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

із дисципліни *«Технології розроблення програмного забезпечення»*

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

Виконав:

Студент групи ІА-23 Хохол М.В.

Перевірив:

Мягкий М.Ю.

Київ 2024

Варіант №7

..7 Редактор зображень (state, prototype, memento, facade, composite, client-server)

Редактор зображень має такі функціональні можливості: відкриття/збереження зображень у найпопулярніших форматах (5 на вибір студента), застосування ефектів, наприклад поворот, розтягування, стиснення, кадрування зображення, можливість створення колажів шляхом «нашарування» зображень.

Завдання:

1. Ознайомитися з короткими теоретичними відомостями.

2. Реалізувати частину функціоналу робочої програми у вигляді класів та їх взаємодій для досягнення конкретних функціональних можливостей.

3. Застосування одного з розглянутих шаблонів при реалізації програми.

Зміст

Хід роботи:

Крок 1. Короткі теоретичні відомості.

1) Анти-патерн - це поширена, але невдала практика вирішення задач, яка на перший погляд може здаватися правильною, але насправді призводить до проблем в проєкті.

Приклад анти-патерна в управлінні розробкою ПЗ:

- Функції для галочки: Додавання в програму функцій, які непотрібні. Наприклад, додавання чат-бота в програму для обліку фінансів, який працює некоректно і нікому не потрібен.

Приклад анти-патерна в розробці ПЗ:

- Роздування інтерфейсу (Interface bloat): Виготовлення інтерфейсу дуже потужним і дуже важким для реалізації

Приклад анти-патерна в об'єктно-орієнтованому програмуванні

- Полтергейст (Poltergeist): Об'єкти, чиє єдине призначення - передавати інформацію іншим об’єктам;

Приклад анти-патерну в програмуванні:

- Потік лави (Lava flow): Збереження небажаного (зайвого або низькоякісного) коду через те, що його видалення занадто дороге або буде мати непередбачувані наслідки

Приклад методологічного анти-патерну:

- Передчасна оптимізація (Premature optimization): Оптимізація на основі недостатньої інформації

Приклад анти-патерну управління конфігурацією:

- Пекло залежностей (Dependency hell): Проблеми з версіями потрібних продуктів, особливо в системах UNIX / GNU / Linux

Приклад організаційного анти-патерну:

- Єдина обізнана людина (Single head of knowledge, SHOK): коли життєво важливими для проекту відомостями або навичками володіє тільки одна людина в команді; без неї робота зупиняється

2) Builder

Тип: Створюючий паттерн.
Призначення: Відокремлення процесу створення складного об'єкта від його представлення. Це дозволяє створювати різні варіації об'єктів із однаковим кодом побудови.

Ключові моменти:

- Використовується, коли об'єкт має багато параметрів.
- Забезпечує покрокову побудову об'єкта.

Приклад: Побудова автомобіля з варіантами двигуна, кольору та аксесуарів.

3) Command

Тип: Поведінковий паттерн.
Призначення: Інкапсулює запит як об'єкт, що дозволяє передавати, зберігати та виконувати його в різний час.

Ключові моменти:

- Забезпечує реалізацію **Undo/Redo**.
- Використовується для зниження залежності між відправником і виконавцем запитів.

Приклад: Кнопка "Зберегти" у текстовому редакторі виконує команду SaveCommand.

4) Chain of Responsibility

Тип: Поведінковий паттерн.
Призначення: Організовує послідовність обробників запиту так, щоб кожен обробник мав можливість або обробити запит, або передати його наступному в ланцюжку.

Ключові моменти:

- Зменшує залежність між відправником і отримувачем запиту.
- Ланцюжок може динамічно змінюватися.

Приклад: Система логування з різними рівнями (інформація, попередження, помилка).

5) Prototype

Тип: Створюючий паттерн.
Призначення: Створення нових об'єктів шляхом копіювання існуючих екземплярів (прототипів).

Ключові моменти:

- Корисний, коли створення об'єкта "з нуля" є дорогим або складним.
- Реалізується через метод **clone()**.

6) Adapter

Тип: Структурний паттерн.
Призначення: Дозволяє об'єктам з несумісними інтерфейсами працювати разом, обгортаючи один інтерфейс у потрібний для клієнта.

Ключові моменти:

- Виступає "перекладачем" між двома класами або системами.
- Адаптер може бути реалізований через обгортання об'єкта або наслідування.

Крок 2. Реалізація шаблону Prototype:

1. Інтерфейс Copyable<T>

   Призначення:

   - Це базовий контракт, який визначає метод copy().
   - T — це узагальнений тип (generics), що дозволяє класам, які реалізують цей інтерфейс, повертати копію свого типу.
   - Гарантує, що кожен клас, який реалізує цей інтерфейс, матиме метод для створення копій.

```
package com.proImg.image_editor.prototype;

8 usages  4 implementations
public interface Copyable<T> {
    4 implementations
    T copy();
}
```

Рис 1. Інтерфейс Copyable

2. Клас ProjectCloner

   Призначення:

   - Виконує роль "обгортки" для копіювання проектів.
   - Використовує метод copy() об'єкта Project для створення копії.

```
package com.proImg.image_editor.prototype;

import com.proImg.image_editor.entities.Project;

3 usages
public class ProjectCloner {

    2 usages
    private final Project project;

    1 usage
    public ProjectCloner(Project project) { this.project = project; }

    1 usage
    public Project cloneProject() { return project.copy(); }
}
```

Рис 2. Клас ProjectCloner

3. Метод copy() в класі Project

Призначення:

