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

**Технології розроблення програмного забезпечення**

Лабораторна робота №4
«ШАБЛОНИ «SINGLETON», «ITERATOR», «PROXY», «STATE», «STRATEGY».»

Виконав:
студент групи ІА–22:
Клочков І. Ф.

Перевірив:
Мягкий Михайло Юрійович

Київ 2024

**Зміст**

**Тема:** шаблони «singleton», «iterator», «proxy», «state», «strategy».

**Мета:** ознайомитися з шаблонами проектування «singleton», «iterator», «proxy», «state», «strategy». Реалізувати частину функціоналу програми за допомогою одного з розглянутих шаблонів.

**Хід роботи**

**..18 Shell (total commander) (state, prototype, factory method, template method, interpreter, client-server)**

Оболонка повинна вміти виконувати основні дії в системі - перегляд файлів папок в файлової системі, перемикання між дисками, копіювання, видалення, переміщення об'єктів, пошук.

# 1. Короткі теоретичні відомості

**Що таке шаблони проектування?**

Будь-який патерн проектування, використовуваний при розробці інформаційних систем, являє собою формалізований опис, який часто зустрічається в завданнях проектування, вдале рішення даної задачі, а також рекомендації по застосуванню цього рішення в різних ситуаціях. Крім того, патерн проектування обов’язково має загальновживане найменування. Правильно сформульований патерн проектування дозволяє, відшукавши одного разу вдале рішення, користуватися ним знову і знову. Варто підкреслити, що важливим початковим етапом при роботі з патернами є адекватне моделювання розглянутої предметної області. Це є необхідним як для отримання належним чином формалізованої постановки задачі, так і для вибору відповідних патернів проектування.Відповідне використання патернів проектування дає розробнику ряд незаперечних переваг. Наведемо деякі з них:

- Модель системи, побудована в межах патернів проектування, фактично є структурованим виокремленням тих елементів і зв’язків, які значимі при вирішенні поставленого завдання.
- Модель, побудована з використанням патернів проектування, більш проста і наочна у вивченні, ніж стандартна модель. Проте, незважаючи на простоту і наочність, вона дозволяє глибоко і всебічно опрацювати архітектуру розроблюваної системи з використанням спеціальної мови.

Застосування патернів проектування підвищує стійкість системи до зміни вимог та спрощує неминуче подальше доопрацювання системи. Крім того, важко переоцінити роль використання патернів при інтеграції інформаційних систем організації. Також слід зазначити, що сукупність патернів проектування, по суті, являє собою єдиний словник проектування, який, будучи уніфікованим засобом, незамінний для спілкування розробників один з одним.
Таким чином, шаблони представляють собою, підтверджені роками розробок в різних компаніях і на різних проектах, «ескізи» архітектурних рішень, які зручно застосовувати у відповідних обставинах.

