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 №**7**
з предмету «Технології розроблення програмного забезпечення»
на тему «**Шаблони “Mediator”, “Facade”, “Bridge”, “Template Method”**»

Виконав
студент групи ІА-23:
Воронюк Є. В.

Перевірив:
Мягкий М. Ю.

Київ 2024

Зміст

Мета: реалізація частини функціоналу робочої програми у вигляді класів та їхньої взаємодії із застосуванням одного із розглянутих шаблонів проектування.

Завдання 1: ознайомитися з короткими теоретичними відомостями.
Завдання 2: реалізувати частину функціонала робочої програми у вигляді класів і їх взаємодій для досягнення конкретних функціональних можливостей.
Завдання 3: застосування одного з даних шаблонів при реалізації програми.

Тема для виконання:

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.

Короткі теоретичні відомості:

Шаблон «Mediator»

Шаблон «Mediator» (посередник) використовується для визначення взаємодії об'єктів за допомогою іншого об'єкта (замість зберігання посилань один на одного). Даний шаблон схожий на шаблон «команда», проте в даному випадку замість зберігання даних про конкретну дію, зберігаються дані про взаємодії між компонентами.

Даний шаблон зручно застосовувати у випадках, коли безліч об'єктів взаємодіє між собою деяким структурованим чином, однак складним для розуміння. У такому випадку вся логіка взаємодії виноситься в окремий об'єкт. Кожен із взаємодіючих об'єктів зберігає посилання на об'єкт «медіатор».

«Медіатор» нагадує диригента при управлінні оркестром. Диригент стежить за тим, щоб кожен інструмент грав в правильний час і в злагоді з іншими інструментами. Функції «медіатора» повністю це повторюють.

Переваги та недоліки:

+ Усуває залежності між компонентами, дозволяючи повторно їх використовувати.
+ Спрощує взаємодію між компонентами.
+ Централізує управління в одному місці.
- Посередник може сильно роздутися.

Шаблон «Facade»

Шаблон «Facade» (фасад) передбачає створення єдиного уніфікованого способу доступу до підсистеми без розкриття внутрішніх деталей підсистеми. Оскільки підсистема може складатися з безлічі класів, а кількість її функцій - не більше десяти, то щоб уникнути створення «спагеті-коду» (коли все тісно пов'язано між собою) виділяють один загальний інтерфейс доступу, здатний правильним чином звертатися до внутрішніх деталей.

Це також відволікає користувачів від змін в підсистемі (внутрішня реалізація може змінюватися, а наданої послуги немає), що також скоротить кількість змін в використовуваних фасад класах (без фасаду довелося б змінювати вихідні коди в безлічі точок).

Звичайно, твердої умови повного закриття внутрішніх класів підсистеми не стоїть - при необхідності можна звертатися до окремих класів безпосередньо, минаючи об'єкт фасад.

Переваги та недоліки:

+ Ізолює клієнтів від компонентів складної підсистеми.
- Фасад ризикує стати божественним об'єктом, прив'язаним до всіх класів програми.

Шаблон «Bridge»

Шаблон «Bridge» (міст) використовується для поділу інтерфейсу і його реалізації. Це необхідно у випадках, коли може існувати кілька різних абстракцій, над якими можна проводити дії різними способами.

В даному випадку при появі нових типів вікон буде необхідність створення нових класів їх реалізацій. Це веде до стрімкого збільшення кількості класів і плутанини в вихідному коді.

Таким чином можна відокремити абстракцію від її реалізацій, і додавати нові абстракції незалежно від реалізацій.

Переваги та недоліки:

+ Дозволяє будувати платформо-незалежні програми.
+ Приховує зайві або небезпечні деталі реалізації від клієнтського коду.
+ Реалізує принцип відкритості / закритості.
- Ускладнює код програми через введення додаткових класів.

Шаблон «Template Method»

Шаблон «Template Method» (шаблонний метод) дозволяє реалізувати покроково алгоритм в абстрактному класі, але залишити специфіку реалізації подклассам. Можна привести в приклад формування веб-сторінки: необхідно додати заголовки, вміст сторінки, файли, що додаються, і нижню частину сторінки. Код для додавання вмісту

сторінки може бути абстрактним і реалізовуватися в різних класах - AspNetCompiler, HtmlCompiler, PhpCompiler і т.п. Додавання всіх інших елементів виконується за допомогою вихідного абстрактного класу з алгоритмом.

