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 №8

Технологія розробки програмного забезпечення

Тема роботи: «ШАБЛОНИ «COMPOSITE», «FLYWEIGHT», «INTERPRETER», «VISITOR»»

Виконала
студентка групи ІА-12:
Яковенко Дар’я

Перевірив:
вик. Колеснік В. М.

Київ 2023

Тема: ШАБЛОНИ «COMPOSITE», «FLYWEIGHT», «INTERPRETER», «VISITOR»

Мета: Ознайомитися з шаблонами, Реалізувати частину функціоналу робочої програми у вигляді класів та їх взаємодій для досягнення конкретних функціональних можливостей. Застосування одного з даних шаблонів при реалізації програми.

Завдання:
28 JSON Tool (ENG) (strategy, command, observer, template method, flyweight) Display JSON schema with syntax highlight. Validate JSON schema and display errors. Create user friendly table\list box\other for read and update JSON schema properties metadata (description, example, data type, format, etc.). Auto save\restore when edit, maybe history. Can check JSON value by schema (Put schema and JSON = valid\invalid, display errors). Export schema as markdown table. JSON to "flat" view.

Хід роботи

У моєму застосунку необхідно було реалізувати шаблон flyweight.
Патерн Flyweight застосовується для мінімізації використання оперативной пам’яті.

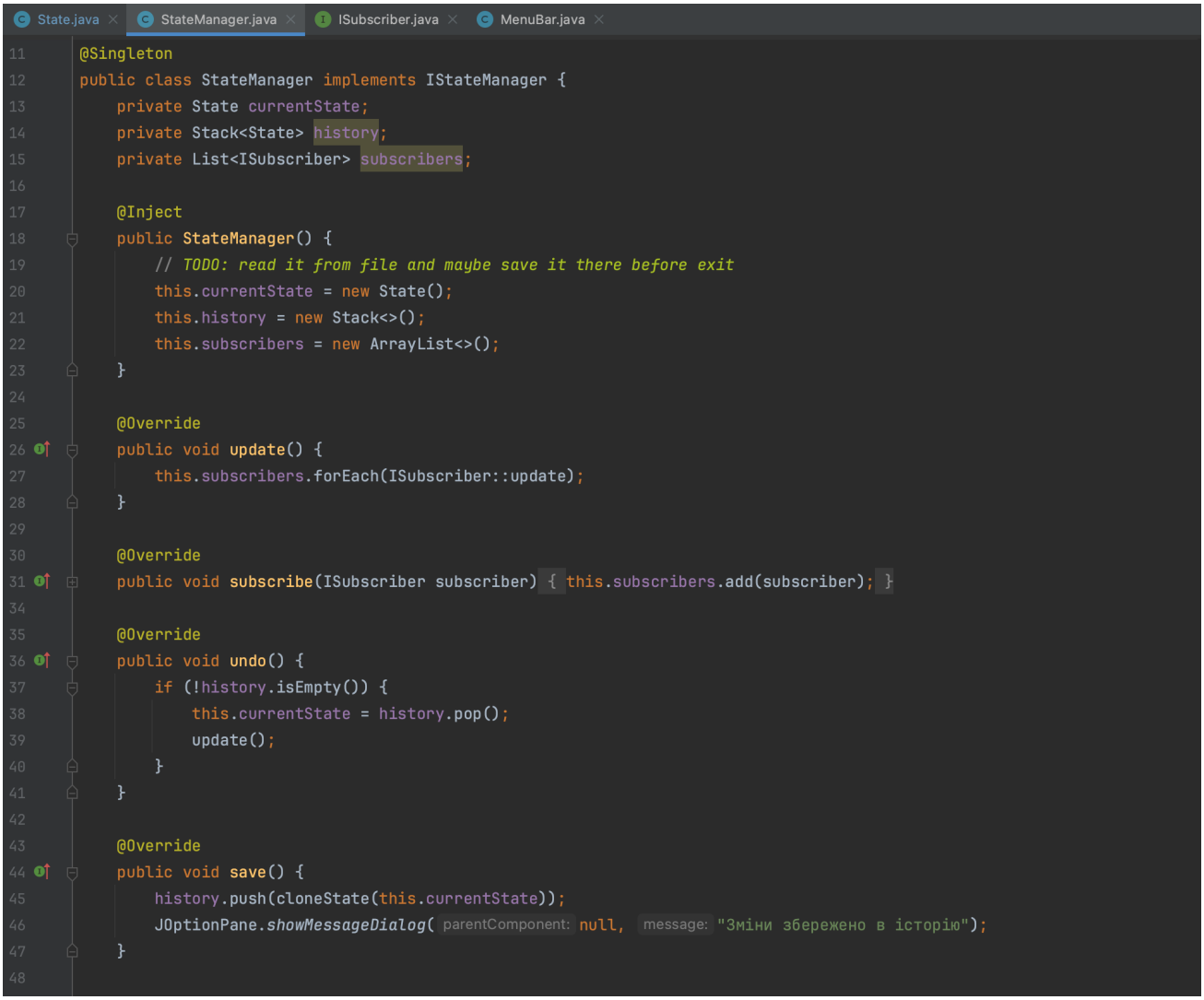
В моєму випадку можна розглядати об'єкти класу State як "легковажні" об'єкти, що містять дані, які можуть бути спільно використані багатьма іншими. Основна ідея полягає в тому, щоб виділити спільні дані в окремий клас. У мене це назва json файлу, назва файлу метаданих, шляхи до них, їх вміст, змінна поточного часу і дати. Ці змінні необхідні при створені нових файлів, коли користувач відкриває поточний файл, чи бажає зберегти його. Ці об’єкти являються