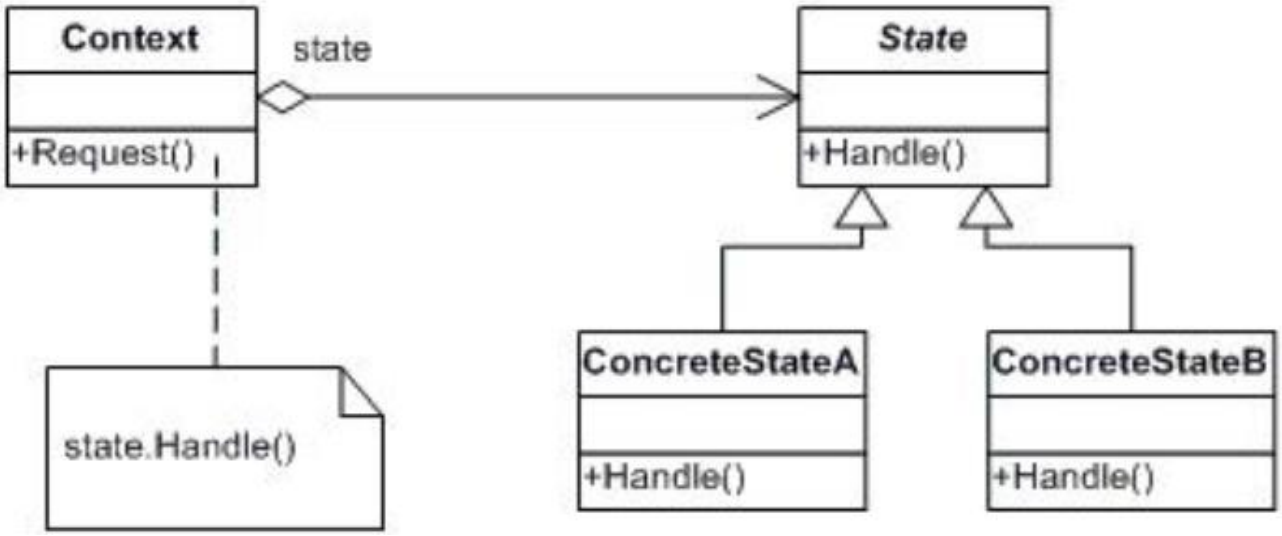
**Навіщо використовувати шаблони?**

Застосування шаблонів проектування не гарантує, що розроблена архітектура буде кристально чистою і зручною з точки зору програмування. Однак у потрібних місцях застосування шаблонів дозволить досягти наступних вигод:

- Зменшення трудовитрат і часу на побудову архітектури;
- Надання проектованій системі необхідних якостей (гнучкість, адаптованість тощо);
- Зменшення накладних витрат на подальшу підтримку системи;
- Та інші;

Варто також зазначити, що знання шаблонів проектування допомагає не тільки архітекторам програмних систем, але й розробникам. Коли кожна людина в команді знає значення і властивості шаблонів, архітекторові простіше донести загальну ідею архітектури системи, а розробникам - простіше зрозуміти.
Оскільки, врешті-решт, кожен бізнес зводиться до грошей, шаблони проектування також є економічно виправданим вибором між побудовою власного «колеса» та реалізацією закріплених і гарантованих спільнотою розробників практик і підходів. Це, звичайно ж, не означає, що їх необхідно використовувати в кожному проекті на кожну вимогу. Підходи не є догмою, їх потрібно використовувати з головою.

**Шаблон «State». Структура:**




Призначення: Шаблон «State» (Стан) дозволяє змінювати логіку роботи об'єктів у випадку зміни їх внутрішнього стану. Наприклад, відсоток нарахованих на картковий рахунок грошей залежить від стану картки: Visa Electron, Classic, Platinum і т.д. Або обсяг послуг, які надані хостинг компанією, змінюється в залежності від обраного тарифного плану (стану членства - бронзовий, срібний або золотий клієнт). Реалізація даного шаблону полягає в наступному: пов'язані зі станом поля, властивості, методи і дії виносяться в окремий загальний інтерфейс (State); кожен стан являє собою окремий клас (ConcreteStateA, ConcreteStateB), які реалізують загальний інтерфейс.

Об’єкти, що мають стан (Context), при зміні стану просто записують новий об'єкт в поле state, що призводить до повної зміни поведінки об'єкта.

Це дозволяє легко додавати в майбутньому і обробляти нові стани, відокремлювати залежні від стану елементи об'єкта в інших об'єктах, і відкрито проводити заміну стану (що має сенс у багатьох випадках).

# 2. Реалізація не менше 3-х класів відповідно до обраної теми.

## 2.1 Структура класів

Структура проекту з реалізованими класами зображена на рисунку 1.

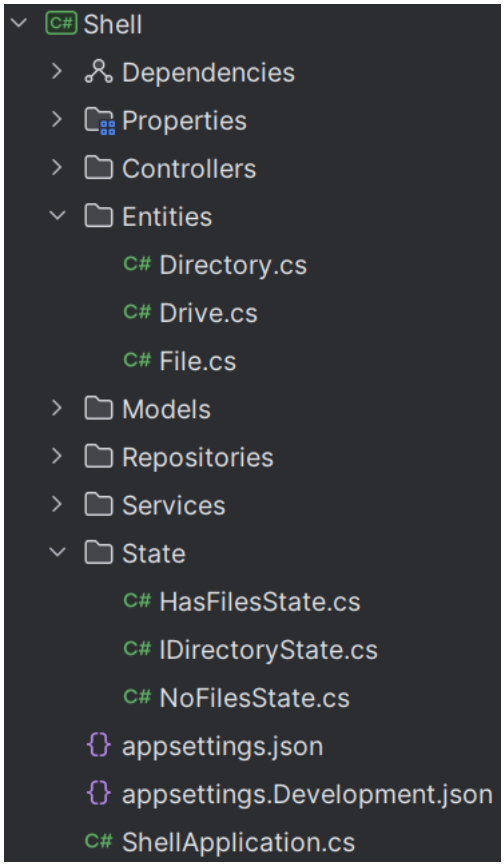



Рисунок 1 – структура проекту

## 2.2 Опис класів

Під час виконання лабораторної роботи було реалізовано функціонал 4 класів. Детальний опис кожного з класів:

**Directory:** Цей клас є контекстом, який зберігає поточний стан (IDirectoryState) і делегує виконання логіки активному стану.

