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: «Патерни проектування»

Виконав:

Студент групи ІА-31

Калиновський В.О

Перевірив:

Мягкий Михайло Юрійович

Київ-2025

Мета: Вивчити структуру шаблонів «Adapter», «Builder», «Command», «Chain of responsibility», «Prototype» та навчитися застосовувати їх в реалізації програмної системи.

Тема роботи:

Графічний редактор (proxy, prototype, decorator, bridge, flyweight, SOA)

Графічний редактор повинен вміти створювати / редагувати растрові (або векторні на розсуд студента) зображення в 2-3 основних популярних форматах (bmp, png, jpg), мати панель інструментів для створення графічних примітивів, вибору кольорів, нанесення тексту, додавання найпростіших візуальних ефектів (ч/б растр, інфрачервоний растр, 2-3 на вибір учня), роботи з шарами.

Теоретичні відомості

Анти-шаблони (анти-патерни) – це поширені рішення, які здаються правильними на перший погляд, але насправді вони можуть призвести до проблем у проектуванні і розвитку програмного забезпечення. Вони часто виникають через недостатнє розуміння проблеми або через використання швидких рішень, що не враховують довгострокові наслідки. Замість того, щоб бути кращими практиками, анти-шаблони можуть стати джерелом технічного боргу, що ускладнює підтримку і розширення систем

Шаблон «Adapter» використовується для приведення інтерфейсу одного об’єкта до інтерфейсу іншого, з яким працює програма. Це корисно, коли потрібно інтегрувати компоненти або бібліотеки з різними інтерфейсами, але однаковим функціоналом, не змінюючи їх внутрішній код. Адаптери дозволяють створити єдиний уніфікований інтерфейс для роботи з різними реалізаціями.

Шаблон «Builder» використовується для поетапного створення складних об’єктів, відокремлюючи процес створення від представлення. Це дозволяє легко створювати різні варіанти одного об’єкта або об’єкти, які потребують багато налаштувань, без нагромадження конструкторів із великою кількістю параметрів.

Шаблон «Chain of Responsibility» організовує обробку запиту у вигляді ланцюга об’єктів, де кожен об’єкт має можливість обробити запит або передати його далі. Це дозволяє розділити обов’язки між кількома об’єктами і спростити логіку прийняття рішень, наприклад, при підписанні документів або обробці подій.

Шаблон «Prototype» дозволяє створювати нові об’єкти шляхом клонування вже існуючих прототипів. Це зручно, коли відомо, як має виглядати кінцевий об’єкт, або коли створення об’єкта з нуля є складним або ресурсозатратним. Використання прототипів зменшує ієрархію класів і спрощує створення складних об’єктів під час виконання програми.

Шаблон команда (Command) є структурним шаблоном проектування, який дозволяє змінювати інтерфейс одного класу так, щоб він став сумісним з інтерфейсом іншого класу. Це корисно, коли вам потрібно інтегрувати компоненти або системи, які мають різні інтерфейси, але ви не хочете змінювати їх код.

Хід роботи

1. Діаграма класів для реалізації шаблону команда.

