Архітектурні патерни: таємниця професійного програміста

доцент Андрій Креневич

https://github.com/krenevych/KickOffToDesignPatterns

Архітектура... навіщо?

Найважливіші метрики при розробці комерційних програмних продуктів:

вартість і **час** розробки

Повторне використання програмної архітектури та коду — це один з найбільш поширених **способів зниження вартості розробки**.

Основні проблеми, що не дозволяють пристосувати старий код до нових умов:

- занадто **тісні зв'язки** між компонентами,
- **залежність** коду **від конкретних** класів, а не абстрактних інтерфейсів,
- **вшиті** в код **операції**, які неможливо розширити

Шаблони проектування

Шаблони проектування програмного забезпечення

Шаблон проектування програмного забезпечення, (*патерн проектування, eng software design patterns*) – це **типовий спосіб вирішення певної проблеми**, що часто зустрічається при проектуванні архітектури програм.

- На відміну від готових функцій чи бібліотек, патерн не можна просто взяти й скопіювати в програму.

- Патерн це **не якийсь конкретний код**, а **загальний принцип** вирішення певної проблеми, який майже завжди треба підлаштовувати для потреб тієї чи іншої програми.

Патерни... навіщо?

Патерни проектування дозволяють розробникам створювати програми, які є більш ефективними, легкими для розуміння, підтримки та масштабування.

Класифікація (архітектурних) патернів

- **Породжуючі шаблони** (англ. Creational patterns) — це шаблони проєктування, що абстрагують процес побудови об'єктів.

- **Структурні патерни** (англ. structural patterns) — показують різні способи побудови зв’язків між об’єктами.

- **Поведінкові патерни** (англ. behavioral patterns) піклуються про ефективну комунікацію між об’єктами.

Історія

- Концепцію патернів вперше у 70х рр XX століття описав архітектор Крістофер Александер у книзі Мова шаблонів. Міста. Будівлі. Будівництво.

- У **1987** році Кент Бек (англ. Kent Beck) і Вард Каннігем (англ. Ward Cunningham) узяли ідеї Крістофера Александра та розробили шаблони відповідно до розробки програмного забезпечення для розробки графічних оболонок мовою Smalltalk.

- У **1988** році Ерік Ґамма (англ. Erich Gamma) почав писати докторську роботу про загальну переносимість цієї методики на розробку програм.

- У **1989–1991** роках Джеймс Коплін (англ. James Coplien) трудився над розробкою ідіом для програмування мовою C++ та опублікував у **1991** році книгу «Advanced C++ Idioms».

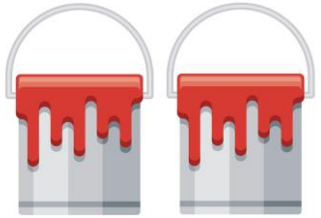
Історія

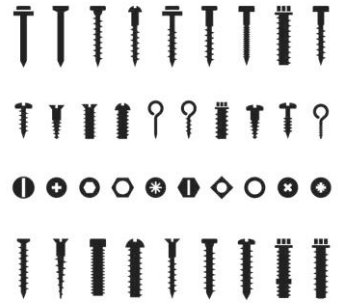
- У **1994** році Ерік Ґамма, Річард Гелм (англ. Richard Helm), Ральф Джонсон (англ. Ralph Johnson) та Джон Вліссідс (англ. John Vlissides) публікує книгу «Патерни проектування: повторно використовувані елементи архітектури об’єктно-орієнтованого програмного забезпечення» («**Design Patterns — Elements of Reusable Object-Oriented Software**»). Ця книга послужила приводом до прориву методу шаблонів.

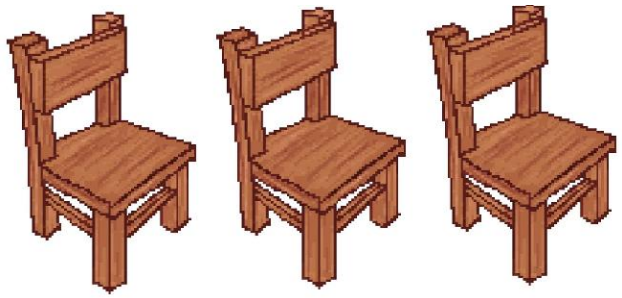
- Також команда авторів цієї книги відома суспільству під назвою Банда чотирьох (англ. **Gang of Four - GoF**).

Меблевий магазин


```
public class Shop {

    // Сировина для виготовлення меблів
    private String material = "Wood";  // матеріал
    private String color = "Crimson";  // фарба
    private int nails = 100;           // цвяхи, шурупи тощо

    // Список стільців, що є в наявності в магазині
    private final List<Chair> chairs = new ArrayList<>();
    // Список столів, що є в наявності в магазині
    private final List<Table> tables = new ArrayList<>();

    // Демонструвати меблі в салоні
    void demonstrateFurniture(){
        chairs.forEach(Chair::demo);
        tables.forEach(Table::demo);
    }
```



