяє абстрагувати різні алгоритми обробки та застосовувати їх залежно від конкретних умов. Це підвищує гнучкість і масштабованість програми, оскільки дає можливість без зміни основного коду додавати нові стратегії обробки. Крім того, патерн забезпечує зручну підтримку та розширення програми без ризику порушення існуючої логіки.