Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

# ЛАБОРАТОРНА РОБОТА №6

з дисципліни «Технології розроблення програмного забезпечення»

Тема: «Патерни проектування. FTP-server»

**Виконав:**
студент групи ІА-34
Сухоручкін Гліб

**Перевірив:**
Асистент кафедри ІСТ
Мягкий М.Ю.

Київ – 2025

**Зміст:**

**Вступ**

Метою цієї роботи є вивчити структуру шаблонів «Abstract Factory», «Factory Method», «Memento», «Observer», «Decorator» та навчитися застосовувати їх в реалізації програмної системи.

Проєкторваною системою цієї лабороторної роботи є FTP-server (використовуючи 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**

**Шаблон «Memento»**

**Призначення**: Шаблон використовується для збереження і відновлення стану об'єктів без порушення інкапсуляції. Об'єкт «Memento» служить виключно для збереження змін над початковим об'єктом (Originator). Лише початковий об'єкт має можливість зберігати і отримувати стан об'єкту «Memento» для власних цілей, цей об'єкт є «порожнім» для кого-небудь ще. Об'єкт «Caretaker» використовується для передачі і зберігання мементо об'єктів в системі.

Таким чином вдається досягти наступних цілей:

- зберігання стану повністю відділяється від початкових об'єктів, що полегшує їх реалізацію;
- передача об'єктів «Memento» лягає на плечі Caretaker об'єктів, що дозволяє гнучкіше управляти станами об'єктів і спростити дизайн класів початкових об'єктів;
- збереження і відновлення стану реалізовані у вигляді двох простих методів і є закритими для кого-небудь ще окрім початкових об'єктів, таким чином не порушуючи інкапсуляцію.

Шаблон «Мементо» дуже зручно використати разом з шаблоном «Команда» для реалізації «скасовних» дій – дані про дію зберігаються в мементо, а команда має можливість вважати і відновити початкове положення відповідних об'єктів.

**Переваги та недоліки**:

+ Не порушує інкапсуляцію вихідного об’єкта.

+ Спрощує структуру вихідного об’єкта. Не потрібно зберігати історію версій свого стану.

- Вимагає багато пам’яті, якщо клієнти дуже часто створюють знімки.

- Може спричинити додаткові витрати пам’яті, якщо об’єкти, що зберігають історію, не звільняють ресурси, зайняті застарілими знімками.

## Хід роботи

## Діаграма класів

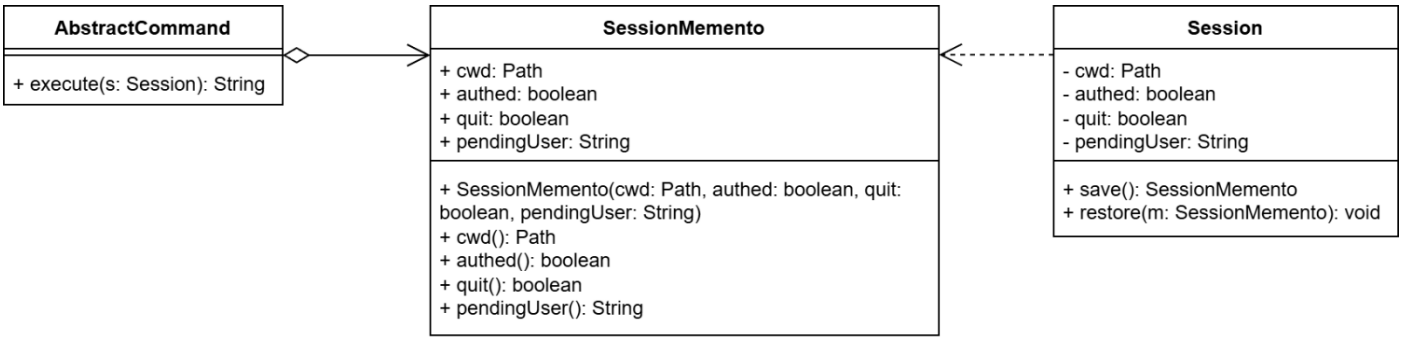



Рис 1. Патерн «Memento» у FTP-сервері.

