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 №5

# Технології розроблення програмного забезпечення

*Тема роботи: «*ШАБЛОНИ «ADAPTER», «BUILDER», «COMMAND», «CHAIN OF RESPONSIBILITY», «PROTOTYPE»*»*

Виконав студент групи ІА-23:

Каширов Д. О.

Перевірив:

Мягкий Михайло Юрійович

Київ 202

**Мета цієї роботи** — налаштувати та протестувати використання Jest для юніт-тестування в проекті, який включає адаптери для завантаження веб-сторінок через `axios` і проксі-сервер. Завдання полягає в перевірці коректності роботи адаптерів шляхом мокування HTTP-запитів та обробки можливих помилок, а також налаштуванні Jest для автоматичного виконання тестів.

## Хід роботи

## Варіант:

**..6** Web-browser (proxy, chain of responsibility, factory method, template method, visitor, p2p) Веб-браузер повинен мати можливість зробити наступне: мати адресний рядок для введення адреси сайту, переміщатися і відображати структуру html документа, переглядати підключений javascript та css файли, перегляд всіх підключених ресурсів (зображень), коректна обробка відповідей з сервера (коди відповідей HTTP) - переходи при перенаправленнях, відображення сторінок 404 і 502/503.

**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.

## Короткі теоретичні відомості:

**Юніт-тестування —** це метод тестування програмного забезпечення, при якому окремі компоненти або модулі програми перевіряються на коректність їх роботи в ізоляції від інших частин системи. Це дозволяє швидко виявити помилки на ранніх етапах розробки та забезпечити стабільність коду при подальших змінах.

**Jest —** це популярний фреймворк для тестування JavaScript-кодів, розроблений компанією Facebook. Він забезпечує просту настройку, зручні можливості для мокування та асинхронних тестів, а також вбудовану підтримку для тестування React-компонентів. Jest має функції для перевірки виконання тестів, створення звітів про покриття коду та асинхронних тестів.

**Мокування (mocking) —** це техніка в тестуванні, яка дозволяє замінювати реальні функції, методи чи об'єкти на їхні тестові версії для ізоляції тестованих частин коду від зовнішніх залежностей, таких як бази даних, зовнішні API або файли. Це дає можливість точніше контролювати умови тестування і забезпечити відтворювані результати.

У даній роботі використовується мокування для тестування адаптерів, що завантажують веб-сторінки через бібліотеки axios та проксі-сервери, щоб перевірити їх поведінку при успішних і неуспішних HTTP-запитах.

