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”, “Proxy”, “Command”, “Chain of responsibility”, “Prototype”

Мета: ознайомитись з шаблонами “Adapter”, “Builder”, “Proxy”, “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:

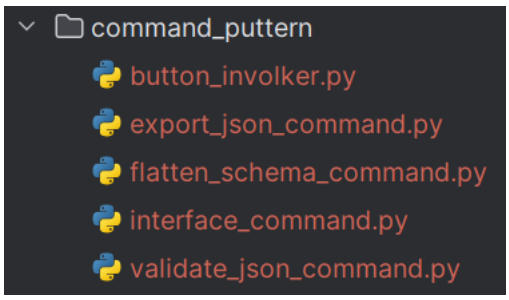
Варіант:

..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 інстументів. 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.

```python
from abc import ABC, abstractmethod


6 usages
class InterfaceCommand(ABC):
    13 usages (13 dynamic)
    @abstractmethod
    def execute(self):
        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().

```python
from app.services.json_schema_service import JSONSchemaService
from app.views.command_puttern.interface_command import InterfaceCommand


2 usages
class ValidateJSONCommand(InterfaceCommand):
    # Створюємо об'єкт класу JSONSchemaService()
    json_schema_service = JSONSchemaService()

    def __init__(self, schema_text):
        super().__init__()
        self.schema_text = schema_text

    13 usages (13 dynamic)
    def execute(self):
        # Викликаємо метод валідації json
        self.json_schema_service.validate_schema(self.schema_text)
        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().

```python
from app.services.json_schema_service import JSONSchemaService
from app.views.command_puttern.interface_command import InterfaceCommand


2 usages
class ExportJSONCommand(InterfaceCommand):
    # Створюємо об'єкт класу JSONSchemaService()
    json_schema_service = JSONSchemaService()

    def __init__(self):
        super().__init__()

    13 usages (13 dynamic)
    def execute(self):
        # Викликаємо метод експорту json як таблиці Markdown
        self.json_schema_service.export_schema()
        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().

```python
from app.services.json_schema_service import JSONSchemaService
from app.views.command_puttern.interface_command import InterfaceCommand


2 usages
class FlattenSchemaCommand(InterfaceCommand):
    # Створюємо об'єкт класу JSONSchemaService()
    json_schema_service = JSONSchemaService()

    def __init__(self):
        super().__init__()

    13 usages (13 dynamic)
    def execute(self):
        # Викликаємо метод для перетворення json в flatten вигляд
        self.json_schema_service.flatten_schema()
        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.

```python
4 usages
class ButtonInvoker:
    def __init__(self, command):
        self.command = command

    3 usages
    def invoke(self):
        # Виликаємо метод execute() для команди
        self.command.execute()
```

Створення кнопок відбувається в класі toolbar_view.

```python
from tkinter import *
from app.services.json_schema_service import JSONSchemaService
from app.views.command_puttern.button_involker import ButtonInvoker
from app.views.command_puttern.export_json_command import ExportJSONCommand
from app.views.command_puttern.flatten_schema_command import FlattenSchemaCommand
from app.views.command_puttern.validate_json_command import ValidateJSONCommand


2 usages  new *
class ToolbarView:
    # Створюємо об'єкт класу JSONSchemaService()
    json_schema_service = JSONSchemaService()

    new *
    def __init__(self):
        super().__init__()
```

