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 №6
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 «Abstract Factory», «Factory Method», «Memento», «Observer», «Decorator»

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.

Принципи SOLID

1. **S (Single Responsibility Principle – Принцип єдиного обов'язку)**
    - **Суть**: Клас повинен виконувати лише одну задачу і мати одну причину для зміни.
    - **Переваги**:
        - Зменшення складності коду.
        - Спрощення тестування, налагодження та підтримки.
    - **Приклад порушення**: Клас, що одночасно відповідає за логіку збереження даних і відображення інтерфейсу.
    - **Рішення**: Розділити відповідальності між двома окремими класами.
2. **O (Open/Closed Principle – Принцип відкритості/закритості)**
    - **Суть**: Код повинен бути відкритий для розширення, але закритий для модифікації.
    - **Переваги**:
        - Запобігання помилкам у вже протестованому коді.
        - Легкість додавання нової функціональності.
    - **Приклад реалізації**: Використання інтерфейсів і абстрактних класів для визначення поведінки, яку можна розширити.
3. **L (Liskov Substitution Principle – Принцип підстановки Лісков)**
    - **Суть**: Кожен об'єкт базового класу повинен бути замінний об'єктом його підкласу без порушення працездатності системи.
    - **Порушення**: Якщо підклас змінює поведінку методу базового класу.
    - **Приклад**: Метод у підкласі не повинен кидати виняток, якщо базовий метод цього не робить.
4. **I (Interface Segregation Principle – Принцип розділення інтерфейсу)**

- **Суть**: Краще створювати декілька вузькоспеціалізованих інтерфейсів, ніж один великий.
- **Переваги**:
  - Клієнти працюють тільки з тими методами, які їм дійсно потрібні.
- **Приклад**: Поділ інтерфейсу "Printer" на "Scanner", "Copier", "Printer" для багатофункціональних пристроїв.

5. **D (Dependency Inversion Principle – Принцип інверсії залежностей)**
   - **Суть**: Модулі верхнього рівня не повинні залежати від модулів нижнього рівня. Обидва повинні залежати від абстракцій.
   - **Переваги**:
     - Зменшення зв’язності системи.
     - Збільшення можливостей для тестування.
   - **Реалізація**: Інверсія залежностей через інтерфейси або фабричні методи.

Патерни проектування

1. Abstract Factory (Абстрактна фабрика)

- **Суть**: Дозволяє створювати сімейства пов’язаних об’єктів без вказівки їхніх конкретних класів.
- **Переваги**:
  - Забезпечує узгодженість створюваних об'єктів.
  - Полегшує заміну наборів об'єктів.
- **Приклад використання**: Різні графічні бібліотеки для Windows і macOS, де фабрика створює взаємопов’язані елементи інтерфейсу.

2. Factory Method (Фабричний метод)

- **Суть**: Визначає інтерфейс для створення об'єктів, але дозволяє підкласам вибирати конкретний тип об'єкта.
- **Переваги**:
  - Виносить логіку створення об’єктів за межі клієнтського коду.

    - Полегшує підтримку та розширення коду.
- **Приклад**: Створення об'єктів у різних форматах, наприклад, текстових чи XML-файлів.

3. Memento (Знімок)
- **Суть**: Зберігає внутрішній стан об'єкта, щоб можна було повернути його до цього стану.
- **Переваги**:
    - Забезпечує інкапсуляцію стану.
    - Зручність реалізації функції "Скасувати" (Undo).
- **Приклад**: Текстовий редактор, де зберігаються проміжні стани документа.

4. Observer (Спостерігач)
- **Суть**: Встановлює залежність "один-до-багатьох" між об'єктами, де один об'єкт (спостерігач) автоматично оновлюється при зміні іншого.
- **Переваги**:
    - Зручність для роботи з подіями.
    - Зменшення зв’язності компонентів.
- **Приклад**: Система новин, де всі підписники отримують повідомлення про нові статті.

5. Decorator (Декоратор)
- **Суть**: Динамічно додає нову поведінку до об'єкта, залишаючи незмінним його інтерфейс.
- **Переваги**:
    - Гнучка альтернатива наслідуванню.
    - Можливість багаторазового комбінування поведінок.
- **Приклад**: Декоратори для графічних елементів, таких як рамки, тіні тощо.

Хід роботи.

