Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційні систем та технологій

# Лабораторна робота №8

із дисципліни «*Технології розробки програмного забезпечення*»

**Тема: «Патерни проектування»**

Виконав:
Студент групи ІА-31
Клим’юк В.Л.

Перевірив:
Мягкий М.Ю.

Київ-2025

**Тема**: Система запису на прийом до лікаря (strategy, adapter, observer, facade, visitor, client-server).
Система дозволяє пацієнтам шукати та записуватись на прийоми до лікарів, де їм виноситься висновок за результатами прийому. Також наявні функції відміни прийому та додавання відгуків про лікарів.

Репозиторій: https://github.com/StaticReadonly/kpi-trpz-labs-klim

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

**Теоретичні відомості**:

“Composite” використовується для складання об'єктів в деревоподібну структуру для подання ієрархій типу «частина цілого». Даний шаблон дозволяє уніфіковано обробляти як поодинокі об'єкти, так і об'єкти з вкладеністю.

Простим прикладом може служити складання компонентів всередині звичайної форми. Форма може містити дочірні елементи (поля для введення тексту, цифр, написи, малюнки тощо); дочірні елементи можуть в свою чергу містити інші елементи. Наприклад, при виконанні операції розтягування форми необхідно, щоб вся ієрархія розтягнулася відповідним чином. В такому випадку 90 форма розглядається як композитний об'єкт і операція розтягування застосовується до всіх дочірніх елементів рекурсивно.

Даний шаблон зручно використовувати при необхідності подання та обробки ієрархій об'єктів. Крім того, патерн «Composite» (Компонувальник) краще використовувати, коли ви представляєте структуру даних у вигляді дерева.

Переваги та недоліки:

+ Спрощує представлення деревоподібної структури.

+ Додає гнучкості в роботі з складними об’єктами та рекурсивними операціями.

+ Дозволяє додавати та видаляти об’єкти в ієрархії без впливу на клієнтський код.

- Потрібні додаткові зусилля для початкового впровадження.

- Вимагає гарно спроєктованого загального інтерфейсу.

“Flyweight” використовується для зменшення кількості об'єктів в додатку шляхом поділу цих об'єктів між ділянками додатку. Flyweight являє собою поділюваний об'єкт. Дуже важливою є концепція «внутрішнього» і «зовнішнього» станів. Внутрішній стан відображає дані, характерні саме поділюваному об'єкту (наприклад, код букви); зовнішній стан несе інформацію про його застосування в додатку (наприклад, рядок і стовпчик). Внутрішній стан зберігається в самому поділюваному об'єкті, зовнішній – в об'єктах додатку (контексту використання поділюваного об'єкта).

Даний шаблон дуже добре застосовувати у випадках, коли використовується безліч однакових об'єктів (наприклад, графічних примітивів).

Переваги та недоліки:

+ Заощаджує оперативну пам’ять.

- Витрачає процесорний час на пошук/обчислення контексту.

- Ускладнює код програми внаслідок введення безлічі додаткових класів.

“Interpreter” використовується для подання граматики і інтерпретатора для вибраної мови (наприклад, скриптової). Граматика мови представлена термінальними і нетермінальними символами, кожен з яких інтерпретується в контексті використання.

Клієнт передає контекст і сформовану пропозицію в використовувану мову в термінах абстрактного синтаксичного дерева (деревоподібна структура, яка однозначно визначає ієрархію виклику підвиразів), кожен вираз інтерпретується окремо з використанням контексту. У разі наявності дочірніх виразів, батьківський вираз інтерпретує спочатку дочірні (рекурсивно), а потім обчислює результат власної операції.

Шаблон зручно використовувати в разі невеликої граматики (інакше розростеться кількість використовуваних класів) і відносно простого контексту (без взаємних залежностей і т.п.). 94 Даний шаблон визначає базовий каркас інтерпретатора, який за допомогою рекурсії повертає результат обчислення пропозиції на основі результатів окремих елементів.

