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

Тема: «Патерни проектування.»

Виконала:
Студентка групи ІА-34
Сизоненко А.О
Перевірив:
Мягкий Михайло Юрійович

Київ-2025

Мета: Вивчити структуру шаблонів «Mediator», «Facade», «Bridge», «Template method» та навчитися застосовувати їх в реалізації програмної системи.

Тема роботи: Текстовий редактор (strategy, command, observer, template method, flyweight, SOA)

Текстовий редактор повинен вміти розпізнавати текстові файли в будь-якій кодуванні, мати розширені функції редагування: макроси, сніппети, підказки, закладки, перехід на рядок / сторінку, підсвічування синтаксису (для однієї мови програмування або розмітки на розсуд студента).

Теоретичні відомості

Mediator (Посередник)

- Організовує взаємодію між об’єктами через центральний об’єкт, зменшуючи прямі залежності.
- **Переваги:** знижує зв’язність, спрощує управління взаємодіями, полегшує тестування.
- **Недоліки:** посередник може стати занадто складним («God Object»).

Facade (Фасад)

- Створює спрощений інтерфейс для підсистеми, приховуючи внутрішні деталі.
- **Переваги:** інкапсуляція внутрішньої структури, спрощений доступ до підсистеми.
- **Недоліки:** знижує гнучкість налаштування підсистеми.

Bridge (Міст)

- Розділяє абстракцію та реалізацію, щоб їх можна було змінювати незалежно.
- **Переваги:** гнучкість, простіший супровід, запобігає комбінаторному росту кількості класів.
- **Недоліки:** ускладнює структуру через додаткові проміжні рівні.

Template Method (Шаблонний метод)

- Визначає алгоритм у базовому класі, залишаючи частини реалізації підкласам.
- **Переваги:** мінімізує дублювання коду, легко додавати нові варіанти алгоритму, контролює порядок кроків.
- **Недоліки:** жорстка структура алгоритму, залежність підкласів від базового класу.

Хід роботи

1. Діаграма класів для реалізації шаблону шаблонний метод “templateMethod”.

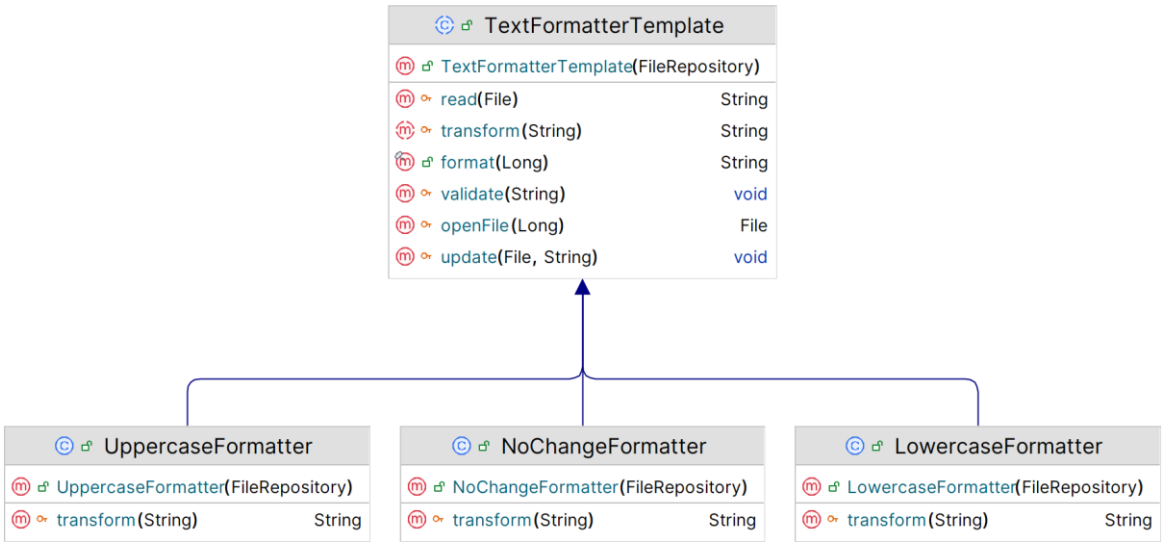


Рисунок 1 – Діаграма класів

Опис діаграми класів

1. TextFormatterTemplate – визначає скелет алгоритму обробки тексту, не залежно від формату. Але сам формат можна вибрати трасформації тексту можна вибрати. Метод transform реалізує конкретну трансформацію (uppercase, lowercase, no change)
2. LowercaseFormatter – конкретна реалізація перетворює текст у нижній регістр.
3. UppercaseFormatter – конкретна реалізація перетворює текст у верхній регістр.
4. NoChangeFormatter – конкретна реалізація залишає текст у початковому стані.

Приклад реалізації шаблону:

1. App створює файли і зберігає їч.
2. Потім створюється конкретний Formatter (UppercaseFormatter тощо).
3. Викликається formatter.format(fileId):
    - відкриває файл,
    - читає текст,
    - виконує трансформацію через transform(),
    - зберігає оновлений текст назад.