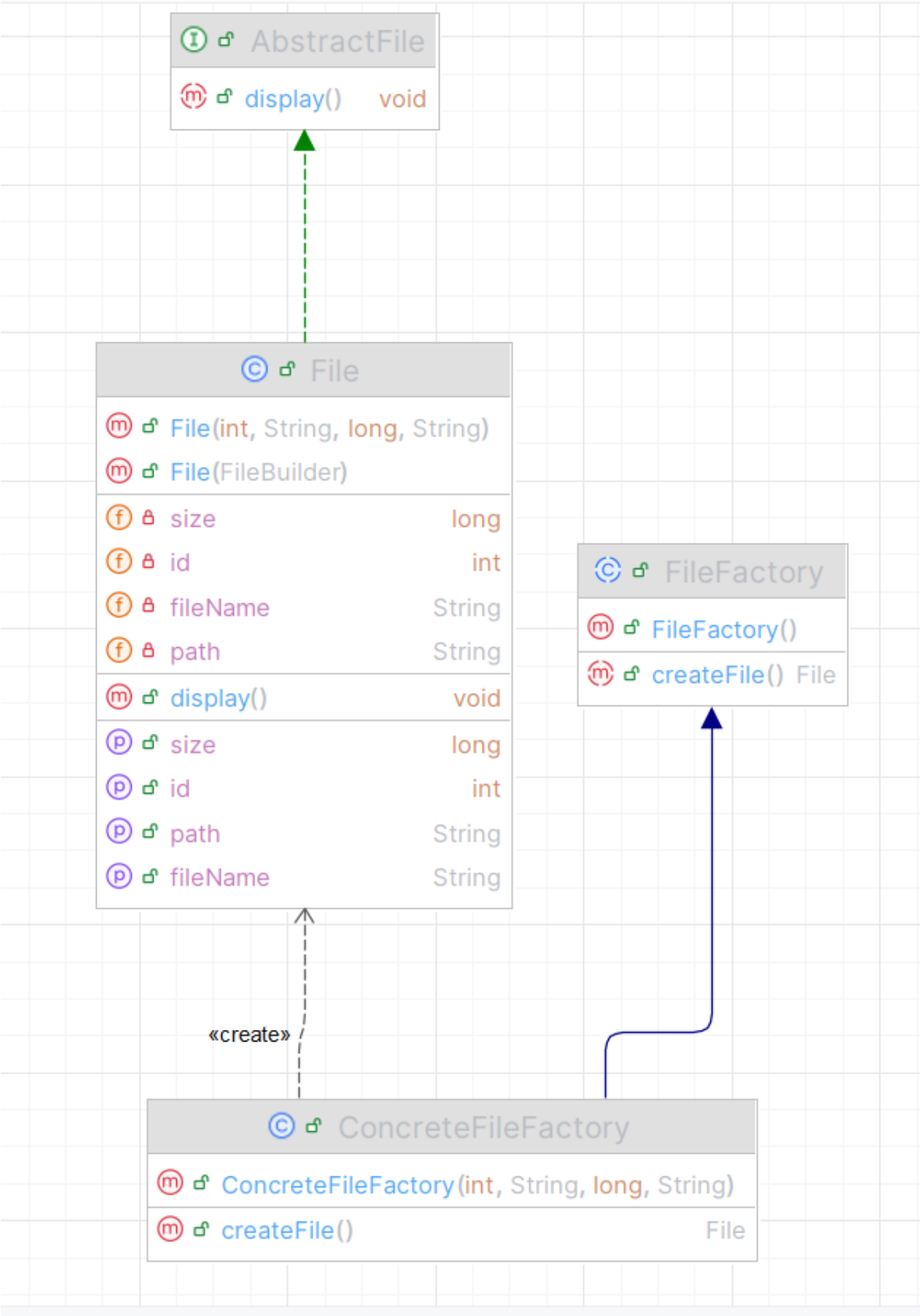

Рисунок №1 – Діаграма класів , згенерована IDE, реалізації шаблону Factory

Діаграма містить чотири основні компоненти:

1. **AbstractFile**
   - Це абстрактний клас або інтерфейс, який містить метод display(). Цей метод є абстрактним і має бути реалізований у підкласах.
   - display() — метод без реалізації (void), який служить для відображення інформації про файл.
2. **File**
   - Це клас, який успадковує (extends) AbstractFile.
   - Містить поля:
     - id (int) — ідентифікатор файлу.
     - fileName (String) — назва файлу.
     - size (long) — розмір файлу.
     - path (String) — шлях до файлу.
   - Методи:
     - Конструктори:
       - File(int, String, long, String) — приймає параметри для ініціалізації об'єкта File.
       - File(FileBuilder) — конструктор, який приймає об'єкт FileBuilder для побудови екземпляра.
     - display() — реалізація абстрактного методу display() з класу AbstractFile.
   - Геттери (getId(), getFileName(), getSize(), getPath()) для отримання значень полів класу.
3. **FileFactory**
   - Це інтерфейс, який містить метод createFile().
   - createFile() — абстрактний метод, який повертає об'єкт типу File. Його реалізація буде визначена в класах, що реалізують цей інтерфейс.
4. **ConcreteFileFactory**
   - Це конкретний клас (реалізація шаблону "Factory Method"), який реалізує інтерфейс FileFactory.
   - Конструктор ConcreteFileFactory(int, String, long, String) — приймає параметри, які використовуються для створення об'єкта File.
   - Метод createFile() — повертає новий об'єкт File з параметрами, які були передані під час створення фабрики.

Стосунки між класами:

- **Спадкування**:
  - Клас File успадковує AbstractFile, що вказано пунктирною зеленою стрілкою. Це означає, що File реалізує метод display(), оголошений в AbstractFile.
- **Інтерфейсна реалізація**:
  - Клас ConcreteFileFactory реалізує інтерфейс FileFactory. Це показано суцільною стрілкою з пустим трикутником.
- **Залежності**:
  - Клас ConcreteFileFactory створює екземпляр File через метод createFile(). Цей зв'язок позначено підписом «create», що демонструє використання фабрики для створення об'єкта.

Робота паттерну

```
Інформація про файл:
ID: 1
Назва файлу: document1.txt
Розмір: 1024 байт
Шлях: /user/documents/document1.txt

Результат методу display():
FileName id: 1 size: 1024 fileName: document1.txt path: /user/documents/document1.txt
```

Висновок:

В ході виконання лабораторної роботи було досліджено основні принципи SOLID та їх важливість у розробці програмного забезпечення. Вивчено теоретичні основи шаблонів проектування "Abstract Factory", "Factory Method", "Memento", "Observer" та "Decorator", їх призначення та випадки застосування.