Переваги та недоліки:

+ Граматику стає легко розширювати та змінювати, реалізації класів, що описують вузли абстрактного синтаксичного дерева схожі (легко кодуються).

+ Можна легко змінювати спосіб обчислення виразів.

- Супроводження граматики з великою кількістю правил є проблематичним.

“Visitor” дозволяє вказувати операції над елементами без зміни структури конкретних елементів. Таким чином вкрай зручно додавати нові операції, проте дуже важко додавати нові елементи в ієрархію (необхідно додавати відповідні методи для обробки їх відвідувань в кожного відвідувача). Даний шаблон дозволяє групувати однотипні операції, що застосовуються над різнотипними об'єктами.

**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.

**Хід роботи**:

**Діаграма класів:**

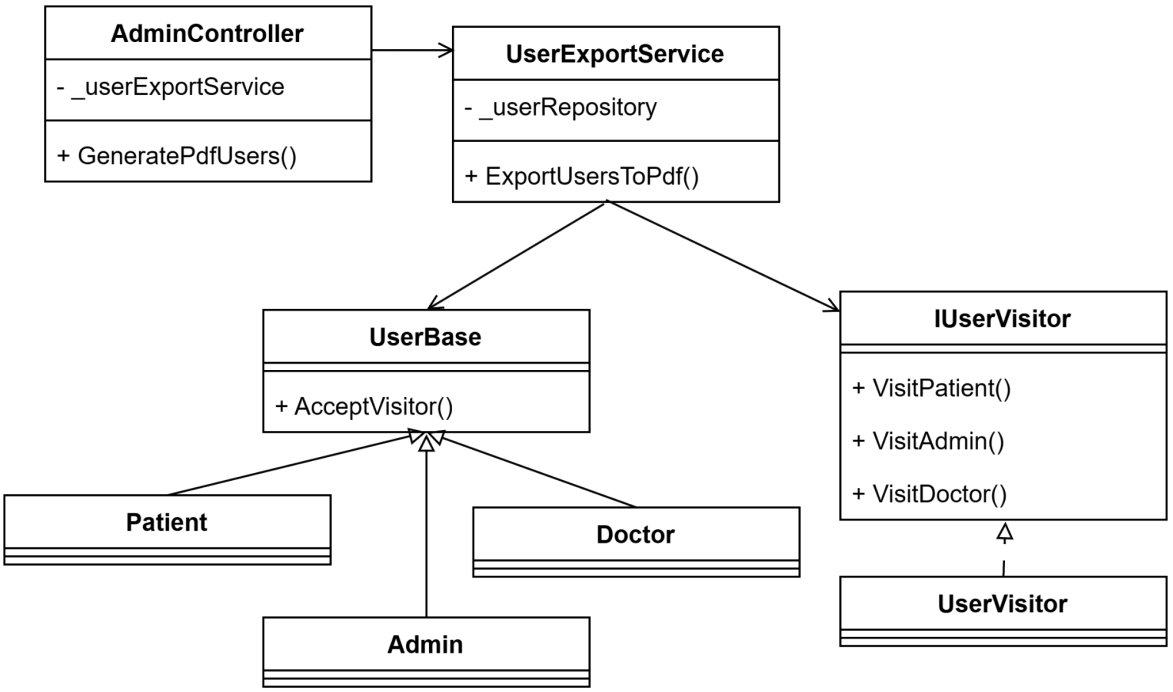



Рис.1 - Діаграма класів шаблону “Відвідувач”

На цій діаграмі:
AdminController - контролер для обробки запитів від адміністратора.
UserExportService - клас для експорту даних користувачів. Використовує дані користувачів та клас-відвідувач для генерації документу з даними.
UserBase - базовий клас для класів даних користувачів.
Patient, Doctor, Admin - класи даних користувачів: пацієнта, доктора, адміністратора. Мають метод AcceptVisitor() для прийняття відвідувача.
IUserVisitor - Інтерфейс відвідувача. Має методи VisitPatient(), VistiAdmin(), VisitDoctor() для відвідування користувачів та виконання певних дій.
UserVisitor - Реалізація інтерфейсу відвідувача, яка проходить користувачів та генерує документ в форматі PDF.