TextFormatterTemplate визначає алгоритм обробки тексту (format). Підкласи визначають тільки конкретну трансформацію. Зміни алгоритму (наприклад, додати перевірку або новий крок) робляться в базовому класі, а підкласи залишаються простими.

Демонстрація роботи шаблону Observer

```
public class App {
    public static void main(String[] args) {
        fileJson.setId(1L);
        fileJson.setFileName("file_json1");
        fileJson.setFilePath("C:\\Users\\АНЮТА\\IdeaProjects\\trpz\\json");
        fileJson.setContent("\"text\": \"Hello JSON\"");

        File filetxt = new File();
        filetxt.setId(2L);
        filetxt.setFileName("file_txt1");
        filetxt.setFilePath("C:\\Users\\АНЮТА\\IdeaProjects\\trpz\\txt");
        filetxt.setContent("\"TEXT\": \"Hello TXT\"");

        Command save1 = new SaveFileCommand(repository, fileJson);
        Command save2 = new SaveFileCommand(repository, filetxt);
        invoker.executeCommand(save1);
        invoker.executeCommand(save2);

        TextFormatterTemplate formatter = new UppercaseFormatter(repository);
        TextFormatterTemplate formatter2 = new LowercaseFormatter(repository);
        String result = formatter.format( fileId: 1L);
        String result2 = formatter2.format( fileId: 2L);
        System.out.println(result);
        System.out.println(result2);
    }
}
```

Рисунок 2 – код класу App

Результат виконання програми:

```
Файл збережено як JSON: C:\Users\АНЮТА\IdeaProjects\trpz\json\file_json1.json
[LOG] File 'file_json1' event: saved
[StatusBar] Action: saved | Symbols in content: 20
Збережено як TXT: C:\Users\АНЮТА\IdeaProjects\trpz\txt\file_txt1.txt
[LOG] File 'file_txt1' event: saved
[StatusBar] Action: saved | Symbols in content: 19
"TEXT": "HELLO JSON"
"text": "hello txt"
```

Рисунок 3 – вивід повідомлення в консолі

Переваги використання шаблону TemplateMethod для моєї теми проекту:

1. Всі операції над текстом (читання, перевірка, обробка, збереження) виконуються за єдиним шаблоном у базовому класі TextFormatterTemplate.
2. Підкласам не потрібно дублювати спільні кроки — вони реалізують лише конкретну трансформацію (uppercase, lowercase, no change). Це зменшує кількість повторюваного коду.
3. Якщо з’являється новий формат або алгоритм обробки, достатньо створити новий підклас і реалізувати метод transform().

Недоліки використання шаблону TemplateMethod у текстовому редакторі:

1. Жорстка структура алгоритму

Базовий клас визначає послідовність кроків, і підкласи не можуть змінювати порядок без зміни базового класу.

2. Можливе надмірне наслідування

   Для кожного нового формату тексту треба створювати окремий підклас → може зрости кількість класів.

3. Обмежена гнучкість підкласів

   Підкласи можуть змінювати тільки ті кроки, які явно передбачені як абстрактні.

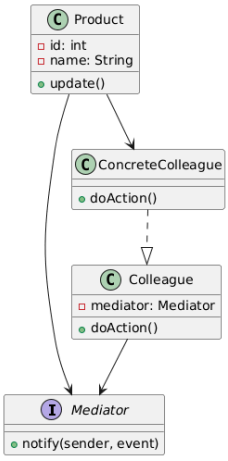
Висновок: У роботі реалізовано патерн Template Method для уніфікованої обробки тексту у редакторі. Абстрактний клас TextFormatterTemplate визначає загальний алгоритм: відкриття файлу, читання контенту, перевірка валідності, трансформація та збереження зміненого тексту. Конкретні підкласи, такі як UppercaseFormatter, LowercaseFormatter та NoChangeFormatter, реалізують лише метод transform(), який визначає конкретний спосіб обробки тексту. Це дозволяє легко додавати нові формати або типи обробки без зміни базового алгоритму та мінімізує дублювання коду. Було створено діаграму класів, яка наочно демонструє взаємодію компонентів системи: базовий клас визначає послідовність дій, підкласи реалізують конкретні кроки, а репозиторій файлів (FileRepository) забезпечує доступ до контенту файлів для обробки.

Відповіді на контрольні питання

1. Яке призначення шаблону «Посередник»?

Організовує взаємодію між об’єктами через центральний об’єкт, зменшуючи прямі залежності між ними.

2. Нарисуйте структуру шаблону «Посередник».



3. Які класи входять в шаблон «Посередник», та яка між ними взаємодія?