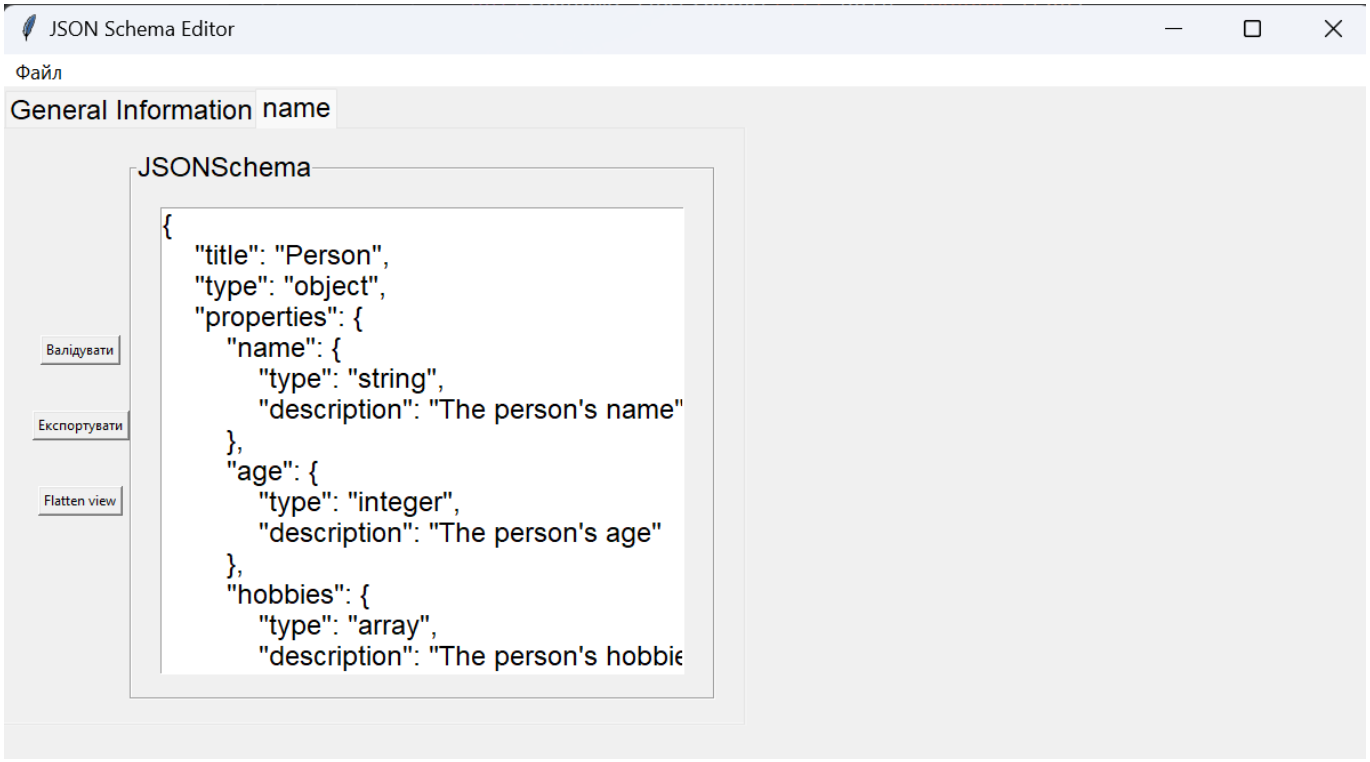
```python
    def create_toolbar(self, main_frame, schema_text):
        # Створюємо об'єкт панелі інструментів
        self.toolbar = Frame(main_frame)
        # Створюємо команду валідації
        validate_json_command = ValidateJSONCommand(schema_text)
        # Додаємо кнопку для валідації JSON-схеми
        self.validate_button = Button(self.toolbar, text="Валідувати", command=ButtonInvoker(validate_json_command).invoke)
        self.validate_button.grid(row=0, column=0, pady=20, padx=0)
        # Створюємо команду експорту
        export_json_command = ExportJSONCommand()
        # Додаємо кнопку для експорту JSON-схеми як таблиці Markdown
        self.export_button = Button(self.toolbar, text="Експортувати", command=ButtonInvoker(export_json_command).invoke)
        self.export_button.grid(row=1, column=0, pady=20, padx=0)
        # Створюємо команду перетворення JSON-схеми в "flatten" вигляд
        flatten_schema_command = FlattenSchemaCommand()
        # Додаємо кнопку для перетворення JSON-схеми в "flatten" вигляд
        self.flatten_button = Button(self.toolbar, text="Flatten view", command=ButtonInvoker(flatten_schema_command).invoke)
        self.flatten_button.grid(row=2, column=0, pady=20, padx=0)
        return self.toolbar
```

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

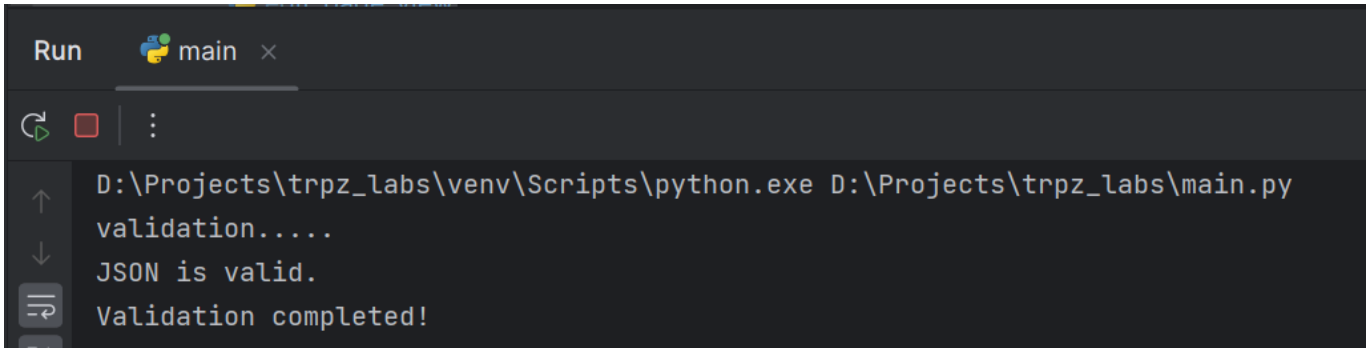
В ході лабораторної роботи було розроблено клас toolbar_view для відображення панелі інструментів, а також розроблено метод валідації json в класу json_schema_service.

```python
    1 usage  new *
    def validate_schema(self, json_text_frame):
        print("validation.....")
        # Записуємо вміст редактора в змінну
        json_text = json_text_frame.get(1.0, END)
        try:
            json.loads(json_text)
            print("JSON is valid.")
            return True
        except json.JSONDecodeError as e:
            print(f"JSON is not valid. Error: {e}")
            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”, “Proxy”, “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.