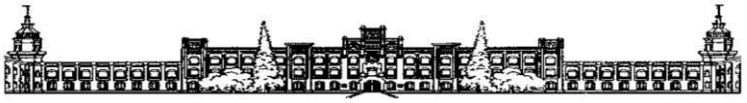

Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут імені Ігоря Сікорського” Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

**Лабораторна робота №4**

із дисципліни *«Технології розроблення програмного забезпечення»*

**Тема: «ШАБЛОНИ «SINGLETON», «ITERATOR», «PROXY», «STATE», «STRATEGY»»**

Виконав:

Студент групи ІА-23  Хохол М.В.

Перевірив:

Мягкий М.Ю.

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 їхньої взаємодії для досягнення конкретних функціональних можливостей.

3. Застосування одного з розглянутих шаблонів при реалізації програми.

# Зміст

**Хід роботи:**

**Крок 1. Короткі теоретичні відомості.**

1) **Singleton**

- Це шаблон, що гарантує існування тільки одного екземпляра класу і надає глобальну точку доступу до нього. Він використовується, коли необхідно забезпечити єдине джерело для спільного доступу, наприклад, для конфігураційних файлів або керування сеансами.

2) **Iterator**

- Iterator надає спосіб послідовного доступу до елементів колекції, не розкриваючи її внутрішню структуру. Він відокремлює функцію обходу від колекції, що дозволяє створювати різні способи навігації, такі як прямий чи зворотний обхід.

3) **Proxy**

- Proxy дозволяє створювати об'єкт-дублер для іншого об'єкта. Він виконує ті самі функції, що і основний об'єкт, але додає додаткову логіку, наприклад, контроль доступу, оптимізацію ресурсів або затримку створення основного об'єкта до моменту, коли він буде дійсно потрібен.

4) **State**

- Шаблон State дозволяє об'єкту змінювати свою поведінку залежно від його поточного стану. Кожен стан реалізується як окремий клас, що дозволяє організувати поведінку програми так, щоб вона динамічно змінювалась відповідно до зміни станів, наприклад, розблокований або заблокований телефон.

5) **Strategy**

- Шаблон Strategy дозволяє обирати один із кількох алгоритмів для виконання задачі. Він винесений в окремі класи, що дає змогу змінювати алгоритми "на льоту". Наприклад, у додатку-навігаторі можна використовувати різні стратегії прокладання маршруту: для автомобілів, пішоходів, громадського транспорту тощо.

**Крок 2. Реалізація шаблону State**

**1) Контекст — клас Project:**

- Клас **Project** виступає у ролі контексту. Він знає про своє поточне стан і має поле state, яке визначає, який саме стан активний. Стан обирається на основі значень полів editingEnabled та downloadEnabled. Метод setState() визначає, який саме стан потрібно використовувати, виходячи з комбінації цих значень.

2) **Стан — інтерфейс ProjectState та його реалізації**:

- Інтерфейс **ProjectState** визначає метод state(Project project, AnnouncementService announcementService), який реалізують конкретні стани. Це дає можливість змінювати поведінку об'єкта залежно від його стану, не змінюючи код контексту.

3) **Конкретні стани**: **EditableState, ClosedDownloadableState, ClosedState**

- **EditableState** — проект знаходиться в стані, де доступне редагування та завантаження зображень.
- **ClosedDownloadableState** — проект знаходиться в закритому стані, але зображення все ще доступні для завантаження.
- **ClosedState** — проект повністю закритий, тільки перегляд дозволений.

**AnnouncementService** — сервіс, що використовується для створення повідомлень про зміну стану проекту. Це дозволяє додавати функціонал, який залежить від стану проекту, в окремий сервіс, а не ускладнювати контекст.

**Чому обрано паттерн State:**

1) **Зручне керування поведінкою**: Паттерн "Стан" дозволяє відокремити логіку поведінки об'єкта в різних станах, що зменшує складність і робить код більш підтримуваним та розширюваним. Клас Project не обтяжений логікою для кожного стану, що полегшує його підтримку.
2) **Спрощення логіки переходу**: Метод setState() визначає, який стан потрібно використовувати, і це забезпечує автоматичне переключення між станами залежно від параметрів. Такий підхід дозволяє легко додавати нові стани без необхідності змінювати логіку контексту (Project).

```
5 usages   3 implementations
public interface ProjectState {

    2 usages   3 implementations
    void state(Project project, AnnouncementService announcementService);
}
```

Зображення 1. Реалізація інтерфейсу

```
3 usages
public class ClosedDownloadableState implements ProjectState {

    2 usages
    @Override
    public void state(Project project, AnnouncementService announcementService) {
        String message = String.format("Project '%s' by owner '%s' is in 'Closed - images available for download' state." +
                        " Only image downloads are allowed.",
                project.getName(), project.getUser().getUsername());

        announcementService.createAnnouncement(project, message);

        project.setEditingEnabled(false);
        project.setDownloadEnabled(true);
    }
}
```

Зображення 2. Реалізація стану ClosedDownloadableState

```
3 usages
public class ClosedState implements ProjectState {

    2 usages
    @Override
    public void state(Project project, AnnouncementService announcementService) {
        String message = String.format("Project '%s' by owner '%s' is in 'Closed' state. View only allowed.",
                project.getName(), project.getUser().getUsername());

        announcementService.createAnnouncement(project, message);

        project.setEditingEnabled(false);
        project.setDownloadEnabled(false);
    }
}
```

Зображення 3. Реалізація стану ClosedState

```
3 usages
public class EditableState implements ProjectState {

    2 usages
    @Override
    public void state(Project project, AnnouncementService announcementService) {
        String message = String.format("Project '%s' by owner '%s' is in 'Editable' state. " +
                        "Editing of the project and downloading of images are allowed.",
                project.getName(), project.getUser().getUsername());

        announcementService.createAnnouncement(project, message);

        project.setEditingEnabled(true);
        project.setDownloadEnabled(true);
    }
}
```

Зображення 4. Реалізація стану EditableState

```
1 usage
public Project updateProjectState(Long projectId, ProjectStateType stateType) {
    Project project = projectRepository.findById(projectId)
            .orElseThrow(() -> new RuntimeException("Project not found"));

    switch (stateType) {
        case CLOSED:
            project.setState(new ClosedState());
            break;
        case CLOSED_DOWNLOADABLE:
            project.setState(new ClosedDownloadableState());
            break;
        default:
            project.setState(new EditableState());
    }

    project.getState().state(project, announcementService);

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

Зображення 5. Оновлення стану проекту

```
3 usages
public void setState() {
    if (Boolean.TRUE.equals(editingEnabled) && Boolean.TRUE.equals(downloadEnabled)) {
        this.state = new EditableState();
    } else if (Boolean.FALSE.equals(editingEnabled) && Boolean.TRUE.equals(downloadEnabled)) {
        this.state = new ClosedDownloadableState();
    } else if (Boolean.FALSE.equals(editingEnabled) && Boolean.FALSE.equals(downloadEnabled)) {
        this.state = new ClosedState();
    } else {
        throw new IllegalStateException("Invalid project state configuration");
    }
}
```

Зображення 6. Метод, який визначає, який стан слід встановити для об'єкта Project

Детальніше код можна переглянути в репозиторії проекту:

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

**Висновок**: В цій лабораторній роботі я познайомився з паттернами **Singleton**, **Iterator**, **Proxy**, **State**, **Strategy**. Було програмно реалізовано паттерн **State**, що дозволило впровадити динамічне управління станами об'єкта Project, розділивши логіку на окремі класи та підвищивши гнучкість та підтримуваність системи.