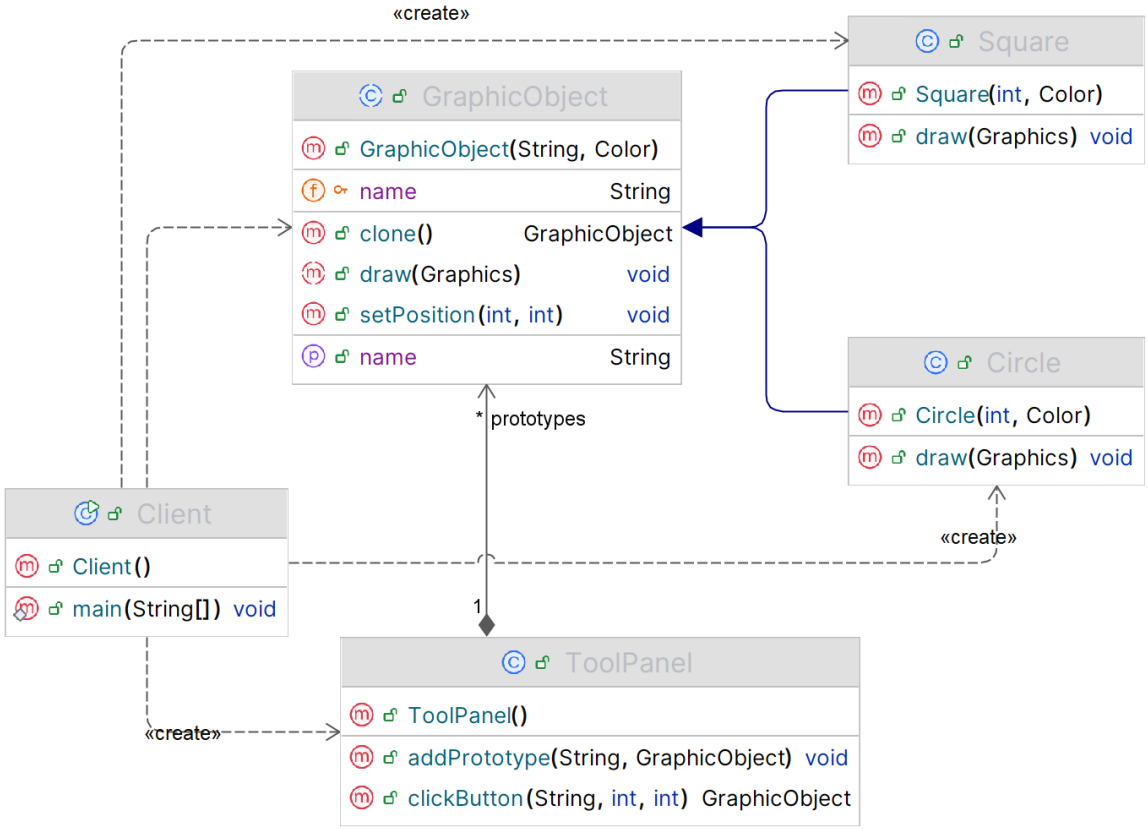


Рисунок 1 – Діаграма класів

Опис діаграми класів

GraphicObject
Роль: Абстрактний прототип.
Відповідає за визначення спільних властивостей графічних об’єктів (позиція, колір, ім’я) та реалізацію методу clone() для створення копій об’єкта.

Circle
Роль: Конкретний прототип.
Відповідає за реалізацію конкретного графічного об’єкта — кола, а також за його малювання.

Square
Роль: Конкретний прототип.
Відповідає за реалізацію іншого конкретного графічного об’єкта — квадрата, і за його малювання.

ToolPanel
Роль: Клієнт у шаблоні Prototype.
Відповідає за зберігання прототипів для панелі інструментів та створення клонів об’єктів при натисканні кнопок.

Client
Роль: Демонстраційний клієнт.
Відповідає за використання панелі інструментів для додавання клонованих об’єктів на полотно графічного редактора.

Приклад використання Prototype у графічному редакторі
Спочатку Client створює об’єкти, які беруть участь у патерні:

- Створюється ToolPanel, який зберігає прототипи графічних об’єктів.
- Створюються конкретні прототипи об’єктів (Circle, Square), які додаються у ToolPanel.

Далі Client натискає кнопки на панелі інструментів:

- Коли користувач натискає кнопку «Circle», ToolPanel повертає клон прототипу Circle.
- Коли користувач натискає кнопку «Square», ToolPanel повертає клон прототипу Square.

Потім Client використовує клоновані об’єкти:

1. ToolPanel викликає clone() на відповідному прототипі.
2. Клонований об’єкт отримує координати або інші налаштування, задані користувачем.
3. Client може викликати draw() на об’єкті, щоб намалювати його на екрані або обробити іншим чином.

Якщо потрібно додати ще один об’єкт того ж типу:

- Client просто знову натискає кнопку.
- ToolPanel повертає ще один клон прототипу, без створення об’єкта з нуля.

Демонстрація роботи шаблону Prototype

```java
public class Client {public static void main(String[] args) {
    ToolPanel panel = new ToolPanel();

    // додаємо прототипи для панелі
    panel.addPrototype( name: "Circle", new Circle( radius: 40, Color.RED));
    panel.addPrototype( name: "Square", new Square( size: 50, Color.BLUE));

    // список об'єктів на полотні редактора
    List<GraphicObject> canvas = new ArrayList<>();

    // імітуємо натискання кнопок
    canvas.add(panel.clickButton( name: "Circle",  x: 10,  y: 10));
    canvas.add(panel.clickButton( name: "Square",  x: 100,  y: 50));
    canvas.add(panel.clickButton( name: "Circle",  x: 200,  y: 150));

    // малюємо об'єкти (можна в GUI, зараз просто в консолі)
    for (GraphicObject obj : canvas) {
        obj.draw( g: null);
    }
}
```

Рисунок 2 – код класу Client

Результат виконання програми:

```
Draw Circle at (10,10), radius=40
Draw Square at (100,50), size=50
Draw Circle at (200,150), radius=40
```

Рисунок 3 – вивід повідомлення в консолі

Переваги використання шаблону Prototype у графічному редакторі:

1. **Швидке створення об'єктів** — нові графічні об'єкти можна отримати шляхом клонування існуючого прототипу, без повторного налаштування всіх параметрів.
2. **Зменшення дублювання коду** — не потрібно створювати нові екземпляри кожного типу об'єкта вручну.
3. **Гнучкість і масштабованість** — легко додавати нові типи фігур чи елементів, просто створивши прототип і додавши його на панель інструментів.
4. **Повна копія об'єкта** — клон включає всі властивості, включно з приватними полями, що полегшує відтворення складних об'єктів.
5. **Спрощене налаштування** — можна створити базовий шаблон об'єкта з потрібними налаштуваннями (колір, розмір) і клонувати його для користувача.