**IDirectoryState:** Це інтерфейс, який визначає спільний контракт для різних станів папки (HasFilesState, NoFilesState). Він має метод Handle, який реалізують класи станів. Кожен з них керує своїм станом та виконує дії відповідно до нього.

**HasFilesState:** Цей стан відповідає за обробку папки, якщо вона містить файли. Він перевіряє наявність файлів та якщо вони відсутні – переводить папку у відповідний стан.

**NoFilesState:** Цей стан відповідає за обробку каталогу, якщо він не містить файли. Він перевіряє наявність файлів та якщо вони присутні – переводить папку у відповідний стан.

# 3. Реалізація шаблону за обраною темою

## 3.1 Опис шаблону

Для реалізації функціональності файлової оболонки я використав шаблон проектування State (Стан). Цей патерн дозволив динамічно змінювати поведінку об'єкта в залежності від його поточного стану.

**Опис реалізації:**

У папки в файловій оболонці є два основні стани:

- HasFilesState – стан коли папка не містить файлів.
- NoFilesState – стан коли папка містить файли.

Кожен із цих станів має свою роль: коли створюється, копіюється, оновлюється, видаляється файл, папка делегує її стану обробити цю операцію і відповідно до неї стан може змінитись на інший.

**Проблема, яку допоміг вирішити шаблон State:**

До використання цього патерну, код для керування станами міг би бути складним та переплутаним з багатьма умовними конструкціями if-else або switch-case, де кожний метод сервісу потребував би перевірки поточного стану папки. Це б призвело до поганої підтримуваності та складності розширення.

Використовуючи шаблон State, вдалося:

- Розділити логіку різних станів в окремі класи, що зробило код зрозумілішим і чистішим.
- Легко додавати нові стани або змінювати існуючі без необхідності модифікувати базовий код системи.
- Зробити поведінку кожного стану інкапсульованою, з чітко визначеними обов'язками.

**Переваги застосування шаблону State:**

- Зручне додавання нових станів: Якщо з'являться нові стани для папки (наприклад, має чи не має дочірні папки), їх легко додати, не змінюючи базовий код.
- Зменшення кількості умовних конструкцій: Логіка керування станами розподіляється між класами, що дозволяє уникнути громіздких умовних операторів у методах.
- Покращена підтримуваність та розширюваність: Кожен стан реалізований у своєму класі, що полегшує підтримку та розширення системи.

## 3.2 Діаграма класів

Діаграма класів, які реалізують шаблон Стан(State) зображена на рисунку 2.

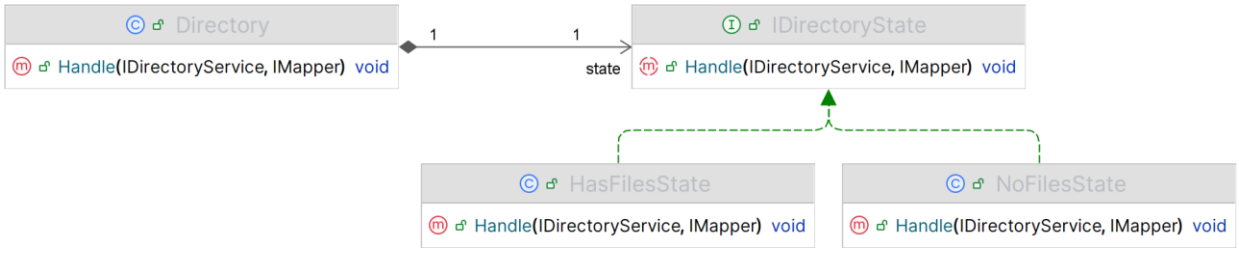



Рисунок 2 – реалізація шаблону Стан(State)

Посилання на код: https://github.com/ivanchikk/trpz

**Висновок:** виконавши дану лабораторну роботу, я використав шаблон проектування Стан(State) для файлової оболонки, за допомогою якого була впроваджена ефективна зміна станів папок залежно від обставин. Були реалізовані інтерфейс та класи станів IDirectoryState, HasFilesState, NoFilesState. Завдяки цьому шаблону вдалося зробити файлову оболонку більш гнучкою і легкою у підтримці та розширенні, бо логіка кожного стану була винесена в окремий клас. Це дозволило уникнути використання багатьох умовних операторів, що призвело б до заплутаного коду, що значно покращило читабельність.