



Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Лабораторна робота №5
Технологія розробки програмного забезпечення
*«ШАБЛОНИ «ADAPTER», «BUILDER», «COMMAND», «CHAIN OF
RESPONSIBILITY», «PROTOTYPE».»*
Варіант 28

Виконала:
студентка групи ІА–13
Хижняк Валерія Валеріївна

Перевірив:
Мягкий М.Ю.

Київ 2023

Тема: Шаблони “Adapter”, “Builder”, “Command”, “Chain of responsibility”, “Prototype”

Мета: ознайомитись з шаблонами “Adapter”, “Builder”, “Command”, “Chain of responsibility”, “Prototype”, реалізувати частину функціоналу робочої програми у вигляді класів та їхньої взаємодії для досягнення конкретних функціональних можливостей, застосувати один з шаблонів при реалізації програми.

Хід роботи:

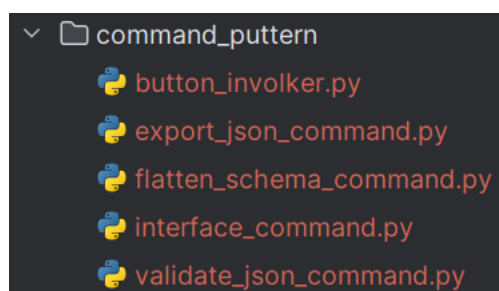
Варіант:

..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.

Згідно з моїм варіантом, необхідно реалізувати шаблон Command. Суть цього шаблону полягає в тому, щоб перетворити звичайний виклик методу в клас, щоб дії в системі стали повноправними об'єктами. Об'єкт команда не виконує ніяких дій окрім перенаправлення запиту одержувачеві. Патерн Command дозволяє розділити клас, який викликає операцію, від класу, який її виконує.

В моєму проекті я реалізувала цей шаблон для створення кнопок панелі інструментів. Реалізація цього патерну знаходиться в директорії command_puttern.



`interface_command` це інтерфейс, який декларує метод виконання (`execute`).

Кожен конкретний клас команди повинен реалізувати цей інтерфейс.

```
1 from abc import ABC, abstractmethod
2
3
4 class InterfaceCommand(ABC):
5     @abstractmethod
6     def execute(self):
7         pass
```

`validate_json_command` це клас, який реалізує інтерфейс `interface_command`. Він містить конкретний об'єкт, який отримує в якості параметру та метод `execute()`, який викликається для виконання команди. В нашому випадку в методі `execute()` викликається метод валідації `json` з класу `JSONSchemaService()`.

```
1 from app.services.json_schema_service import JSONSchemaService
2 from app.views.command_patterner.interface_command import InterfaceCommand
3
4
5 class ValidateJSONCommand(InterfaceCommand):
6     # Створюємо об'єкт класу JSONSchemaService()
7     json_schema_service = JSONSchemaService()
8
9     def __init__(self, schema_text):
10         super().__init__()
11         self.schema_text = schema_text
12
13     def execute(self):
14         # Викликаємо метод валідації json
15         self.json_schema_service.validate_schema(self.schema_text)
16         print("Validation completed!")
```

`export_json_command` це клас, який реалізує інтерфейс `interface_command`.

Він містить конкретний об'єкт, який отримує в якості параметру та метод

execute(), в якому викликається метод експорту json як Markdown з класу JSONSchemaService().

```
1 from app.services.json_schema_service import JSONSchemaService
2 from app.views.command_puttern.interface_command import InterfaceCommand
3
4
5 2 usages
6 class ExportJSONCommand(InterfaceCommand):
7     # Створюємо об'єкт класу JSONSchemaService()
8     json_schema_service = JSONSchemaService()
9
10     def __init__(self):
11         super().__init__()
12
13 13 usages (13 dynamic)
14 def execute(self):
15     # Викликаємо метод експорту json як таблиці Markdown
16     self.json_schema_service.export_schema()
17     print("Export completed!")
```

flatten_schema_command це клас, який реалізує інтерфейс interface_command. Він містить конкретний об'єкт, який отримує в якості параметру та метод execute(), в якому викликається метод перетворення json в “flatten” вигляд з класу JSONSchemaService().

```
1 from app.services.json_schema_service import JSONSchemaService
2 from app.views.command_puttern.interface_command import InterfaceCommand
3
4
5 2 usages
6 class FlattenSchemaCommand(InterfaceCommand):
7     # Створюємо об'єкт класу JSONSchemaService()
8     json_schema_service = JSONSchemaService()
9
10     def __init__(self):
11         super().__init__()
12
13 13 usages (13 dynamic)
14 def execute(self):
15     # Викликаємо метод для перетворення json в flatten вигляд
16     self.json_schema_service.flatten_schema()
17     print("JSON to flatten view completed!")
```

