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 №6

Технологія розробки програмного забезпечення

*Тема роботи: «*Шаблони «Abstract Factory», «Factory Method», «Memento», «Observer», «Decorator»

Виконала
студентка групи ІА-12:
Яковенко Дар’я

Перевірив:
вик. Колеснік В. М.

Київ 2023

Тема: Шаблони «Abstract Factory», «Factory Method», «Memento», «Observer», «Decorator»

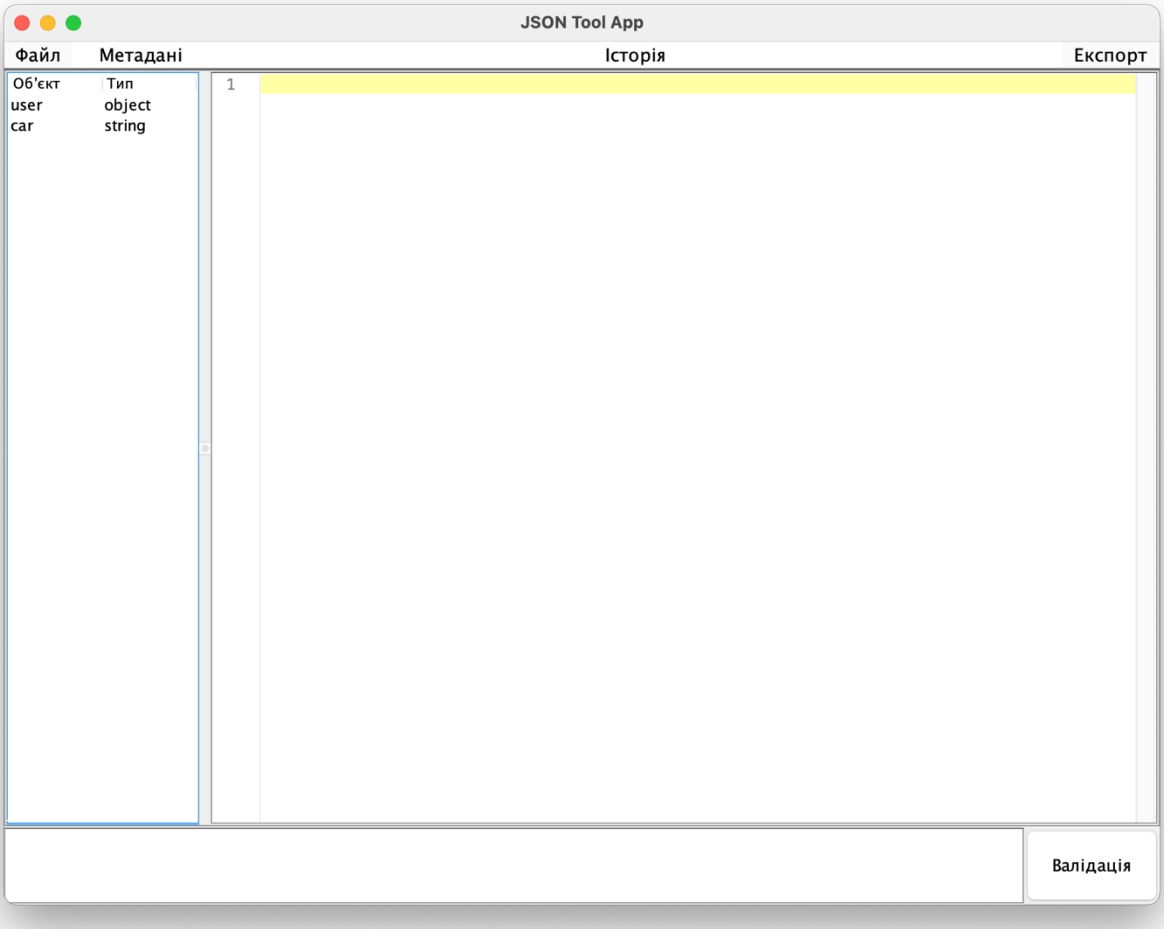
Мета: Ознайомитися з шаблонами, Реалізувати частину функціоналу робочої програми у вигляді класів та їх взаємодій для досягнення конкретних функціональних можливостей. Застосування одного з даних шаблонів при реалізації програми.

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 завданні зустрічається потрібно реалізувати шаблон «Observer» (Шаблон визначає залежність "один-до-багатьох" таким чином, що коли один об'єкт змінює власний стан, усі інші об'єкти отримують про це сповіщення і мають можливість змінити власний стан також).
У своїй програмі я розділила основне вікно на дві частини: на поле для введення json та таблиці, що буде відображати додані метадані, оскільки так буде

зручніше працювати з редактором:



Клас, що реалізує відображення метаданих, повинен «підписатися» на те, щоб дізнаватися про оновлення метаданих властивостей. Саме тому необхідно реалізувати шаблон “Observer”.

Observer.java | State.java | StateManager.java | TableComponent.java

```java
package state;

public interface Observer
{
    void update(State state);
}
```

Observer.java | State.java | StateManager.java | TableComponent.java | metada

```java
package state;

import java.util.HashMap;
import java.util.Map;

public class State
{
    public Map<String, String> tableComponentMap = new HashMap<>();

}
```

Observer.java | State.java | StateManager.java | TableComponent.java | metadata.json

```java
package state;

import java.util.ArrayList;
import java.util.List;

public class StateManager
{
    State state = new State();
    private final List<Observer> observers = new ArrayList<>();

    public State getState() { return state; }

    public void setState(State state)
    {
        this.state = state;
        notifyObservers();
    }

    public void subscribe(Observer observer) { observers.add(observer); }

    private void notifyObservers()
    {
        for (Observer observer : observers)
        {
            observer.update(state);
        }
    }
}
```

Візьмемо для прикладу класи MetadataEditor, TableComponent, MetadataTableComponent і приклад додавання рядка в таблицю:

```
public MetadataTableComponent(DefaultTableModel tableModel, StateManager stateManager)
{
    super(tableModel);
    this.stateManager = stateManager;
    this.tableModel = tableModel;
    stateManager.subscribe( observer: this);
```

```
public TableComponent(DefaultTableModel tableModel, StateManager stateManager)
{
    super(tableModel);
    this.stateManager = stateManager;
    this.tableModel = tableModel;
    stateManager.subscribe( observer: this);
```

Підписка на отримання сповіщень про зміни.

```
private void addRow() {
    State state = stateManager.getState();
    String name = nameField.getText();
    String type = (String) typeComboBox.getSelectedItem();
    state.tableComponentMap.put(name, type);
    stateManager.setState(state);
}
```

В класі MetadataEditor при додаванні об’єкту ми дізнаємося поточний стан програми із StateManager. Отримуємо з полів назву та тип об’єкту і додаємо новий запис в мапу tableComponentMap об'єкта state. А також здійснюємо оновлення стану програми за допомогою stateManager, і тим самим сповіщаємо класи, які підписані, про зміну.

Висновки:

У даній лабораторній роботі я ознайомилася з шаблонами, реалізувала додаткові класи відповідно до обраної теми, а також застосувала шаблон «Observer» для оновлення таблиць метаданих.