## Пояснення діаграми классів

Session – originator: зберігає внутрішній стан сесії (cwd, authed, quit, pendingUser) і сам уміє робити «знімок» (save()) та відновлення (restore(m)). Інкапсуляція не порушується – лише Session знає, що саме слід зберігати/відновлювати.

SessionMemento – memento: незмінний «знімок» стану сесії з полями cwd, authed, quit, pendingUser; має конструктор і гетери. Не містить логіки – лише дані для відновлення.

AbstractCommand – caretaker: керує життєвим циклом знімка під час execute(s): робить save() перед діями, викликає виконання, а у разі помилки – restore(m) і повертає помилкову FTP-відповідь. Не заглядає всередину SessionMemento, лише зберігає та передає його назад у Session.

Зв’язки на діаграмі: стрілка від AbstractCommand до SessionMemento показує, що команда використовує знімок під час виконання; пунктир від SessionMemento до Session – відновлення стану.

## Вихідний код

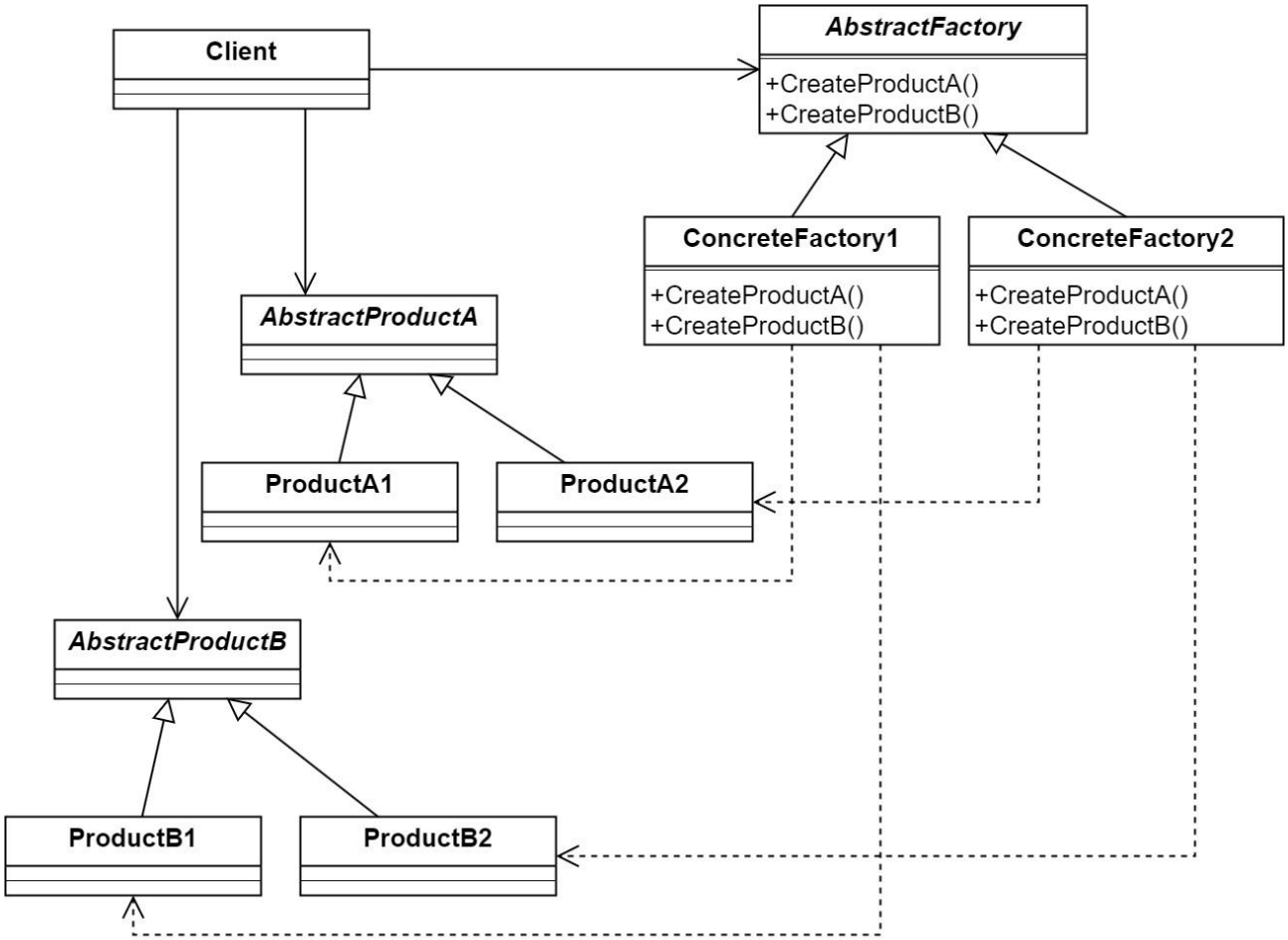
https://github.com/GlebTiKsTRPZ/lab6

**Відповіді на питання до лабораторної роботи**

**1. Яке призначення шаблону «Абстрактна фабрика»?**

Створює сімейства узгоджених об’єктів без вказівки їхніх конкретних класів (є спільний інтерфейс фабрики та реалізації під різні сімейства продуктів).

