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

Технологія розробки програмного забезпечення

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

Виконала
студентка групи ІА-12:
Яковенко Дар’я

Перевірив:
вик. Колеснік В. М.

Київ 2023

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

Мета: Ознайомитися з шаблонами, Реалізувати частину функціоналу робочої програми у вигляді класів та їх взаємодій для досягнення конкретних функціональних можливостей.

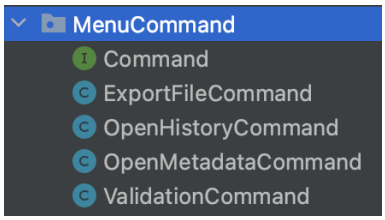
Завдання:

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 завданні зустрічається потрібно реалізувати шаблон «COMMAND» (Шаблон "command" (команда) перетворить звичайний виклик методу в клас. Таким чином дії в системі стають повноправними об'єктами).

Оскільки моя програма схожа з програмою текстового редактору – то застосуємо цей шаблон для кнопок. Створимо підкласи для кожної кнопки та перевизначимо в них методи дії для різних завдань.



Command.java

```
1  package main.MenuCommand;
2
3  public interface Command {
4      void execute();
5  }
```

OpenMetadataCommand.java

```java
package main.MenuCommand;

import metadata.MetadataEditor;

public class OpenMetadataCommand implements Command {

    private void showMetadata() {
        MetadataEditor metadataEditor = new MetadataEditor();
        metadataEditor.setVisible(true);
    }

    @Override
    public void execute() {
        showMetadata();
    }
}
```

Також перевагою цього шаблону є те, що він дозволяє збирати складні команди з простих.

ExportFileCommand.java

```java
package main.MenuCommand;

import ...

public class ExportFileCommand implements Command {
    private final String exportType;
    private final String json;

    public ExportFileCommand(String exportType, String json) {
        this.exportType = exportType;
        this.json = json;
    }

    @Override
    public void execute() { chooseExportOption(); }

    private void chooseExportOption() {
        if (json.isEmpty()) {
            MessageDisplay.showError("JSON editor is empty. Enter JSON before exporting.");
            return;
        }

        ExportStrategy strategy = switch (exportType) {
            case "table" -> new TableExport();
            case "text" -> new TextExport();
            default -> null;
        };

        if (strategy != null) {
            strategy.exportJson(json);
        }
    }
}
```

AppMenu.java

```java
public class AppMenu extends JMenuBar {
    public Command openMetadata;
    public Command openHistory;

    public AppMenu(JsonToolApp parent) {
        openMetadata = new OpenMetadataCommand();
        openHistory = new OpenHistoryCommand();

        JMenu fileMenu = new JMenu( s: "Файл");
        JMenuItem createNewFile = new JMenuItem( text: "Створити новий");
        createNewFile.addActionListener(e -> createNew());
        JMenuItem openFile = new JMenuItem( text: "Відкрити існуючий...");
        openFile.addActionListener(e -> openExisting());

        JMenuItem metadataItem = new JMenuItem( text: "Метадані");
        metadataItem.addActionListener(e -> openMetadata.execute());

        JMenuItem historyItem = new JMenuItem( text: "Історія");
        historyItem.addActionListener(e -> openHistory.execute());
        this.add(historyItem);

        JMenu exportMenu = new JMenu( s: "Експорт");
        JMenuItem exportTable = new JMenuItem( text: "Таблиця");
        JMenuItem exportText = new JMenuItem( text: "Текст");

        exportTable.addActionListener(e -> {
            String json = parent.getAppBody().getJsonText();
            Command exportCommand = new ExportFileCommand( exportType: "table", json);
            exportCommand.execute();
        });

        exportText.addActionListener(e -> {
            String json = parent.getAppBody().getJsonText();
            Command exportCommand = new ExportFileCommand( exportType: "text", json);
            exportCommand.execute();
        });
```

Висновки:

У даній лабораторній роботі я ознайомилася з шаблонами, реалізувала додаткові класи відповідно до обраної теми, а також застосувала шаблон «COMMAND» для кнопок у створеному застосунку.