



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 №6
Технології розроблення програмного забезпечення
*«Шаблони «Abstract Factory», «Factory Method»,
«Memento», «Observer», «Decorator»»*

Виконала
студентка групи ІА-14:

Логінова І.С.

Перевірив:

Мягкий М.Ю.

Київ 2023

Тема: Шаблони «Abstract Factory», «Factory Method», «Memento», «Observer», «Decorator».

Завдання:

1. Ознайомитися з короткими теоретичними відомостями.
2. Реалізувати частину функціоналу робочої програми у вигляді класів та їх взаємодій для досягнення конкретних функціональних можливостей.
3. Застосування одного з розглянутих шаблонів при реалізації програми.

Варіант:

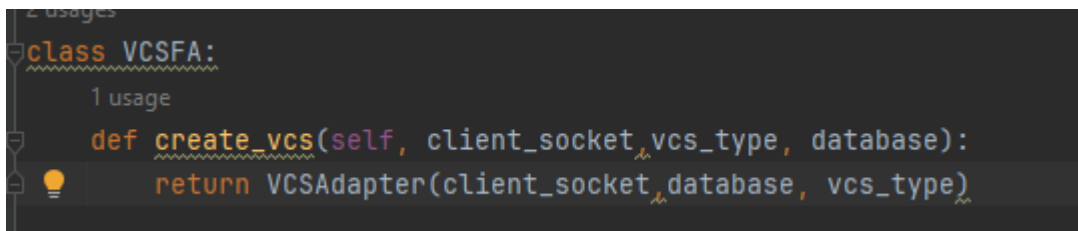
10. VCS all-in-one (iterator, adapter, factory method, facade, visitor, p2p)

Клієнт для всіх систем контролю версій повинен підтримувати основні команди і дії (commit, update, push, pull, fetch, list, log, patch, branch, merge, tag) для 3-х основних систем управління версіями (svn, git, mercurial), а також мати можливість вести реєстр репозиторіїв (і їх типів) і відображати дерева фіксації графічно

Хід роботи

Паттерн Фабричний метод (Віртуальний конструктор, Factory Method)

Фабричний метод - це породжувальний патерн проектування, який визначає загальний інтерфейс для створення об'єктів у суперкласі, дозволяючи підкласам змінювати тип створюваних об'єктів.



```
class VCSFA:
    1 usage
    def create_vcs(self, client_socket, vcs_type, database):
        return VCSAdapter(client_socket, database, vcs_type)
```

Паттерн Фабричний метод (Factory Method) використовується для створення об'єктів різних типів, які реалізують інтерфейс VCSInterface. У цьому контексті паттерн Фабричний метод використовується для створення об'єктів, які представляють конкретні системи керування версіями, такі як Git, Mercurial та SVN.

Основна ідея полягає в тому, що підкласи GIT, Mercurial, та SVN реалізують методи інтерфейсу VCSInterface, а фабричний метод `create_vcs` в класі VCSFactory створює об'єкт відповідно типу VCS

Цей паттерн дозволяє додавати підтримку нових систем керування версіями без зміни вже існуючого коду. Клас VCSFA відповідає за створення об'єктів VCS, і, якщо з'явиться нова система керування версіями, достатньо лише додати новий клас, який реалізує інтерфейс VCSInterface.

Отже, паттерн Фабричний метод допомагає досягнути принципу "відкриття для розширення, закриття для зміни", дозволяючи додавати нові функціональності без зміни вже існуючого коду.

Висновок: при виконанні цієї лабораторної роботи я реалізувала частину функціоналу робочої програми у вигляді класів та їхньої взаємодії для досягнення конкретних функціональних можливостей із застосуванням паттерну Factory Method при реалізації програми.