ButtonInvoker це об'єкт, який знає, як викликати команду, але не знає про конкретний клас команди. Він викликає метод `execute` для об'єкту `Command`, який отримує як параметр конструктора.

```
4 usages
1 class ButtonInvoker:
2     def __init__(self, command):
3         self.command = command
4
5     3 usages
6     def invoke(self):
7         # Викликаємо метод execute() для команди
8         self.command.execute()
```

Створення кнопок відбувається в класі `toolbar_view`.

```
1 from tkinter import *
2 from app.services.json_schema_service import JSONSchemaService
3 from app.views.command_pattern.button_invoker import ButtonInvoker
4 from app.views.command_pattern.export_json_command import ExportJSONCommand
5 from app.views.command_pattern.flatten_schema_command import FlattenSchemaCommand
6 from app.views.command_pattern.validate_json_command import ValidateJSONCommand
7
8
9 2 usages new *
10 class ToolbarView:
11     # Створюємо об'єкт класу JSONSchemaService()
12     json_schema_service = JSONSchemaService()
13
14     new *
15     def __init__(self):
16         super().__init__()
```

```

17 def create_toolbar(self, main_frame, schema_text):
18     # Створюємо об'єкт панелі інструментів
19     self.toolbar = Frame(main_frame)
20     # Створюємо команду валідації
21     validate_json_command = ValidateJSONCommand(schema_text)
22     # Додаємо кнопку для валідації JSON-схеми
23     self.validate_button = Button(self.toolbar, text="Валідувати", command=ButtonInvoker(validate_json_command).invoke)
24     self.validate_button.grid(row=0, column=0, pady=20, padx=0)
25     # Створюємо команду експорту
26     export_json_command = ExportJSONCommand()
27     # Додаємо кнопку для експорту JSON-схеми як таблиці Markdown
28     self.export_button = Button(self.toolbar, text="Експортувати", command=ButtonInvoker(export_json_command).invoke)
29     self.export_button.grid(row=1, column=0, pady=20, padx=0)
30     # Створюємо команду перетворення JSON-схеми в "flatten" вигляд
31     flatten_schema_command = FlattenSchemaCommand()
32     # Додаємо кнопку для перетворення JSON-схеми в "flatten" вигляд
33     self.flatten_button = Button(self.toolbar, text="Flatten view", command=ButtonInvoker(flatten_schema_command).invoke)
34     self.flatten_button.grid(row=2, column=0, pady=20, padx=0)
35     return self.toolbar

```

2. Реалізувати не менше 3-х класів відповідно до обраної теми.

