



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

з дисципліни «Технології розроблення програмного забезпечення»

Тема: «Патерни проектування»

Виконав:

студент групи IA-331

Курбатов Кіріл Андрійович

Перевірив:

Амонс Олександр Анатолійович

Мета: Вивчити структуру шаблонів «Composite», «Flyweight» (Пристосуванець), «Interpreter», «Visitor» та навчитися застосовувати їх в реалізації програмної системи.

Завдання

- Ознайомитись з короткими теоретичними відомостями.
- Реалізувати частину функціоналу робочої програми у вигляді класів та їхньої взаємодії для досягнення конкретних функціональних можливостей.
- Реалізувати один з розглянутих шаблонів за обраною темою.
- Реалізувати не менше 3-х класів відповідно до обраної теми.
- Підготувати звіт щодо виконання лабораторної роботи. Поданий звіт повинен містити: діаграму класів, яка представляє використання шаблону в реалізації системи, навести фрагменти коду по реалізації цього шаблону

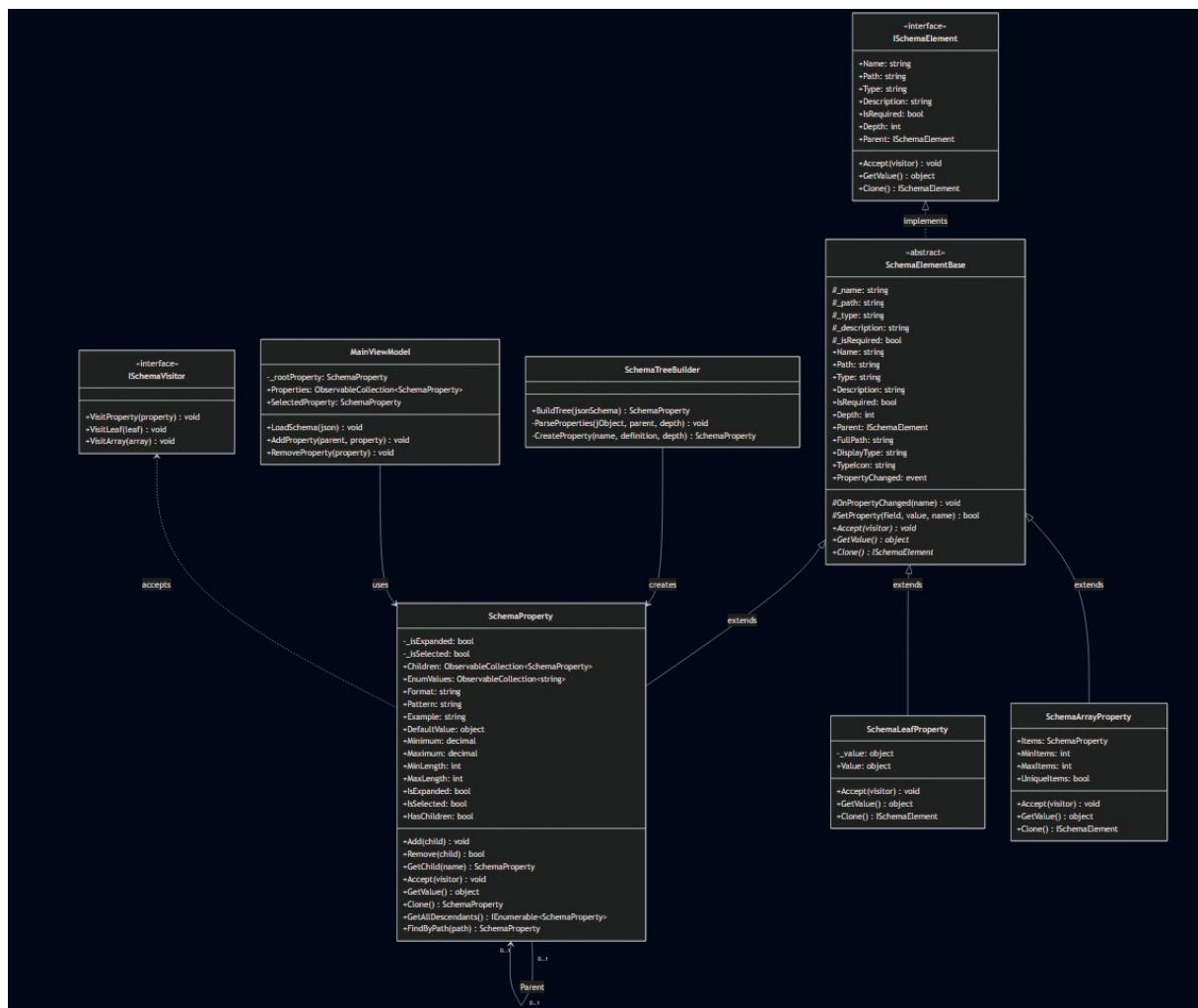


Рис. 8.1 — Діаграма класів патерну Composite для ієархії JSON Schema

Реалізація патерну Composite

```
using Newtonsoft.Json.Linq;

namespace JsonTool.Core.Models;

public class SchemaTreeBuilder
{
    public SchemaProperty BuildTree(string jsonSchema)
    {
        var schema = JObject.Parse(jsonSchema);

        var root = new SchemaProperty
        {
            Name = schema["title"]?.ToString() ?? "root",
            Type = schema["type"]?.ToString() ?? "object",
            Description = schema["description"]?.ToString(),
            Depth = 0
        };

        var required = (schema["required"] as JArray)?
            .Select(r => r.ToString())
            .ToHashSet() ?? new HashSet<string>();

        ParseProperties(schema, root, required, 0);

        return root;
    }

    private void ParseProperties(JObject schema, SchemaProperty parent,
        HashSet<string> required, int depth)
    {
        var properties = schema["properties"] as JObject;
        if (properties == null) return;

        foreach (var prop in properties.Properties())
        {
            var propDef = prop.Value as JObject ?? new JObject();
            var property = CreateProperty(prop.Name, propDef, depth + 1);
            property.IsRequired = required.Contains(prop.Name);

            parent.Add(property);

