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

Технології розроблення програмного забезпечення

ШАБЛОНИ «SINGLETON», «ITERATOR», «PROXY», «STATE», «STRATEGY».

FTP-server

Виконав
студент групи ІА–22:
Білокур Євгеній
Перевірив:
Мягкий Михайло Юрійович

Київ 2024

Зміст

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

Мета: Ознайомитися з основними паттернами проектування «Singleton», «Iterator», «Proxy», «State» та «Strategy». Вивчити їх структуру, особливості використання, а також застосувати на практиці.

Хід роботи

..22 FTP-server (state, builder, memento, template method, visitor, client-server)

FTP-сервер повинен вміти коректно обробляти і відправляти відповіді по протоколу FTP, з можливістю створення користувачів (з паролями) і доступних їм папок, розподілу прав за стандартною схемою (rwe), ведення статистики з'єднань, обмеження максимальної кількості підключень і максимальної швидкості поширення глобально і окремо для кожного облікового запису.

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?

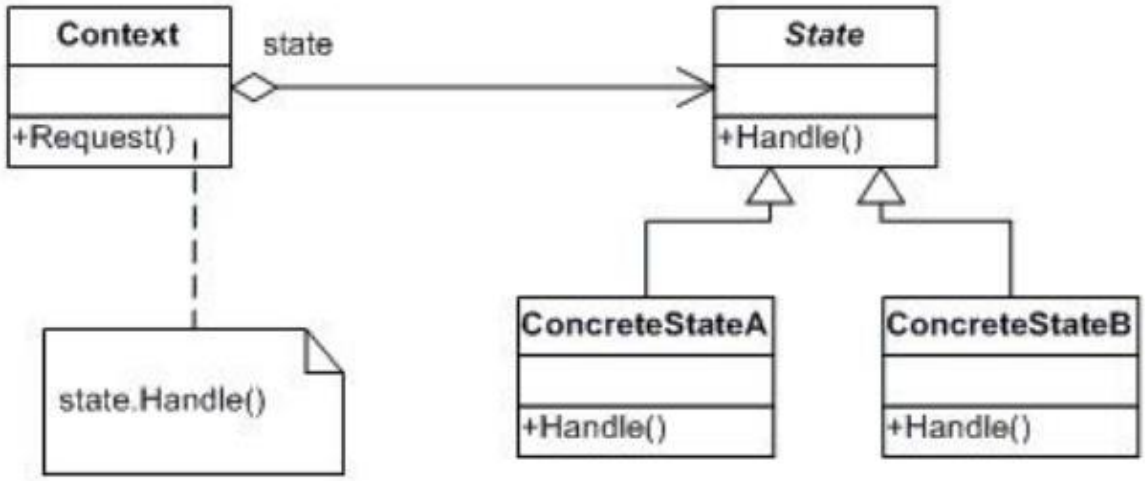
Застосування шаблонів проектування не гарантує, що розроблена архітектура буде кристально чистою і зручною з точки зору програмування. Однак у потрібних місцях застосування шаблонів дозволить досягти наступних вигод:

- 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).

Реалізувати не менше 3-х класів відповідно до обраної теми

Структура класів

Структура проекту з реалізованими класами зображена на рисунку 1

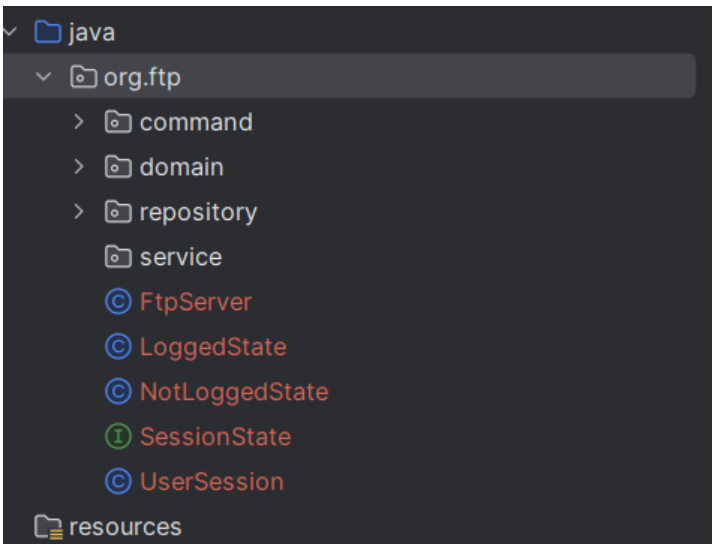


Рисунок 1 – Структура реалізованих класів

Опис класів

Під час виконання лабораторної роботи було реалізовано функціонал 5 класів, які зображенні на рисунку 1. Детальний опис кожного з класів:

FtpServer: Цей клас відповідає за головний сервер FTP, який обробляє підключення клієнтів, виконання команд і керування сесіями користувачів. Він є ядром всієї системи FTP-сервера, забезпечуючи зв'язок між різними компонентами та підтримку основних функцій протоколу FTP.

UserSession: Цей клас відповідає за управління сесією конкретного користувача. Він тримає інформацію про поточний стан користувача (увійшов чи не увійшов), і керує переходами між станами. Важливу роль відіграє зберігання даних користувача, таких як права доступу та поточний статус авторизації.

SessionState: Це інтерфейс , що визначає контракт для різних станів сесії користувача (наприклад, LoggedState і NotLoggedState). Класи, що його реалізують, відповідають за поведінку користувача в різних станах і керують тим, які дії дозволені або заборонені в рамках сесії.

