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

ШАБЛОН «MEDIATOR», «FACADE»,
«BRIDGE», «TEMPLATE METHOD»

Варіант 10

Виконала
студентка групи ІА – 13:
Луценко Юлія Сергіївна

Перевірив:
Мягкий Михайло

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 Фасад(Facade)

Фасад - це структурний патерн проєктування, який надає простий інтерфейс до складної системи класів, бібліотеки або фреймворку.

VCSFacade - це реалізація фасаду, яка надає спрощений інтерфейс для виконання операцій з конкретною системою контролю версій. Користувач може використовувати команди, такі як commit, initialize_repository, watch_history, і інші, і не повинен знати деталей роботи підсистеми.

```python
class VCSFacade:
    new *
    def __init__(self, connection):
        self.connection = connection

    2 usages (1 dynamic)  new *
    def initialize_repository(self, vcs_type, repo_directory):
        version_control = VersionControlFactory.create_version_control_system(connection, vcs_type)
        version_control.initialize_repository(repo_directory, vcs_type)

    1 usage  new *
    def commit_changes(self, version_control, repo_name, file_name, message):
        version_control.commit(repo_name, file_name, message)

    2 usages (1 dynamic)  new *
    def watch_history(self, version_control, repo_name):
        version_control.watch_history(repo_name)
```

```python
        if action == "1":
            if version_control:
                file_name = input("Enter the file name: ")
                message = input("Enter the commit message: ")
                vcs_facade.commit_changes(version_control, repo_name, file_name, message)
            else:
                print("Version control not initialized.")
        elif action == "2":
            if version_control:
                vcs_facade.watch_history(version_control, repo_name)
            else:
                print("Version control not initialized.")
        elif action == "3":
            vcs_facade.show_repositories(vcs_type)
        elif action == "4":
            break
        else:
            print("Invalid choice. Please enter a valid option.")
elif choice == "5":
    print("Exiting...")
    break
```

Висновок: За допомогою паттерну Фасад ми приховуємо складність роботи з системами контролю версій і надаємо зручний інтерфейс для користувача, що дозволяє йому взаємодіяти з різними системами без необхідності знати деталі їхньої реалізації.