**Хід роботи:**

**Діаграма реалізації паттерну Adapter:**

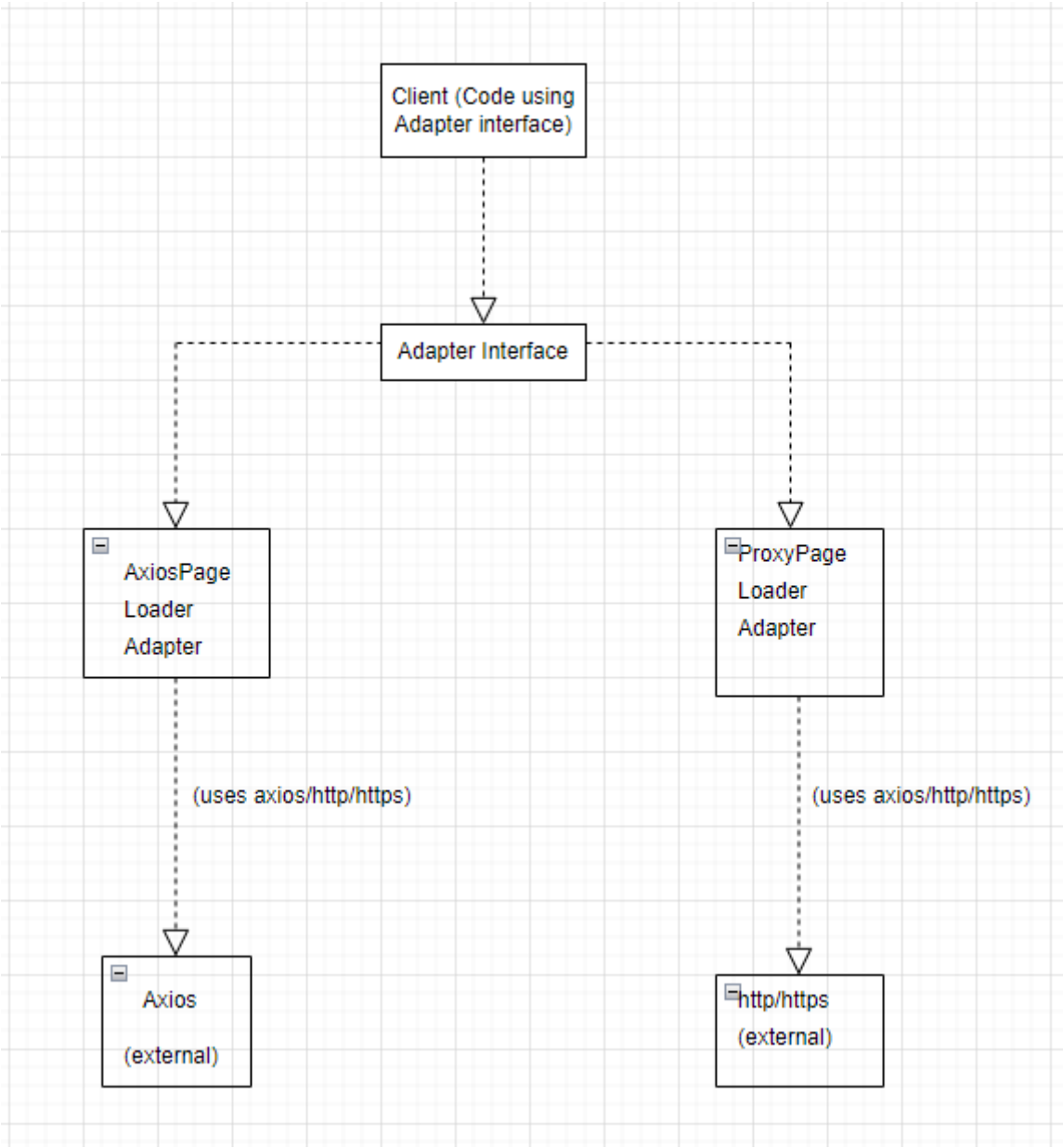


Рис 1- Реалізація паттерну Adapter

**Опис компонентів діаграми:**

**1. Client (Code using Adapter interface)**

- Клієнтський код, який використовує адаптер, — це частина програми, яка взаємодіє з адаптером, не знаючи про його внутрішні деталі. Клієнт лише користується інтерфейсом адаптера, який надає спільний метод для завантаження сторінки.
- Оскільки клієнт не має доступу до конкретних реалізацій адаптера (якщо це AxiosPageLoaderAdapter або ProxyPageLoaderAdapter), він

просто викликає метод інтерфейсу адаптера, наприклад, метод loadPage().

- Це дає змогу змінювати способи завантаження сторінок (через axios або проксі-сервери) без потреби змінювати клієнтський код.

**2. Adapter Interface**

- Інтерфейс адаптера — це абстракція, яка визначає спільні методи для всіх конкретних адаптерів, що реалізують цю інтерфейсну структуру.
- У нашому випадку це інтерфейс з методом, який дозволяє завантажувати сторінки, наприклад, loadPage(url).
- Через цей інтерфейс клієнтський код може викликати методи адаптера без турбот про те, як саме ці сторінки будуть завантажені. Це дозволяє гнучко змінювати реалізацію адаптера.

**3. AxiosPageLoaderAdapter**

- AxiosPageLoaderAdapter — це конкретний адаптер, який використовує бібліотеку axios для здійснення HTTP-запитів до серверів і завантаження сторінок.
- Реалізація цього адаптера виконує завантаження веб-сторінки, використовуючи методи axios.get(). Якщо запит успішний, адаптер повертає дані сторінки. Якщо ж виникає помилка, адаптер генерує помилку.
- За допомогою цього адаптера клієнт може використовувати можливості axios для завантаження сторінок без необхідності працювати з деталями HTTP-запитів.

**4. ProxyPageLoaderAdapter**

- ProxyPageLoaderAdapter — ще один конкретний адаптер, який дозволяє завантажувати сторінки через проксі-сервери. Цей адаптер може використовувати модулі http або https, залежно від типу URL, який передається.
- Використовуючи цей адаптер, клієнт може забезпечити завантаження сторінки, перебігаючи через проміжні проксі-сервери. Це корисно, коли потрібно здійснити додаткові операції з мережею (наприклад, фільтрація чи маршрутизація через проксі).

**5. Axios і http/https**

- Axios — це бібліотека для виконання HTTP-запитів, яку використовує AxiosPageLoaderAdapter для завантаження даних. Вона автоматизує багато рутинних задач при роботі з HTTP-запитами, таких як обробка відповідей, тайм-аути, обробка помилок тощо.
- http/https — це вбудовані в Node.js модулі для здійснення HTTP-запитів. Вони використовуються ProxyPageLoaderAdapter для завантаження сторінок через проксі-сервери. Модулі http та https дозволяють працювати з різними протоколами залежно від URL, який передається у метод.

**Принцип роботи патерну:**

1. Клієнтський код викликає метод інтерфейсу адаптера, не зважаючи на те, чи використовує адаптер axios або проксі.
2. Адаптер обирає, чи потрібно використовувати бібліотеку axios або модулі http/https залежно від своєї реалізації.
3. Кожен конкретний адаптер відповідно до своєї реалізації запитує дані або через axios, або через проксі-сервери.
4. Таким чином, патерн Адаптер дозволяє змінювати способи завантаження сторінок без зміни коду, який взаємодіє з адаптерами.

**Переваги:**

- Гнучкість: Клієнтський код не залежить від конкретної реалізації завантаження сторінки, тому можна легко змінювати реалізацію адаптера без змін в основному коді.
- Розширюваність: Якщо потрібно додати новий спосіб завантаження сторінок (наприклад, використовуючи інший HTTP-клієнт), можна просто створити новий адаптер, не змінюючи клієнтський код.
- Інкапсуляція: Дійсні деталі реалізації запитів (як завантажується сторінка, як обробляються помилки) приховані від клієнта, що спрощує підтримку і тестування коду.

Посилання на Git Hub на гілку main1
[https://github.com/DimytroKashiroff/TRPZ_Kashirov_All-laboratory-work.git]

**Висновок:**

Патерн Адаптер дозволяє ефективно обробляти різні способи завантаження веб-сторінок через єдиний інтерфейс, який спрощує підтримку, тестування та розширення програми.