Структура:

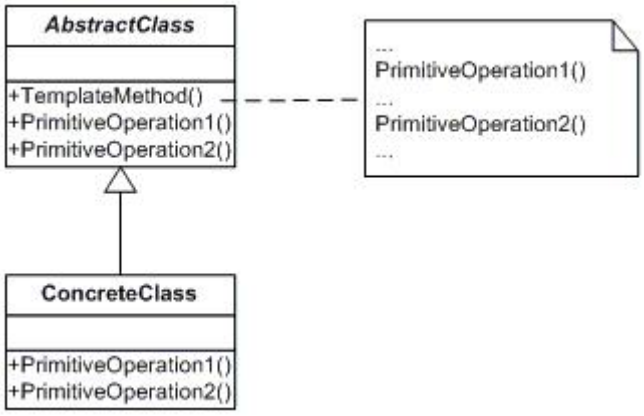

Рисунок 1 – структура шаблону «Template Method»

Даний шаблон дещо нагадує шаблон «фабричний метод», однак область його використання абсолютно інша - для покрокового визначення конкретного алгоритму; більш того, даний шаблон не обов'язково створює нові об'єкти - лише визначає послідовність дій.

Приклад:

Будівельники використовують підхід, схожий на шаблонний метод, при будівництві типових будинків. У них є основний архітектурний проект, в якому розписані кроки будівництва: заливка фундаменту, споруда стін, перекриття даху, встановлення вікон і так далі.

Але, незважаючи на стандартизацію кожного етапу, будівельники можуть вносити невеликі зміни на будь-якому з етапів, щоб зробити будинок трішки несхожим на інші.

Переваги та недоліки:

+ Полегшує повторне використання коду.

- Ви жорстко обмежені скелетом існуючого алгоритму.

- Ви можете порушити принцип підстановки Барбари Лісков, змінюючи базову поведінку одного з кроків алгоритму через підклас.

- З ростом кількості кроків шаблонний метод стає занадто складно підтримувати.

Хід роботи::

Реалізація патерну:

