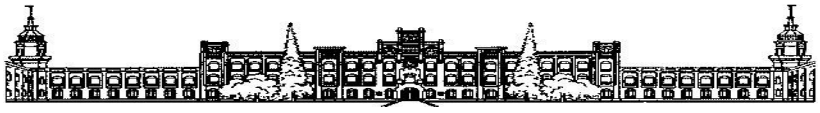

Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №7

# Технології розроблення програмного забезпечення

*«Шаблон «MEDIATOR», «FACADE», «BRIDGE», «TEMPLATE METHOD»»*

Виконала
студентка групи ІА-14:

Логінова І.С.

Перевірив:

Мягкий М.Ю.

Київ 2023

**Тема:** Шаблон «MEDIATOR», «FACADE», «BRIDGE», «TEMPLATE METHOD».

**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.

```python
class VCSF:
    def __init__(self, vcs_adapter):
        self.adapter = vcs_adapter

    1 usage (1 dynamic)
    def commit_changes(self, repo_path, comment):
        self.adapter.commit(repo_path, comment)

    1 usage (1 dynamic)
    def update_repository(self, repo_path):
        self.adapter.update(repo_path)

    1 usage (1 dynamic)
    def push_changes(self, repo_path):
        self.adapter.push(repo_path)

    1 usage (1 dynamic)
    def initialize_repository(self, repository_name, repo_path)
        self.adapter.init_repo(repository_name, repo_path)

    1 usage (1 dynamic)
    def view_commit_history(self, repo_path):
        self.adapter.log(repo_path)

    1 usage (1 dynamic)
    def view_repository_status(self, repo_path):
        self.adapter.status(repo_path)

    1 usage (1 dynamic)
    def add_files(self, repo_path, files):
        self.adapter.add(repo_path, files)

    1 usage (1 dynamic)
    def add_all_changes(self, repo_path):
        self.adapter.add_all(repo_path)

    1 usage (1 dynamic)
    def apply_patch(self, repo_path, patch_file_path):
        self.adapter.patch(repo_path, patch_file_path)

    1 usage (1 dynamic)
    def create_branch(self, repo_path, branch_name):
        self.adapter.branch(repo_path, branch_name)

    1 usage (1 dynamic)
    def merge_branch(self, repo_path, branch_name):
        self.adapter.merge(repo_path, branch_name)

    1 usage (1 dynamic)
    def create_tag(self, repo_path, tag_name):
        self.adapter.tag(repo_path, tag_name)

    2 usages (2 dynamic)
    def list(self, repo_path):
        self.adapter.list(repo_path)
```

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

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.

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 Facade при реалізації програми.