LoggedState: Цей клас реалізує стан користувача, який вже увійшов у систему. У цьому стані користувач має доступ до певних команд FTP, таких як завантаження/вивантаження файлів, перегляд каталогів тощо. Клас визначає, як обробляються запити від користувачів, які пройшли авторизацію.

NotLoggedState: Цей клас відповідає за стан користувача, який ще не увійшов у систему. Він обмежує доступ користувача до команд FTP, дозволяючи лише базові операції, такі як спроби авторизації (login) або запити на інформацію, доступну до авторизації.

Реалізувати один з розглянутих шаблонів за обраною темою

Опис шаблону

Для реалізації функціональності мого FTP-сервера я використав шаблон проектування **State (Стан)**. Цей паттерн дозволив динамічно змінювати поведінку об'єкта в залежності від його поточного стану.

Опис реалізації:

У FTP-сервері існує два основні стани користувача:

1. **NotLoggedState** — стан неавторизованого користувача.
2. **LoggedState** — стан авторизованого користувача.

Кожен із цих станів має свої особливості: неавторизований користувач обмежений у виконанні команд, наприклад, йому доступні тільки команди авторизації, тоді як авторизований користувач має доступ до повного набору команд FTP-сервера (наприклад, управління файлами, перегляд каталогів). Для управління цими станами був створений інтерфейс **SessionState**, який визначає спільний інтерфейс для всіх станів.

Проблема, яку допоміг вирішити шаблон State:

До використання цього паттерну, код для керування станами міг би бути складним та переплутаним з багатьма умовними конструкціями if-else або switch-case, де кожна команда потребувала б перевірки поточного стану користувача. Це б призвело до поганої підтримуваності та складності розширення.

Використовуючи паттерн **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 паттерну State:

1. **Зручне додавання нових станів**: Якщо з'являться нові стани для користувача (наприклад, обмежений стан або стан очікування), їх легко додати, не змінюючи базовий код.
2. **Зменшення кількості умовних конструкцій**: Логіка керування станами розподіляється між класами, що дозволяє уникнути громіздких умовних операторів у методах.

3. **Покращена підтримуваність та розширюваність**: Кожен стан реалізований у своєму класі, що полегшує підтримку та розширення системи.

Діаграма класів

Діаграма класів, які реалізовують паттерн Стан(State) зображена на рисунку 2

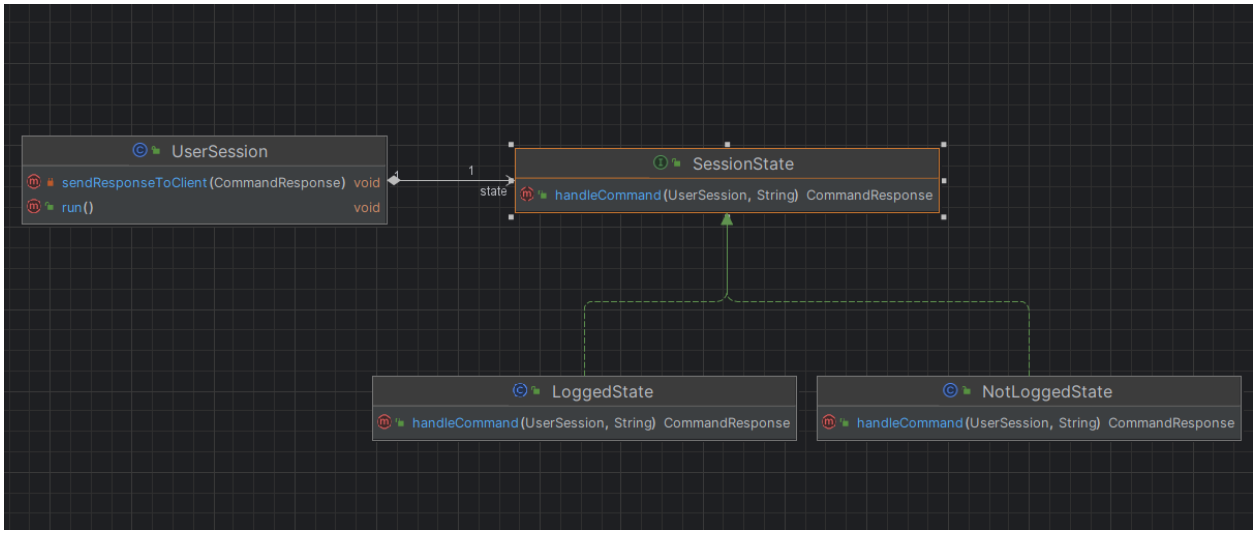


Рисунок 2 – реалізація паттерну Стан(State)

Висновки та код

Код можна знайти за посиланням - https://github.com/B1lok/trpz

Висновок: У ході виконання лабораторної роботи було реалізовано декілька класів для побудови FTP-сервера, зокрема класи для обробки станів користувача, таких як **LoggedState** і **NotLoggedState**. Для ефективного управління різними станами користувача був використаний шаблон проектування **State (Стан)**. Завдяки цьому паттерну вдалося зробити сервер більш гнучким та легким у підтримці, адже логіка кожного стану інкапсульована в окремому класі. Це дозволило уникнути заплутаного коду з великою кількістю умовних операторів, що значно покращило читабельність і розширюваність системи.