**2. Нарисуйте структуру шаблону «Абстрактна фабрика».**



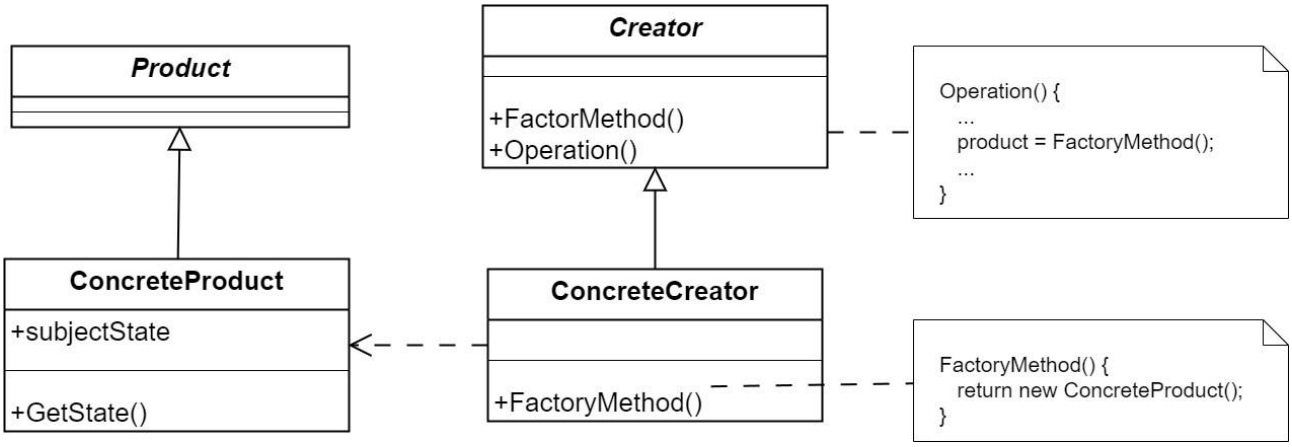

**3. Які класи входять в шаблон «Абстрактна фабрика», та яка між ними взаємодія?**

Типові учасники: AbstractFactory (оголошує методи створення продуктів), ConcreteFactoryX/Y (створюють конкретні продукти певного стилю/сімейства), AbstractProductA/B (спільні інтерфейси продуктів), ConcreteProductAX/BX (конкретні реалізації), Client (працює лише з абстракціями; отримує від фабрики узгоджені продукти). Клієнт отримує фабрику й через її методи створює продукти одного сімейства; взаємозамінність стилів досягається підміною конкретної фабрики.

**4. Яке призначення шаблону «Фабричний метод»?**

Визначає інтерфейс створення об’єктів певного базового типу, дозволяючи підставляти підтипи (інакше кажучи, “віртуальний конструктор”).

**5. Нарисуйте структуру шаблону «Фабричний метод».**




**6. Які класи входять в шаблон «Фабричний метод», та яка між ними взаємодія?**

Creator (абстрактний) оголошує фабричний метод createProduct(); ConcreteCreator перевизначають його й повертають ConcreteProduct, що реалізують Product (абстрактний інтерфейс). Клієнт працює з Product, тоді як вибір конкретного класу інкапсульовано у ConcreteCreator.

