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
Технологія розроблення програмного забезпечення

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

Варіант 25

Виконав
студент групи ІА-23
Калина С. О.

Перевірив:
Мягкий Михайло Юрійович

Київ 2024

Тема.
Шаблони «ADAPTER», «BUILDER», «COMMAND», «CHAIN OF RESPONSIBILITY», «PROTOTYPE»

Мета.
Ознайомитися з принципами роботи шаблонів проектування «ADAPTER», «BUILDER», «COMMAND», «CHAIN OF RESPONSIBILITY», «PROTOTYPE», їх перевагами та недоліками. Набути практичних навичок у застосуванні шаблону Command при розробці програмного забезпечення на прикладі реалізації менеджера завантажень.

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

Хід роботи

25. Installer generator (iterator, builder, factory method, bridge, interpreter, client-server)
Генератор інсталяційних пакетів повинен мати якийсь спосіб налаштування файлів, що входять в установку, установки вікон з інтерактивними можливостями (галочка - створити ярлик на робочому столі; ввести в текстове поле деякі дані, наприклад, ліцензійний ключ і т.д.). Генератор повинен вивести один файл .exe або .msi.

Короткі теоретичні відомості.

Анти-шаблони (або анти-патерни) — це повторювані помилки у проектуванні та розробці програмного забезпечення. Вони виникають через недостатнє планування, неправильне розуміння концепцій або намагання швидко знайти рішення. Вивчення анти-шаблонів допомагає їх уникати в майбутньому.

Основні категорії анти-шаблонів:

1. Анти-шаблони управління розробкою

- Дим і дзеркала: Обіцянка ненаписаних функцій через демонстрацію вигляду системи.
- Роздування ПЗ: Постійне збільшення вимог до ресурсів нових версій.
- Функції для галочки: Непов’язані функції додаються лише для рекламних заяв.

2. Анти-шаблони дизайну ПЗ

- Великий клубок бруду: Невпорядкована структура системи.
- Божественний об'єкт: Один клас виконує забагато функцій.
- Система "димохід": Погано пов’язані компоненти, які важко підтримувати.
- Гонки умов: Непередбачуваний порядок виконання подій.

3. Анти-шаблони об'єктно-орієнтованого програмування (ООП)

- Блоб: Один об'єкт виконує всі завдання.
- Полтергейст: Об’єкти існують лише для передачі інформації іншим.
- Проблема йо-йо: Залежність коду від глибокої ієрархії класів.
- Недоречний "Сінглтон": Неправильне використання патерну "Singleton".

4. Анти-шаблони програмування

- Спагеті-код: Заплутаний код із нелогічним порядком виконання.
- Жорстке кодування: Впровадження залежностей від середовища без гнучкості.
- Мильна бульбашка: Клас, який лише імітує збереження даних.
- Перевірка типу замість інтерфейсу: Визначення типу об'єкта там, де потрібен лише інтерфейс.

Головні принципи для уникнення анти-шаблонів:

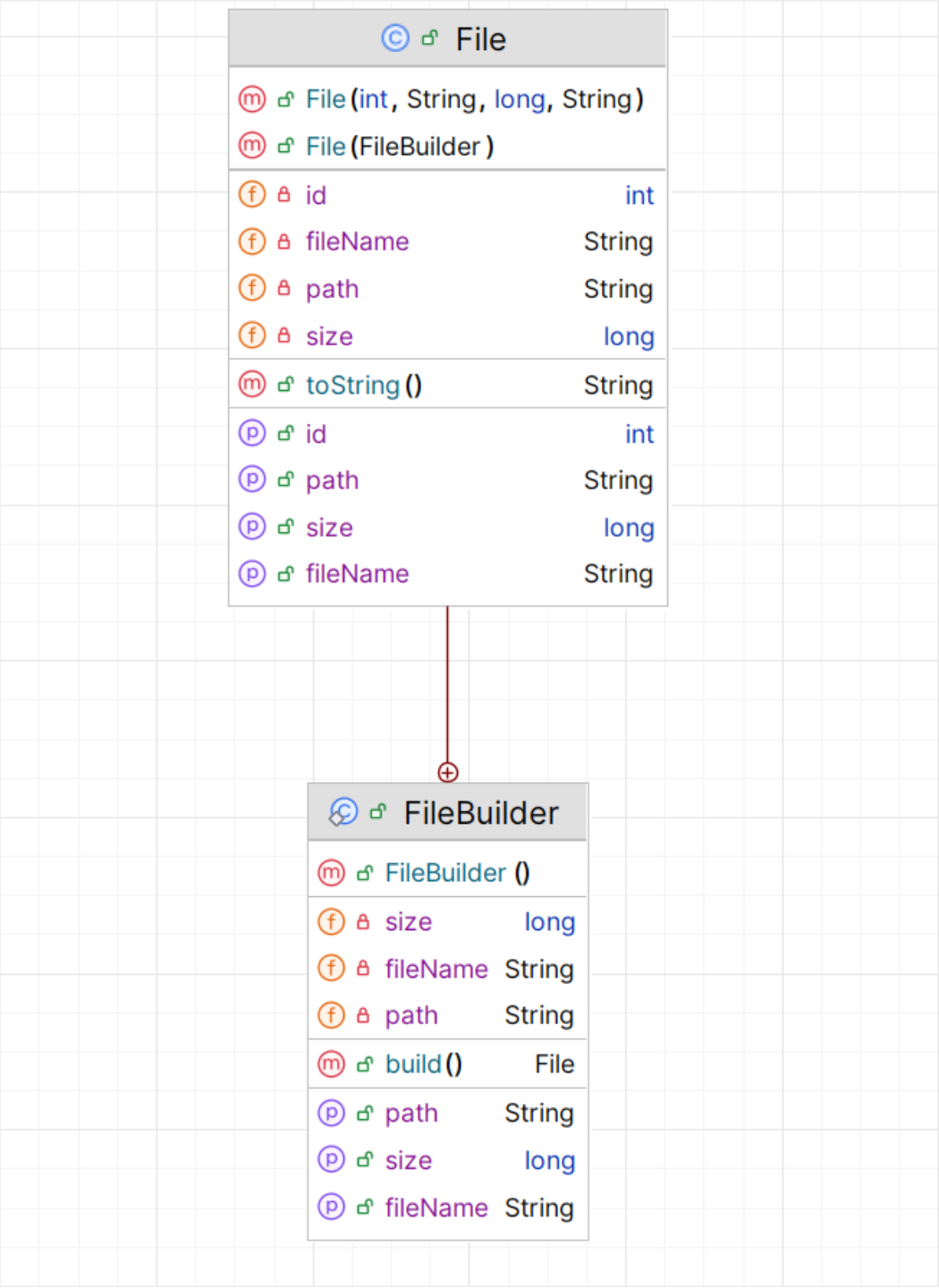
Чітке планування архітектури.