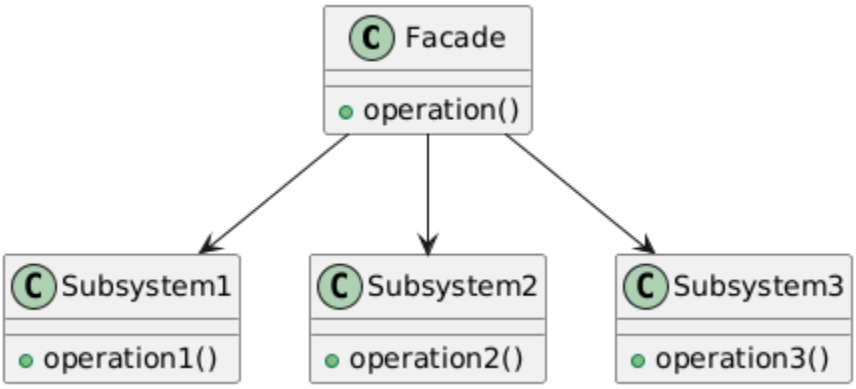
- Mediator (інтерфейс) — визначає методи взаємодії.
- ConcreteColleage — конкретна реалізація колеги.

- Colleague — об’єкти, які взаємодіють через медіатора.
- Взаємодія: колеги не знають один про одного, лише про медіатора; медіатор координує їх дії.

4. Яке призначення шаблону «Фасад»?

Створює спрощений інтерфейс для підсистеми, приховуючи внутрішні деталі і забезпечуючи єдиний доступ.

5. Нарисуйте структуру шаблону «Фасад».



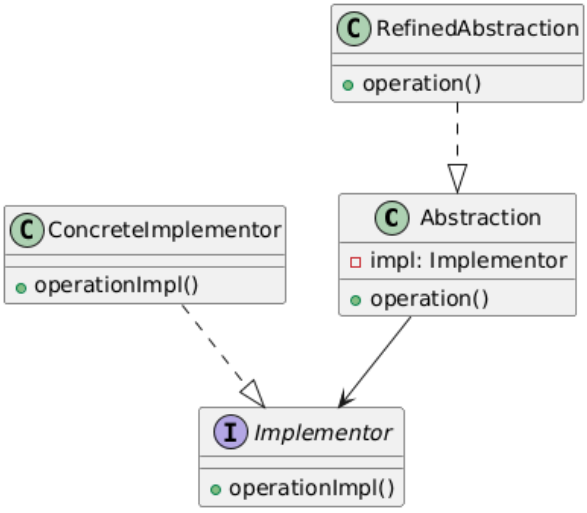
6. Які класи входять в шаблон «Фасад», та яка між ними взаємодія?

- **Facade** — надає спрощений інтерфейс для клієнтів.
- **Subsystem** — внутрішні класи підсистеми.
- **Взаємодія:** клієнт звертається тільки до фасаду, фасад керує підсистемами.

7. Яке призначення шаблону «Міст»?

Розділяє абстракцію і реалізацію, щоб їх можна було змінювати незалежно одна від одної.

8. Нарисуйте структуру шаблону «Міст».



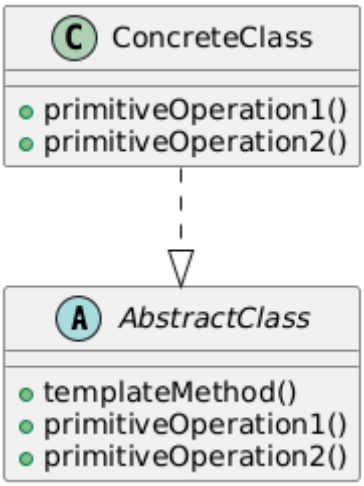
9. Які класи входять в шаблон «Міст», та яка між ними взаємодія?

- **Abstraction** — абстрактний інтерфейс.
- **RefinedAbstraction** — розширення абстракції.
- **Implementor** — інтерфейс реалізації.
- **ConcreteImplementor** — конкретна реалізація.
- **Взаємодія:** абстракція делегує роботу реалізації через інтерфейс Implementor; обидві ієрархії незалежні.

10.Яке призначення шаблону «Шаблоний метод»?

Визначає загальний алгоритм у базовому класі, залишаючи частини реалізації підкласам.

11.Нарисуйте структуру шаблону «Шаблоний метод».



12.Які класи входять в шаблон «Шаблоний метод», та яка між ними

взаємодія?

- **AbstractClass** — містить Template Method та абстрактні методи для підкласів.
- **ConcreteClass** — реалізує специфічну логіку окремих кроків алгоритму.
- **Взаємодія:** Template Method викликає абстрактні методи, реалізовані підкласами; підкласи змінюють тільки специфічні кроки.

13.Чим відрізняється шаблон «Шаблонний метод» від «Фабричного

методу»?

Template Method: визначає алгоритм і делегує частини реалізації підкласам.

Factory Method: визначає інтерфейс для створення об’єкта, а конкретні підкласи вирішують, який саме об’єкт створювати.

Головна різниця: Template Method — про порядок дій алгоритму, Factory Method — про створення об’єктів.

14.Яку функціональність додає шаблон «Міст»?

- Забезпечує незалежне змінення абстракцій та реалізацій.
- Спрощує розширення системи без множення підкласів.
- Дає гнучкість при додаванні нових реалізацій або абстракцій