- Цей клас реалізує інтерфейс Copyable<Project>.
- Метод copy():
    - Створює новий об'єкт Project.
    - Копіює всі прості поля (name, editingEnabled, downloadEnabled).
    - Глибоко копіює вкладені об'єкти (collages і images) за допомогою їхніх методів copy().
    - Поля user і state копіюються напряму, оскільки вважаються незмінними (immutable).

```java
@Override
public Project copy() {
    Project copiedProject = new Project();
    copiedProject.setName(this.name);
    copiedProject.setEditingEnabled(this.editingEnabled);
    copiedProject.setDownloadEnabled(this.downloadEnabled);
    copiedProject.setUser(this.user);
    copiedProject.setMembers(this.members);

    if (this.collages != null) {
        copiedProject.setCollages(this.collages.stream()
                .map(Collage::copy)
                .collect(Collectors.toSet())
        );
    }
    if (this.images != null) {
        copiedProject.setImages(this.images.stream()
                .map(Image::copy)
                .collect(Collectors.toList())
        );
    }

    copiedProject.setState(this.state);
    return copiedProject;
}
```

Рис 3. Метод copy() в класі Project

4. Метод copy() в класі Image

Призначення:

- Реалізує інтерфейс Copyable<Image>.
- Метод copy():
    - Створює новий об'єкт Image.
    - Глибоко копіює масив байтів picByte.
    - Інші поля (project, cell) копіюються напряму.

```
@Override
public Image copy() {
    Image copiedImage = new Image(
            this.name,
            this.type,
            this.picByte.clone(),
            this.project,
            this.cell
    );
    copiedImage.setId(null);
    return copiedImage;
}
```

Рис 4. Метод copy() в класі Image

5. Метод copy() в класі Collage

Призначення:

- Реалізує інтерфейс Copyable<Collage>.
- Метод copy():
    - Створює новий об'єкт Collage.
    - Копіює всі прості поля (backgroundColor, gridGap, тощо).
    - Глибоко копіює вкладені об'єкти cells.

```java
@Override
public Collage copy() {
    Collage copiedCollage = new Collage();
    copiedCollage.setId(null);
    copiedCollage.setBackgroundColor(this.backgroundColor);
    copiedCollage.setGridGap(this.gridGap);
    copiedCollage.setGridX(this.gridX);
    copiedCollage.setGridY(this.gridY);
    copiedCollage.setWidth(this.width);
    copiedCollage.setHeight(this.height);
    copiedCollage.setPadding(this.padding);
    copiedCollage.setLeft(this.left);
    copiedCollage.setTop(this.top);
    copiedCollage.setProject(this.project);

    if (this.cells != null) {
        copiedCollage.setCells(this.cells.stream()
                .map(Cell::copy)
                .collect(Collectors.toSet()));
    }
    return copiedCollage;
}
```

Рис 5. Метод copy() в класі Collage

6. Метод copy() в класі Cell

Призначення:

- Реалізує інтерфейс Copyable<Cell>.
- Метод copy():
    - Створює новий об'єкт Cell.
    - Копіює всі прості поля (startRow, endRow, тощо).
    - Копіює image, якщо воно є.

```java
@Override
public Cell copy() {
    Cell copiedCell = new Cell();
    copiedCell.setId(null);
    copiedCell.setStartRow(this.startRow);
    copiedCell.setEndRow(this.endRow);
    copiedCell.setStartCol(this.startCol);
    copiedCell.setEndCol(this.endCol);
    copiedCell.setIndexInCollage(this.indexInCollage);
    copiedCell.setCollage(this.collage);

    copiedCell.setImage(this.image != null ? this.image.copy() : null);
    return copiedCell;
}
```

Рис 6. Метод copy() в класі Cell

7. Метод cloneProject в класі ProjectService
   Призначення:
   - Отримує оригінальний проект з бази даних.
   - Використовує ProjectCloner для створення копії.
   - Очищає ID скопійованого проекту, щоб Hibernate сприйняв його як новий об'єкт.
   - Зберігає копію в базі даних.

```java
1 usage
public Project cloneProject(Long projectId) {
    Project originalProject = projectRepository.findById(projectId)
            .orElseThrow(() -> new IllegalArgumentException("Project not found with ID: " + projectId));

    ProjectCloner projectCloner = new ProjectCloner(originalProject);
    Project clonedProject = projectCloner.cloneProject();

    clonedProject.setId(null);

    return projectRepository.save(clonedProject);
}
```

Рис 7. Метод cloneProject в класі ProjectService

8. Контролер ProjectController
   Призначення – зв’язок фронтенду та бекенду, передача id проекту який потрібно склонувати

```java
@PostMapping("/{projectId}/clone")
public ResponseEntity<Project> cloneProject(@PathVariable Long projectId) {
    try {
        Project clonedProject = projectService.cloneProject(projectId);
        return ResponseEntity.ok(clonedProject);
    } catch (IllegalArgumentException e) {
        return ResponseEntity.badRequest().body(null);
    }
}
```

Рис 8. Метод cloneProject в класі ProjectController

Детальніше код можна переглянути в репозиторії проекту:

https://github.com/Maxim-Khokhol/image-editor

Висновок: В цій лабораторній роботі я познайомився з паттернами Adapter, Builder, Command, Chain of responsibility, prototype. Було програмно реалізовано паттерн **Prototype**, що дозволило впровадити динамічне управління станами об'єкта Project, розділивши логіку на окремі класи та підвищивши гнучкість та підтримуваність системи.