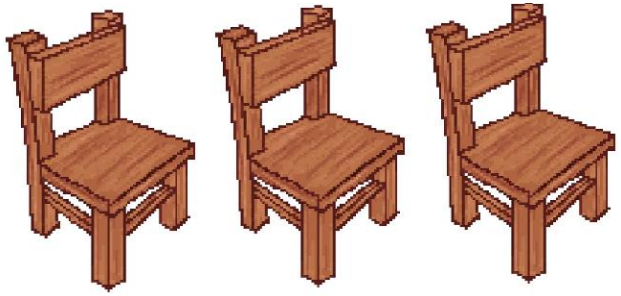




Меблевий магазин

```
public class Shop {

public enum Type {  // Типи меблів
    TABLE,
    CHAIR,
}

// Виготовити нові меблі за типом
void createFurniture(Type type){
    if (type == Type.CHAIR){
        Chair chair = new Chair(material, color, nails);
        chairs.add(chair);
    }else if (type == Type.TABLE){
        Table table = new Table(material, color, nails);
        tables.add(table);
    }
}
```

```
Shop shop = new Shop();   // Магазин з продажу меблів

// Створюємо асортимент меблів - 3 столи і 3
стільці
for ( int i = 0; i < 3; i++){
    shop.createFurniture(Shop.Type.CHAIR);
    shop.createFurniture(Shop.Type.TABLE);
}

// Демонструємо меблі у салоні
shop.demonstrateFurniture();
```



Запрошуємо мебляра

- Зведемо всі меблі до однієї ієрархії, що магазину працювати з усіма меблями незалежно від їхнього типу
- Делегуємо виготовлення меблів майстру, кожна конкретна реалізація якого буде виготовляти конкретний тип меблів

Майстер

```
interface FurnitureMaker {
   Furniture createFurniture(String material, String color, int nailNumber);
}
```

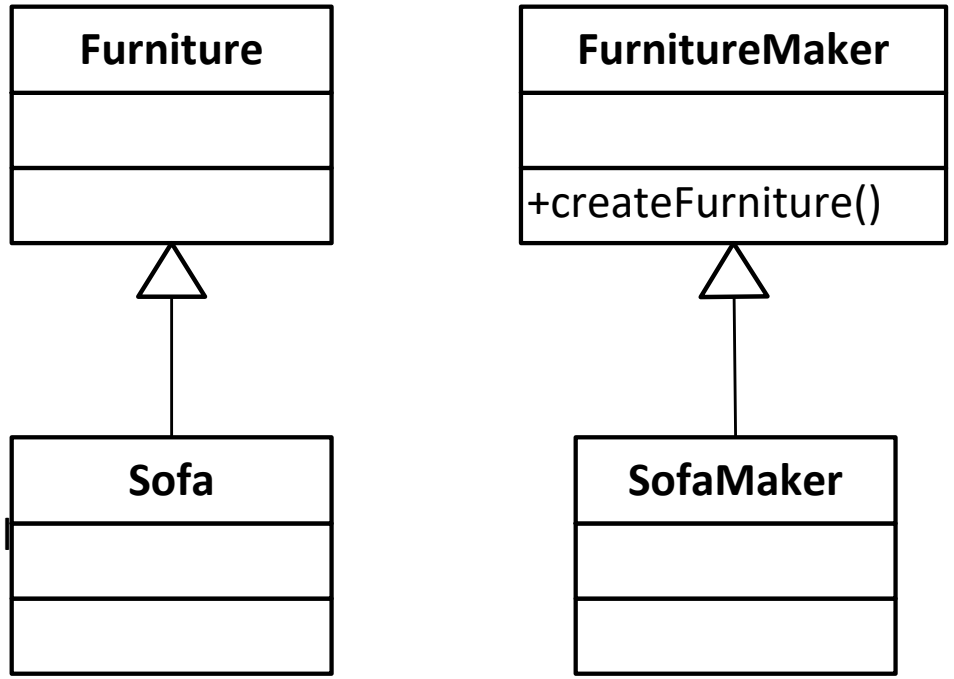
```
public class ChairMaker implements FurnitureMaker {
    @Override
    public Furniture createFurniture(String material, String color, int nailNumber) {
        return new Chair(material, color, nailNumber);
    }
}
```

```
public class Shop {
   // різні поля методи магазину

   // Список меблів, що є магазині
   private final List<Furniture> furnitureList = new ArrayList<>();

   // Виготовити нові меблі за типом
   void createFurniture(FurnitureMaker furnitureMaker){
      Furniture furniture = furnitureMaker.createFurniture(material, color, nails);
      furnitureList.add(furniture);
   }

}
```

```
// клієнтський код
FurnitureMaker chairMaker = new ChairMaker();
FurnitureMaker tableMaker = new TableMaker();

for ( int i = 0; i < 3; i++){
   shop.createFurniture(chairMaker);
   shop.createFurniture(tableMaker);
}
```

Розширення асортименту товарів у магазині

```
public class Sofa extends Furniture {
    public Sofa(String material, String color, int nailNumber) {
        super(material, color, nailNumber);
    }
}

public class SofaMaker implements FurnitureMaker {
    @Override
    public Furniture createFurniture(String material, String color, int nail
        return new Sofa(material, color, nailNumber);
    }
}
```




```
// клієнтський код
FurnitureMaker sofaMaker = new SofaMaker();
shop.createFurniture(sofaMaker);
```