Розбиття функціоналу на модулі.

Дотримання принципів чистого коду.

Використання стандартних шаблонів проектування.

Анти-шаблони демонструють, що неправильні рішення можуть суттєво ускладнити розробку та підтримку програмного забезпечення, тому їх потрібно вчасно розпізнавати й уникати.



На зображенні показана UML-діаграма класів, яка представляє структуру класу File та допоміжного класу FileBuilder.

Клас File

Це основний клас, який представляє файл. Він містить такі елементи:

1. **Конструктори:**
   - File (int, String, long, String) — дозволяє створити об'єкт File з параметрами: ідентифікатор (id), назва файлу (fileName), розмір (size) та шлях (path).
   - File (FileBuilder) — створює об'єкт File на основі екземпляра FileBuilder.
2. **Атрибути:**
   - id (типу int) — унікальний ідентифікатор файлу.
   - fileName (типу String) — назва файлу.
   - path (типу String) — шлях до файлу.
   - size (типу long) — розмір файлу.
3. **Методи:**
   - toString () — повертає текстове представлення об'єкта File.
4. **Властивості (properties):**
   - Геттер і сетер для кожного атрибута (id, path, size, fileName).

Клас FileBuilder

Цей клас реалізує патерн "Builder" для створення об'єктів File.

1. **Конструктор:**
   - FileBuilder () — базовий конструктор без параметрів.
2. **Атрибути:**
   - size (типу long) — розмір файлу.
   - fileName (типу String) — назва файлу.
   - path (типу String) — шлях до файлу.
3. **Методи:**
   - build () — створює і повертає об'єкт File на основі даних, встановлених у FileBuilder.
4. **Властивості (properties):**
   - Геттер і сетер для кожного атрибута (path, size, fileName).

Зв'язок між класами

- **Асоціація:** File пов'язаний із FileBuilder. Конструктор File (FileBuilder) дозволяє створювати екземпляр File на основі даних, визначених у FileBuilder.

```
public static void main(String[] args) {  Kalyna-Serhii *
    File file1 = new File.FileBuilder()
            .setFileName("document1.txt")
            .setSize(1024L)
            .setPath("/user/documents/document1.txt")
            .build();

    File file2 = new File.FileBuilder()
            .setFileName("image1.jpg")
            .setSize(2048L)
            .setPath("/user/images/image1.jpg")
            .build();

    File file3 = new File.FileBuilder()
            .setFileName("presentation.pptx")
            .setSize(4096L)
            .setPath("/user/presentations/presentation.pptx")
            .build();

    File file4 = new File.FileBuilder()
            .setFileName("video.mp4")
            .setSize(8192L)
            .setPath("/user/videos/video.mp4")
            .build();

    File file5 = new File.FileBuilder()
            .setFileName("music.mp3")
            .setSize(5120L)
            .setPath("/user/music/music.mp3")
            .build();

    System.out.println(file1);
    System.out.println(file2);
    System.out.println(file3);
    System.out.println(file4);
    System.out.println(file5);
}
```

```
FileName id: 0 size: 1024 fileName: document1.txt path: /user/documents/document1.txt
FileName id: 0 size: 2048 fileName: image1.jpg path: /user/images/image1.jpg
FileName id: 0 size: 4096 fileName: presentation.pptx path: /user/presentations/presentation.pptx
FileName id: 0 size: 8192 fileName: video.mp4 path: /user/videos/video.mp4
FileName id: 0 size: 5120 fileName: music.mp3 path: /user/music/music.mp3
```

Висновок: в ході виконання лабораторної роботи було вивчено теоретичні основи шаблонів проектування, зокрема детально розглянуто шаблони Adapter, Builder, Command, Chain of Responsibility та Prototype. Реалізовано шаблон Command для системи управління завантаженнями файлів. Отримані знання та практичні навички можуть бути використані при розробці інших програмних систем, де необхідна гнучка обробка команд та можливість скасування операцій.