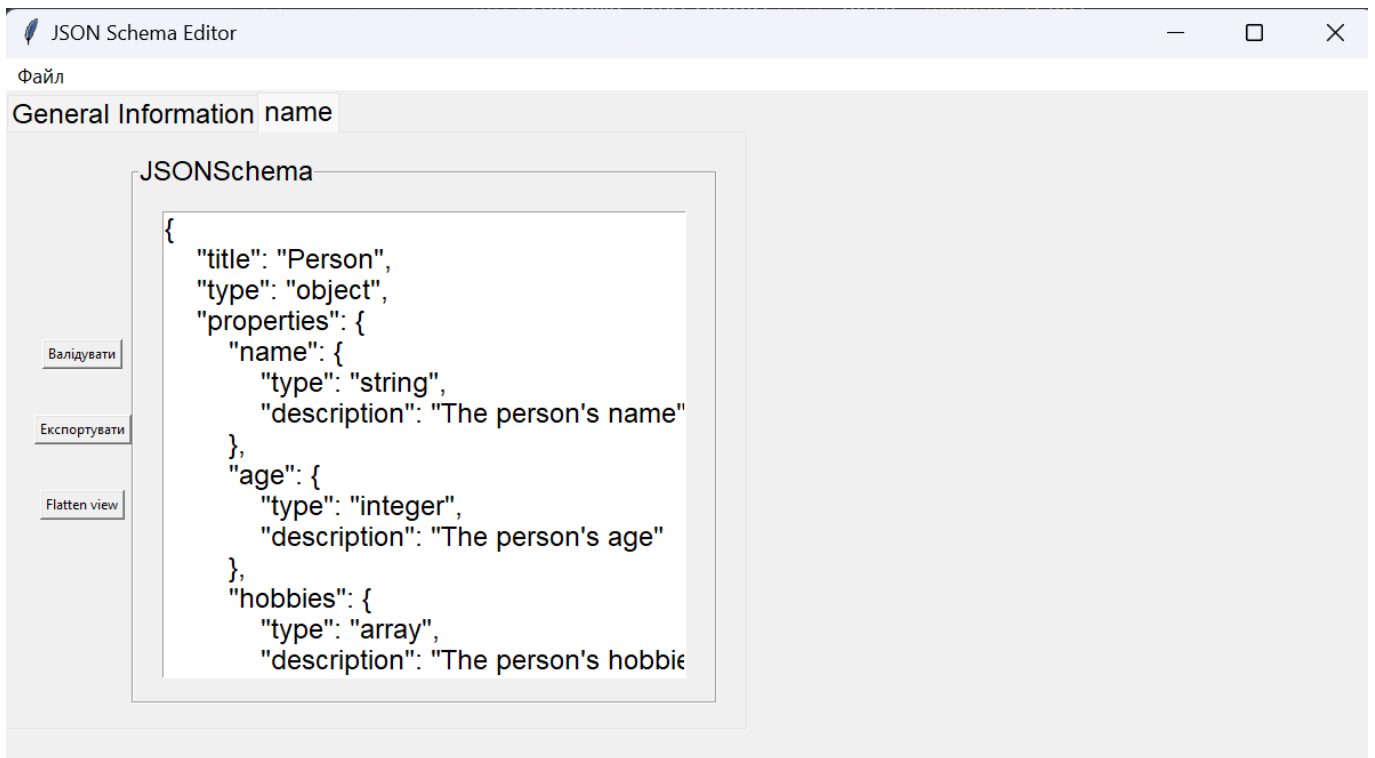
В ході лабораторної роботи було розроблено клас `toolbar_view` для відображення панелі інструментів, а також розроблено метод валідації json в класу `json_schema_service`.

```

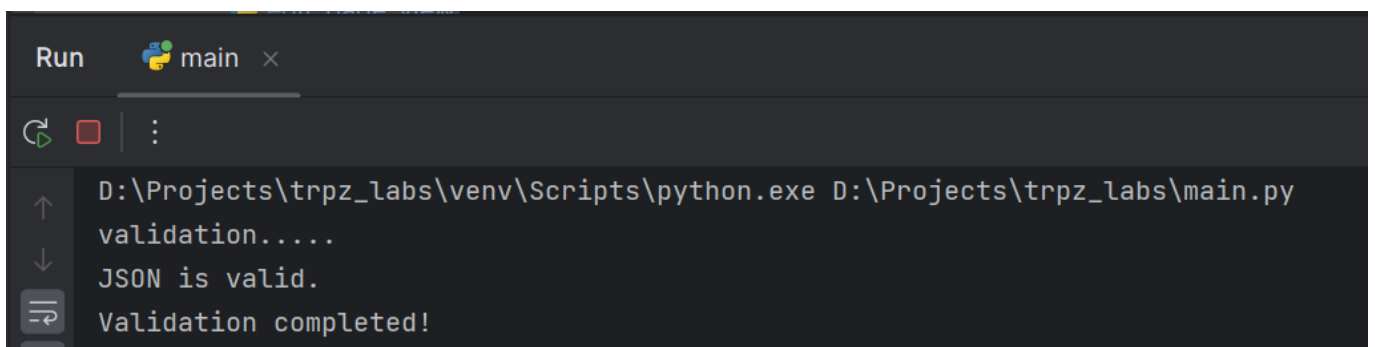
58 def validate_schema(self, json_text_frame):
59     print("validation....")
60     # Записуємо вміст редактора в змінну
61     json_text = json_text_frame.get(1.0, END)
62     try:
63         json.loads(json_text)
64         print("JSON is valid.")
65         return True
66     except json.JSONDecodeError as e:
67         print(f"JSON is not valid. Error: {e}")
68         return False

```

В результаті роботи програми ми отримали наступний результат:



Валідація працює наступним чином:



Висновок: на даній лабораторній роботі я ознайомилась з шаблонами “Adapter”, “Builder”, “Command”, “Chain of responsibility”, “Prototype”, реалізувала частину функціоналу робочої програми у вигляді класів та їхньої взаємодії для досягнення конкретних функціональних можливостей, застосувала один з шаблонів при реалізації програми.