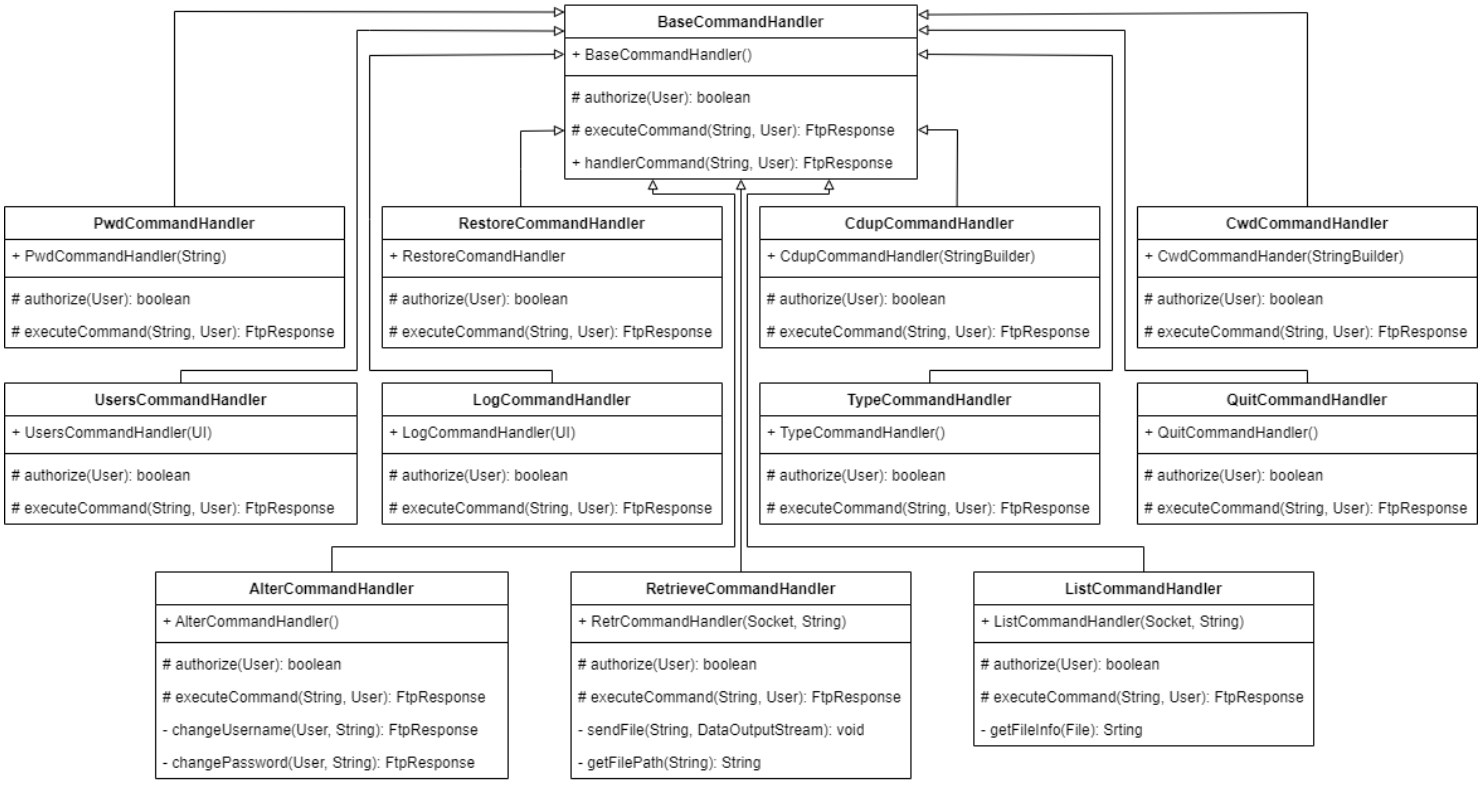

Рисунок 2 – реалізація патерну Template Method

Опис реалізації патерну:

Шаблонний метод використовується для реалізації різних варіацій алгоритму виконання команд на FTP-сервері. Абстрактний клас містить загальну структуру алгоритму та оголошує абстрактні методи, які підкласи повинні реалізувати. Абстрактний клас може визначати послідовність авторизації, перевірку прав користувача та виконання команди. Кожен конкретний клас представляє реалізацію абстрактного класу для конкретної команди. Наприклад, клас для команди "завантаження файлу" реалізує специфічну логіку для цієї операції.

Використання Шаблонного методу дозволяє уникнути дублювання коду та визначити загальну логіку виконання команд та відкриває можливість для розширення та модифікації – можна додавати реалізація обробки нових команд. Отже, застосування шаблону Template Method забезпечує гнучкість та підтримку можливого майбутнього розширення функціоналу.

Опис класів та логіка роботи:

Абстрактний клас BaseCommandHandler визначає основну структуру виконання команди через методи authorize і executeCommand. Обробники команд успадковують цю поведінку і реалізують специфічну логіку для кожної команди.

AlterCommandHandler реалізує зміну імені користувача чи пароля через команду ALTER. При перевірці параметрів метод обробляє два варіанти: username для оновлення імені та password для зміни пароля. Клас використовує шаблон "збереження стану" через об'єкти UserMemento для відкату змін за необхідності.

CdupCommandHandler відповідає за перехід до батьківського каталогу (CDUP). Клас змінює шлях currentDirectoryPath і повертає відповідь, залежно від того, чи існує батьківський каталог.

CwdCommandHandler реалізує команду CWD для зміни робочого каталогу. Клас перевіряє існування каталогу, змінює шлях і враховує доступ адміністратора до операції.

PwdCommandHandler надає інформацію про поточний каталог через команду PWD. Шлях передається у відповідь FTP-серверу.

QuitCommandHandler реалізує вихід користувача з FTP-сервера через команду QUIT. Він викликає сервіс аутентифікації для завершення сесії і повертає відповідний код.

RestoreCommandHandler відповідає за відновлення стану користувача (RESTORE). Клас працює з UserMemento для відкату до попереднього стану користувача, якщо такий існує.

RetrieveCommandHandler реалізує завантаження файлів через команду RETR. Логіка включає читання файлів і передачу даних через сокет, а також обробку помилок, якщо файл недоступний.

TypeCommandHandler обробляє команди зміни типу передачі даних (TYPE). Клас підтримує два типи: ASCII (A) та Image (I), і повертає помилку для інших параметрів.

UsersCommandHandler забезпечує команду USERS, яка вимагає список користувачів для адміністратора. Клас використовує сервіс користувачів для отримання даних та UI для їхнього відображення.

ListCommandHandler реалізує команду LIST (список файлів). Містить один приватний метод: getFileInfo(File): String - отримання інформації про файли.

LogCommandHandler відповідає за обробку команди для перегляду вмісту лог-файлу FTP-сервера. Його основні функції включають перевірку авторизації користувача та відображення логів. Метод authorize дозволяє виконання команди тільки для авторизованих користувачів з правами адміністратора. Метод executeCommand зчитує вміст лог-файлу за допомогою FileHandler.getLogFileContent() і передає його для відображення через об'єкт UI. Після цього повертається відповідь FTP-сервера з кодом 212, який сигналізує про успішну операцію. Обробник дозволяє адміністраторам отримувати доступ до логів для моніторингу чи діагностики роботи сервера.

Вихідний код системи:
Вихідний код системи розміщено на GitHub: https://github.com/dobrogo-dnia/SDT-labs

Висновок: під час виконання даної лабораторної роботи було реалізовано частину функціоналу робочої програми у вигляді класів та їхньої взаємодії із застосуванням одного із розглянутих шаблонів проектування – а саме шаблону Template Method. Існуючий код було оновлено для використання відповідного патерну.