****

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 №8  
**ШАБЛОНИ «COMPOSITE»,**

**«FLYWEIGHT, «INTERPRETER», «VISITOR»**

Варіант 14

Виконав Перевірив:

студент групи ІА – 13: Мягкий М. Ю

Михайленко Андрій

**Завдання:**

1. Ознайомитися з короткими теоретичними відомостями.

2. Реалізувати частину функціонала робочої програми у вигляді класів та їхньої взаємодії для досягнення конкретних функціональних можливостей.

3. Застосування одного з розглянутих шаблонів при реалізації програми

**Варіант:**

14. Архіватор (strategy, adapter, factory method, facade, visitor, p2p)Архіватор повинен являти собою візуальний додаток з можливістю створення і редагування архівів різного типу (.tar.gz, .zip, .rar, .ace) - додавання/ видалення файлів / папок, редагування метаданих (по можливості), перевірка checksum архівів, тестування архівів на наявність пошкоджень, розбиття архівів на частини.

**Хід роботи**

**Паттерн Visitor**

Шаблон відвідувач дозволяє вказувати операції над елементами без зміни структури конкретних елементів. Таким чином вкрай зручно додавати нові операції, проте дуже важко додавати нові елементи в ієрархію (необхідно додавати відповідні методи для обробки їх відвідувань в кожного відвідувача).

Ці зміни дозволяють виконувати додаткові операції з архівами під час їх стискання, не змінюючи основну логіку класів.

Це робить програму більш модульною та розширюваною, оскільки нові операції з архівами можуть легко додаватись без модифікації існуючого коду.

**Мета** використання Відвідувача в цьому випадку може бути обумовлено ключовими факторами:

1. **Розділення функціональності**: Паттерн Відвідувача дозволяє відокремити алгоритми обробки архівних записів від основної логіки стиснення.
2. **Збереження модульності**: Завдяки паттерну Відвідувача можна додавати нові методи обробки для різних типів архівів, не змінюючи існуючі класи стратегій стиснення. Це полегшує розширення програми та зберігає модульність коду.
3. **Підвищення читабельності та обслуговуваності коду**: Розділення алгоритмів обробки архівів робить код більш читабельним та зрозумілим. Це дозволяє розробникам швидше зорієнтуватись у функціональності кожного класу та робити зміни безпечніше.
4. **Виконання специфічних операцій для кожного типу архіву**: Завдяки Відвідувачу можна легко визначати та виконувати конкретні операції для кожного типу архіву. Наприклад, різні типи архівів можуть мати власні методи для обробки їхніх записів.

**Висновок**: ознайомився з короткими теоретичними відомостями. Реалізував частину функціонала робочої програми у вигляді класів та їхньої взаємодії для досягнення конкретних функціональних можливостей. Застосував паттерн Відвідувач для реалізації програми.