            if (propDef["properties"] != null)
            {
                var nestedRequired = (propDef["required"] as JArray)?
                    .Select(r => r.ToString())
                    .ToHashSet() ?? new HashSet<string>();

                ParseProperties(propDef, property, nestedRequired, depth + 1);
            }
        }
    }
}
```

```

        if (property.Type == "array" && propDef["items"] is JObject items)
        {
            if (items["properties"] != null)
            {
                var itemRequired = (items["required"] as JArray)?
                    .Select(r => r.ToString())
                    .ToHashSet() ?? new HashSet<string>();

                var itemsProperty = new SchemaProperty
                {
                    Name = "[items]",
                    Type = items["type"]?.ToString() ?? "object",
                    Description = "Array items"
                };
                parent.Add(itemsProperty);
                ParseProperties(items, itemsProperty, itemRequired, depth + 2);
            }
        }
    }

private SchemaProperty CreateProperty(string name, JObject definition, int depth)
{
    var property = new SchemaProperty
    {
        Name = name,
        Type = definition["type"]?.ToString() ?? "any",
        Format = definition["format"]?.ToString(),
        Pattern = definition["pattern"]?.ToString(),
        Description = definition["description"]?.ToString(),
        Example = definition["example"]?.ToString(),
        DefaultValue = definition["default"]?.ToObject<object>(),
        IsNullable = definition["nullable"]?.Value<bool>() ?? false,
        Depth = depth
    };

    if (definition["minimum"] != null)
        property.Minimum = definition["minimum"]?.Value<decimal>();
    if (definition["maximum"] != null)
        property.Maximum = definition["maximum"]?.Value<decimal>();

    if (definition["minLength"] != null)
        property.MinLength = definition["minLength"]?.Value<int>();
    if (definition["maxLength"] != null)
        property.MaxLength = definition["maxLength"]?.Value<int>();

    if (definition["enum"] is JArray enumArray)
    {
        foreach (var value in enumArray)
        {
            property.EnumValues.Add(value.ToString());
        }
    }
}

return property;
}

```