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 №4
# **ШАБЛОНИ «SINGLETON», «ITERATOR», «PROXY», «STATE», «STRATEGY»**

Виконала
студентка групи ІА – 13:
Майданюк Анастасія

Перевірив:
Мягкий М.Ю.

**Завдання:**

1. Ознайомитися з короткими теоретичними відомостями.

2. Реалізувати частину функціоналу робочої програми у вигляді класів та їхньої взаємодії

для досягнення конкретних функціональних можливостей.

3. Застосування одного з розглянутих шаблонів при реалізації програми

**Варіант:**

**..11 Web crawler (proxy, chain of responsibility, memento, template method, composite, p2p)**

Веб-сканер повинен вміти розпізнавати структуру сторінок сайту, переходити за посиланнями, збирати необхідну інформацію про зазначений термін, видаляти не семантичні одиниці (рекламу, об'єкти javascript і т.д.), зберігати знайдені дані у вигляді структурованого набору html файлів вести статистику відвіданих сайтів і метадані.

## Хід роботи

**Замісник** — це структурний патерн проектування, що дає змогу підставляти замість реальних об’єктів спеціальні об’єкти-замінники. Ці об’єкти перехоплюють виклики до оригінального об’єкта, дозволяючи зробити щось *до* чи *після* передачі виклику оригіналові.

1. **ProxyRequests** інкапсулює використання бібліотеки **requests**. Він переопреділяє методи **get** та **post** так, щоб додати додаткову логіку.
2. **Зміна User-Agent**: Клас автоматично додає заголовок **User-Agent** до кожного запиту. Це робиться за допомогою бібліотеки **fake_useragent**, що дозволяє генерувати випадкові значення User-Agent. Це корисно для обходу деяких обмежень на веб-сайтах, які залежать від User-Agent.

```python
class Memento:
    def __init__(self, state):
        self.state = state

class ProxyRequests:
    def __init__(self):
        self._real_requests = requests
        self.user_agent = UserAgent()

    def get(self, *args, **kwargs):
        headers = kwargs.get('headers', {})
        headers['User-Agent'] = self.user_agent.random
        kwargs['headers'] = headers
        return self._real_requests.get(*args, **kwargs)

    def post(self, *args, **kwargs):
        headers = kwargs.get('headers', {})
        headers['User-Agent'] = self.user_agent.random
        kwargs['headers'] = headers
        return self._real_requests.post(*args, **kwargs)

    def __getstate__(self):
        state = self.__dict__.copy()
        state['_real_requests'] = None
        state['user_agent'] = None
        return state

    def __setstate__(self, state):
        self.__dict__.update(state)
        self._real_requests = requests
        self.user_agent = UserAgent()
```

Висновок:

У лабораторній роботі №4 було досліджено важливість та застосування шаблонів проектування на практичному прикладі. Через реалізацію класу **ProxyRequests**, який є чудовим прикладом використання шаблону Proxy, було продемонстровано ефективність шаблонів у структуруванні та оптимізації коду.