**Вихідний код класів:**

```
using BookingClinic.Services.Visitor;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;

namespace BookingClinic.Controllers
{
    [Route("manage")]
    [Authorize("Admin")]
    Ссылок: 1
    public class AdminController : Controller
    {
        private readonly UserExportService _userExportService;

        Ссылок: 0
        public AdminController(UserExportService userExportService)
        {
            _userExportService = userExportService;
        }

        [HttpGet]
        Ссылок: 0
        public IActionResult Index()
        {
            return View();
        }

        [HttpPost]
        Ссылок: 0
        public IActionResult GeneratePdfUsers()
        {
            _userExportService.ExportUsersToPdf();
            return RedirectToAction("Index");
        }
    }
}
```

Рис.2 - Код класу AdminController

```
1   using BookingClinic.Data.Repositories.UserRepository;
2
3   namespace BookingClinic.Services.Visitor
4   {
        Ссылок: 4
5       public class UserExportService
6       {
7           private readonly IUserRepository _userRepository;
8           private readonly IVisitorFactory _visitorFactory;
9
10
            Ссылок: 0
11          public UserExportService(
12              IUserRepository userRepository,
13              IVisitorFactory visitorFactory)
14          {
15              this._userRepository = userRepository;
16              this._visitorFactory = visitorFactory;
17          }
18
            Ссылок: 1
19          public void ExportUsersToPdf()
20          {
21              string path = Path.Combine(Environment.GetFolderPath(
22                  Environment.SpecialFolder.MyDocuments),
23                  "BookingClinicReports",
24                  $"rep{DateTime.UtcNow:dd-MM-yyyy_HH-mm-ss}.pdf");
25
26
27              using IUserVisitor visitor = _visitorFactory.CreatePDFVisitor(path);
28              var patients = _userRepository.GetVisitorPatients();
29              var doctors = _userRepository.GetVisitorDoctors();
30              var admins = _userRepository.GetVisitorAdmins();
31              var users = patients.Concat(doctors).Concat(admins);
32
33              foreach (var item in users)
34              {
35                  item.AcceptVisitor(visitor);
36              }
37          }
38      }
39  }
```

Рис.3 - Код класу UserExportService

```
using BookingClinic.Data.Entities;

namespace BookingClinic.Services.Visitor
{
    Ссылок: 8
    public interface IUserVisitor : IDisposable
    {
        Ссылок: 2
        void VisitPatient(Patient patient);
        Ссылок: 2
        void VisitAdmin(Admin admin);
        Ссылок: 2
        void VisitDoctor(BookingClinic.Data.Entities.Doctor doctor);
    }
}
```

Рис.4 - Код інтерфейсу IUserVisitor

```
using BookingClinic.Data.Entities;
using QuestPDF.Fluent;
using QuestPDF.Helpers;

namespace BookingClinic.Services.Visitor
{
    Ссылок: 3
    public class UserVisitor : IUserVisitor
    {
        private readonly string _fileName;
        private readonly List<string> _lines = new();
        private bool _disposed;

        Ссылок: 1
        public UserVisitor(string fileName)
        {
            _fileName = fileName;
        }

        Ссылок: 2
        public void VisitPatient(Patient patient)
        {
            if (patient == null) return;

            _lines.Add($"Patient: {patient.Name} {patient.Surname} | Email: {patient.Email} | Phone: {patient.Phone}");

            if (patient.ClientAppointments?.Any() == true)
            {
                _lines.Add($"  Total Appointments: {patient.ClientAppointments.Count}");
                _lines.Add($"  Finished: {patient.ClientAppointments.Count(a => a.IsFinished)}");
                _lines.Add($"  Canceled: {patient.ClientAppointments.Count(a => a.IsCanceled)}");
            }
        }

        Ссылок: 2
        public void VisitAdmin(Admin admin)
        {
            if (admin == null) return;
            _lines.Add($"Admin: {admin.Name} {admin.Surname}");
        }

        Ссылок: 2
        public void VisitDoctor(BookingClinic.Data.Entities.Doctor doctor)
        {
            if (doctor == null) return;

            var speciality = doctor.Speciality?.Name ?? "—";
```