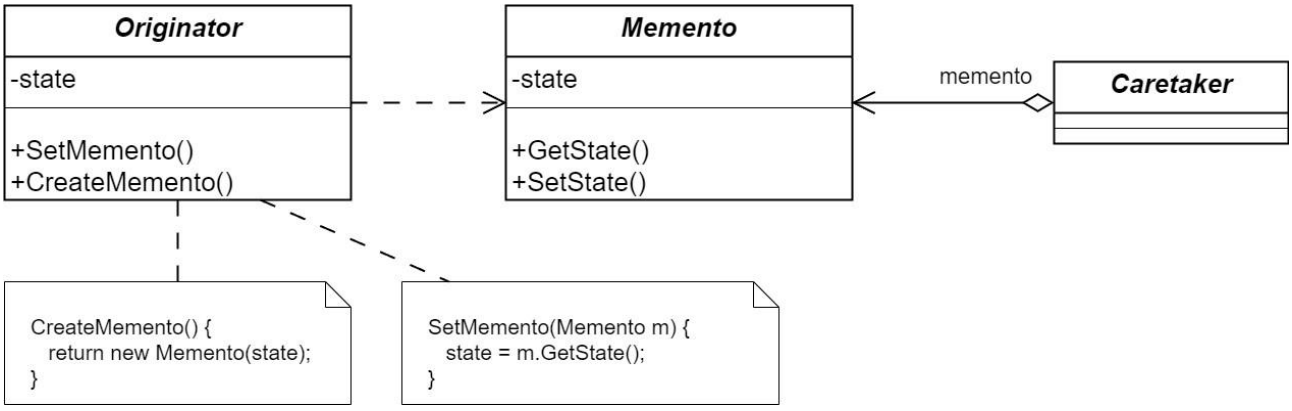
**7. Чим відрізняється шаблон «Абстрактна фабрика» від «Фабричний метод»?**

- Абстрактна фабрика створює сімейства узгоджених продуктів; Фабричний метод створює один продукт певного типу.
- Абстрактна фабрика зазвичай поєднує багато фабричних методів у одному інтерфейсі фабрики; Фабричний метод робить акцент на наслідуванні творця й перевизначенні одного методу.
- Абстрактна фабрика легко додає нові сімейства (нові фабрики), але важко додавати нові типи продуктів; Фабричний метод полегшує додавання нових продуктів через нові підкласи творця, але не забезпечує узгодженість “родин” продуктів.

**8. Яке призначення шаблону «Знімок»?**

Зберігання й відновлення стану об’єкта без порушення інкапсуляції.