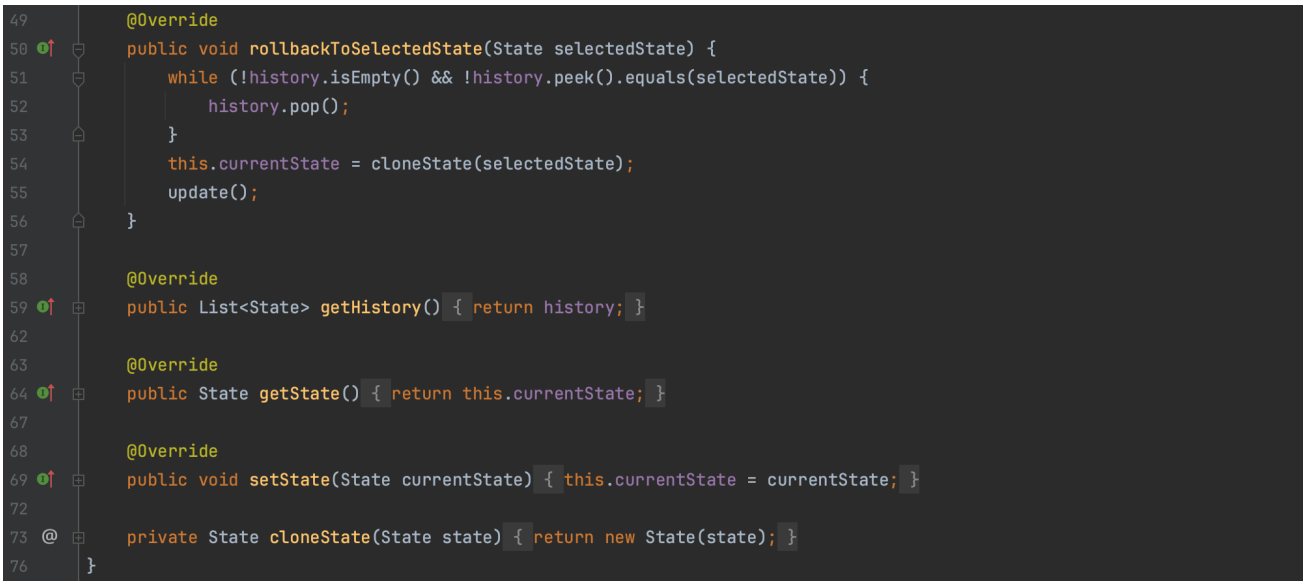
унікальними даними (зонішніми), оскільки для кожного нового файлу вони є різними. Клас State представляє стан конкретного проекту у системі.

State.java | StateManager.java | MenuBar.java

```java
public class State {
    public Map<String, String> metadata = new HashMap<>();
    public String json = "";
    public String currentMetadataFile = null;
    public Path currentJsonFilePath = null;
    public Path currentMetadataFilePath = null;
    public String currentJsonFile = null;
    public LocalDateTime dateTime = LocalDateTime.now();

    public State() {
    }

    // Clone constructor
    public State(State other) {
        this.json = other.json;
        this.currentMetadataFile = other.currentMetadataFile;
        this.currentJsonFile = other.currentJsonFile;
        this.currentJsonFilePath = other.currentJsonFilePath;
        this.currentMetadataFilePath = other.currentMetadataFilePath;
        this.dateTime = other.dateTime;

        // Deep clone of the Map
        this.metadata = new HashMap<>();
        for (Map.Entry<String, String> entry : other.metadata.entrySet()) {
            this.metadata.put(entry.getKey(), entry.getValue());
        }
    }

    @Override
    public String toString() {
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm");
        String formattedDateTime = dateTime.format(formatter);
        return
                currentJsonFile + '\n' +
                        formattedDateTime + '\n' + '\n';
    }
```

Власне загальними даними програми є те, що вона має містити State та історію змін. Поле **subscribers** та відповідні методи (**subscribe** та **update**) використовуються для реалізації патерну "Спостерігач" (Observer pattern).





Клас StateManager виступає як керівник стану та історії змін для проектів. У конструкторі створюється початковий об'єкт стану (currentState), а також

ініціалізуються стек для зберігання історії змін (history) та список підписників (subscribers). Основні методи, такі як undo, save, rollbackToSelectedState, дозволяють управляти станами проектів та їх історією.

Усі об'єкти State використовуються у програмі та зберігаються в стеку history. Це відповідає ідей шаблону Flyweight, оскільки об'єкти стану можуть бути повторно використані та вони управляються централізовано через StateManager.

Висновки:

У цій лабораторній роботі я ознайомилася з шаблонами, реалізувала додаткові класи відповідно до обраної теми, а також застосувала шаблон «Flyweight» для мінімізації використання оперативной пам’яті.