Недоліки використання шаблону Prototype у графічному редакторі:

1. **Пам'ять** — створення великої кількості клонованих об'єктів може займати більше пам'яті, ніж створення легких екземплярів «на льоту».
2. **Менше прозорості коду** — важко відстежувати, як саме створюється об'єкт, оскільки використовується клонування, а не явний конструктор.

3. **Може бути складно налагоджувати** — якщо об’єкти складні або містять посилання на інші ресурси, клонування може призвести до помилок або непередбачуваної поведінки.

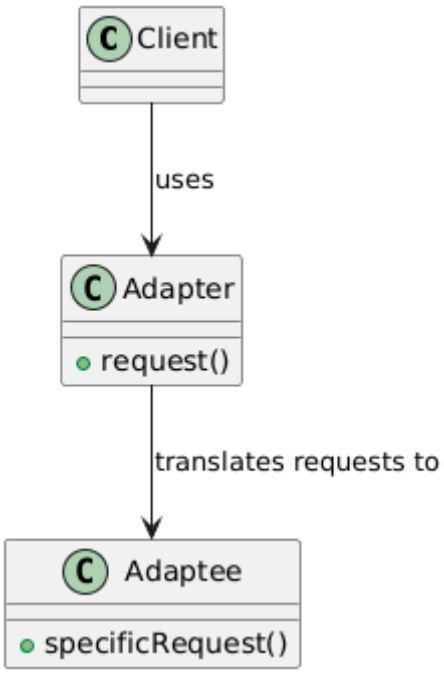
Висновок: У роботі реалізовано патерн Prototype для створення графічних об’єктів у редакторі, таких як коло та квадрат. Це дозволяє відокремити процес створення конкретного об’єкта від його використання, оскільки нові об’єкти отримуються шляхом клонування вже існуючих прототипів, а не створюються з нуля. Основною перевагою є швидке та гнучке додавання нових елементів без дублювання коду і можливість легко налаштовувати властивості об’єктів через прототипи. Недоліком може бути складність при клонуванні складних об’єктів із вкладеними структурами та підвищене навантаження на пам’ять при великій кількості клонів. Було створено діаграму класів, яка демонструє взаємодію компонентів системи: абстрактний прототип GraphicObject визначає метод clone, конкретні прототипи (Circle, Square) реалізують його, а панель інструментів (ToolPanel) керує клонуванням і надає об’єкти для використання у редакторі.

Відповіді на контрольні питання

1. Яке призначення шаблону «Адаптер»?

Шаблон «Adapter» (Адаптер) використовується для приведення інтерфейсу одного об’єкта до інтерфейсу іншого, з яким працює програма, без зміни внутрішнього коду об’єкта.

2. Нарисуйте структуру шаблону «Адаптер».



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

- Client — використовує інтерфейс адаптера.
- Target — визначає інтерфейс, з яким працює клієнт.

- Adapter — реалізує інтерфейс Target і делегує виклики Adaptee.
- Adaptee — клас з іншим інтерфейсом, який треба адаптувати.

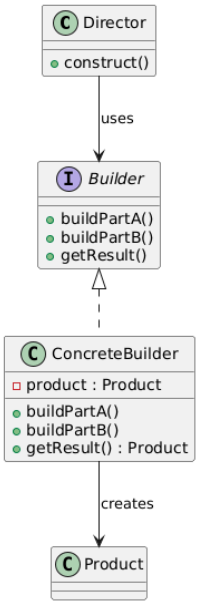
4. Яка різниця між реалізацією «Адаптера» на рівні об’єктів та на рівні класів?

Об’єктний адаптер — використовує композицію, Adapter містить посилання на Adaptee.

Класовий адаптер — використовує наслідування, Adapter наслідує Adaptee та реалізує Target.

5. Яке призначення шаблону «Будівельник»?
Шаблон «Builder» (Будівельник) використовується для поетапного створення складних об’єктів і відокремлює процес створення від представлення об’єкта.

6. Нарисуйте структуру шаблону «Будівельник».