**9. Нарисуйте структуру шаблону «Знімок».**



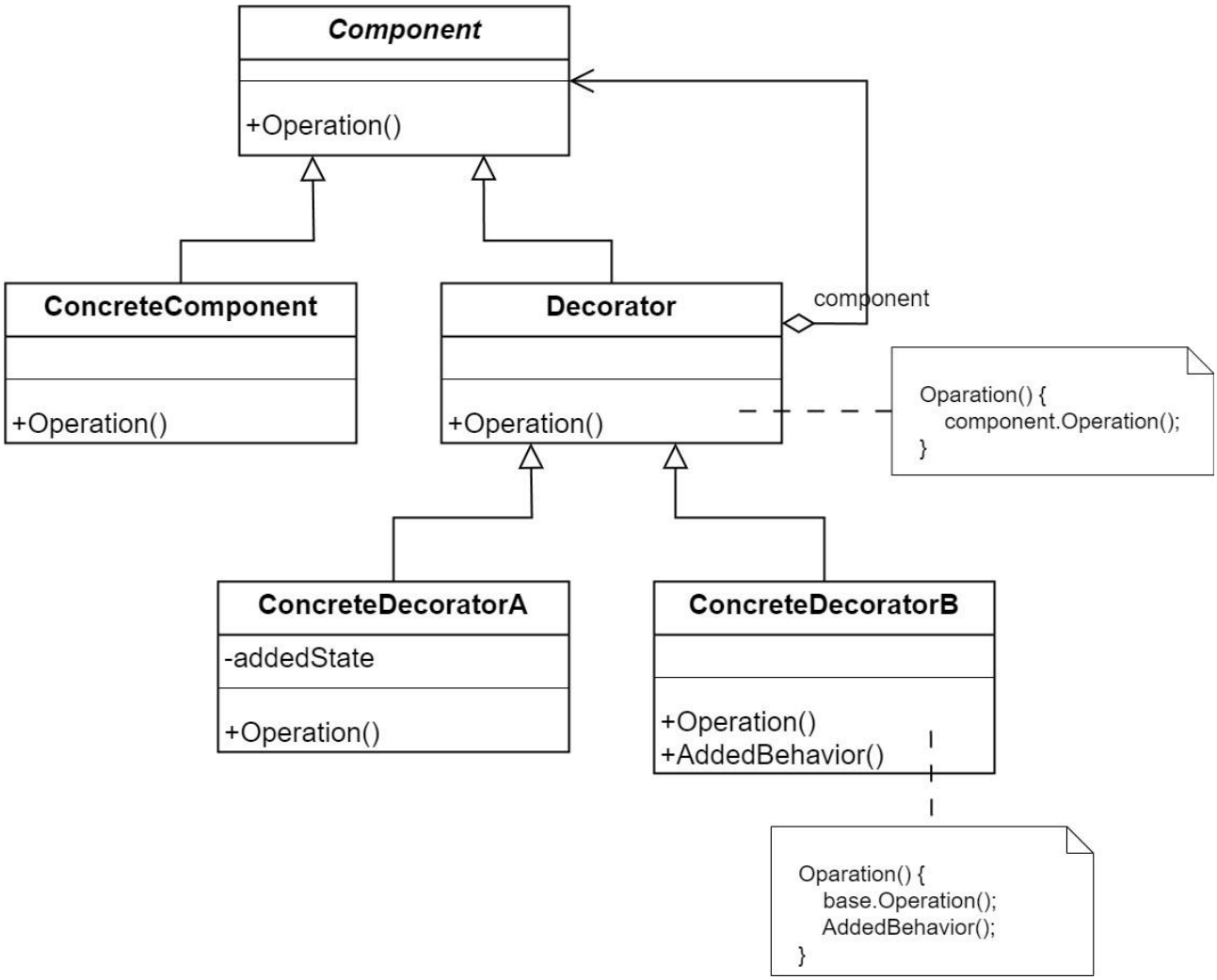

**10. Які класи входять в шаблон «Знімок», та яка між ними взаємодія?**

Originator створює знімки свого стану та відновлює його з Memento; Memento інкапсулює стан і недоступний зовнішньо; Caretaker зберігає/передає знімки, не читаючи їх вміст.

**11. Яке призначення шаблону «Декоратор»?**

Динамічно додає обов’язки (поведінку) окремим об’єктам під час виконання, “обгортаючи” початковий об’єкт зі збереженням його функцій; гнучкіший за спадкування.

**12. Нарисуйте структуру шаблону «Декоратор».**




**13. Які класи входять в шаблон «Декоратор», та яка між ними взаємодія?**

Component (інтерфейс), ConcreteComponent (базовий об’єкт), Decorator (базовий обгортчик, містить посилання на Component і делегує виклики), ConcreteDecoratorA/B (додають поведінку до делегування). Декоратори можна накладати шарами, комбінуючи ефекти.

**14. Які є обмеження використання шаблону «декоратор»**

Багато дрібних класів; складно конфігурувати об’єкти, загорнуті в кілька обгорток одночасно.

**Висновок**

У цій лабораторній роботі вивчено й застосовано шаблон «Memento» у контексті FTP-сервера: Session (Originator) створює/відновлює SessionMemento, а AbstractCommand виконує роль Caretaker для безпечного виконання та можливого відкату дій.

Така можливість «відкату» сесії робить роботу FTP-сервера більш надійною для користувачів: у випадку помилки команди менше ризиків пошкодити критичні дані чи залишити сесію в неконсистентному стані, зменшується обсяг ручних виправлень адміністраторів та кількість інцидентів, пов’язаних із невдалими операціями.