7. Які класи входять в шаблон «Будівельник», та яка між ними взаємодія?

- Director — керує процесом побудови.
- Builder — визначає інтерфейс побудови частин об’єкта.
- ConcreteBuilder — реалізує побудову конкретного продукту.
- Product — кінцевий об’єкт, що створюється.

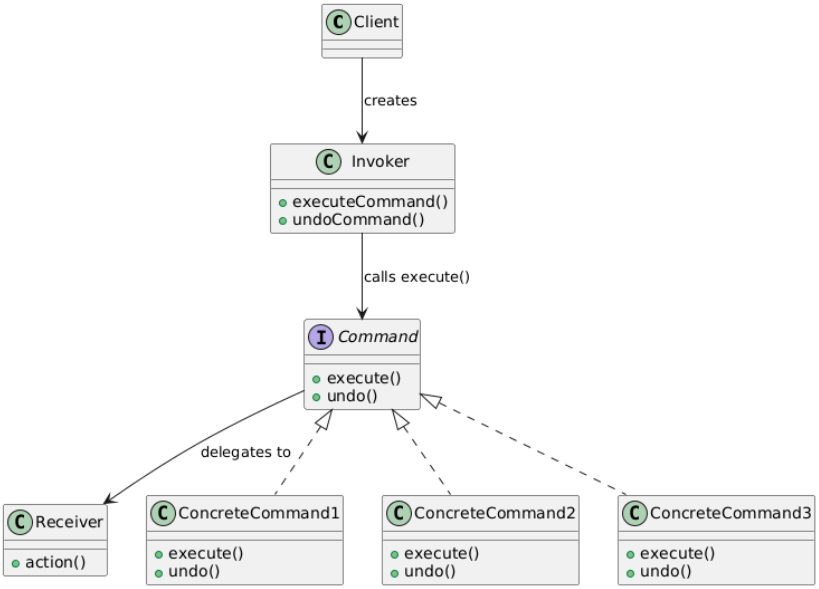
8. У яких випадках варто застосовувати шаблон «Будівельник»?

- Коли об’єкт має складний процес створення.
- Коли потрібні різні варіанти представлення одного об’єкта.

9. Яке призначення шаблону «Команда»?
Шаблон «Command» використовується для інкапсуляції дії у вигляді окремого

об’єкта, що дозволяє відокремити ініціатора дії від виконавця і реалізувати Undo/Redo, логування або черги команд.

10. Нарисуйте структуру шаблону «Команда».



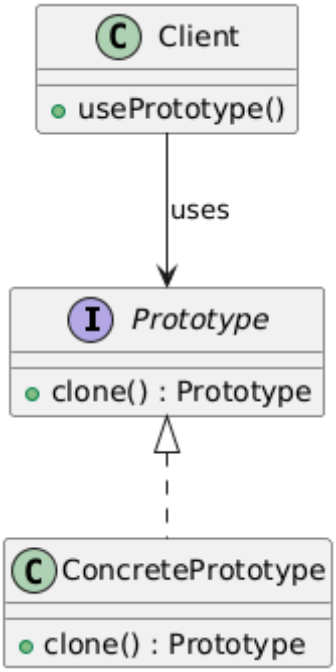
11. Які класи входять в шаблон «Команда», та яка між ними взаємодія?

- Command (інтерфейс) — визначає методи execute() і undo().
- ConcreteCommand — реалізує команду, делегуючи роботу Receiver.
- Receiver — виконує фактичну дію.
- Invoker — зберігає команду та викликає execute().
- Client — створює команду і передає її Invoker.

12. Розкажіть як працює шаблон «Команда».
Клієнт створює команду і передає її Invoker, який виконує команду через метод execute(). Receiver виконує конкретну операцію. Якщо потрібна відміна дії, Invoker викликає метод undo() конкретної команди.

13. Яке призначення шаблону «Прототип»?
Шаблон «Prototype» дозволяє створювати нові об’єкти шляхом копіювання існуючих прототипів, що спрощує створення складних об’єктів і зменшує ієрархію класів.

14. Нарисуйте структуру шаблону «Прототип».



15. Які класи входять в шаблон «Прототип», та яка між ними взаємодія?

- Prototype — визначає метод clone().
- ConcretePrototype — реалізує клонування конкретного об’єкта.
- Client — створює нові об’єкти шляхом клонування прототипу.

16. Які можна привести приклади використання шаблону «Ланцюжок відповідальності»?

- Обробка запитів на підпис документів через ланцюг співробітників.
- Фільтрація і обробка подій у GUI, де подія проходить через кілька обробників.
- Серверна обробка запитів, де кожен об’єкт ланцюга приймає або передає запит далі.