Рис.5 - Код класу UserVisitor

```
42                  var speciality = doctor.Speciality?.Name ?? "—";
43                  var clinic = doctor.Clinic?.Name ?? "—";
44
45
46                  _lines.Add($"Doctor: {doctor.Name} {doctor.Surname} | Speciality: {speciality} | Clinic: {clinic} | Email: {doctor.Email}");
47
48                  if (doctor.DoctorAppointments?.Any() == true)
49                  {
50                      _lines.Add($"  Total Appointments: {doctor.DoctorAppointments.Count}");
51                      _lines.Add($"  Finished: {doctor.DoctorAppointments.Count(a => a.IsFinished)}");
52                      _lines.Add($"  Canceled: {doctor.DoctorAppointments.Count(a => a.IsCanceled)}");
53                  }
54              }
55
                Ссылок: 1
56              private void CreatePdf()
57              {
58                  var directory = Path.GetDirectoryName(_fileName);
59                  if (!string.IsNullOrWhiteSpace(directory) && !Directory.Exists(directory))
60                      Directory.CreateDirectory(directory);
61
62                  if (!_lines.Any())
63                  {
64                      File.WriteAllText(_fileName, "Export produced no entries.");
65                      return;
66                  }
67
68                  Document.Create(container =>
69                  {
70                      container.Page(page =>
71                      {
72                          page.Size(PageSizes.A4);
73                          page.Margin(25);
74                          page.DefaultTextStyle(x => x.FontSize(12));
75                          page.Content().Column(column =>
76                          {
77                              column.Item().Text($"Users export — {DateTime.UtcNow:yyyy-MM-dd HH:mm} UTC").FontSize(14).Bold();
78                              column.Item().LineHorizontal(1).LineColor(Colors.Grey.Lighten2);
79
80                              foreach (var line in _lines)
81                              {
82                                  column.Item().PaddingTop(6).Text(line);
83                              }
84                          });
85                      });
86                  })
```

Рис. 6 - Код класу UserVisitor

```
75                          page.Content().Column(column =>
76                          {
77                              column.Item().Text($"Users export — {DateTime.UtcNow:yyyy-MM-dd HH:mm} UTC").FontSize(14).Bold();
78                              column.Item().LineHorizontal(1).LineColor(Colors.Grey.Lighten2);
79
80                              foreach (var line in _lines)
81                              {
82                                  column.Item().PaddingTop(6).Text(line);
83                              }
84                          });
85                      });
86                  })
87                  .GeneratePdf(_fileName);
88              }
89
                Ссылок: 2
90              protected virtual void Dispose(bool disposing)
91              {
92                  if (_disposed)
93                      return;
94
95                  if (disposing)
96                  {
97                      CreatePdf();
98                  }
99
100                 _disposed = true;
101             }
102
                Ссылок: 0
103             public void Dispose()
104             {
105                 Dispose(true);
106                 GC.SuppressFinalize(this);
107             }
108
                Ссылок: 0
109             ~UserVisitor()
110             {
111                 Dispose(false);
112             }
113         }
114     }
115
```

Рис.7 - Код класу UserVisitor

```
using BookingClinic.Services.Visitor;
using System.ComponentModel.DataAnnotations;

namespace BookingClinic.Data.Entities
{
    Ссылок: 22
    public class UserBase
    {
        [Key]
        Ссылок: 5
        public Guid Id { get; set; }
        Ссылок: 10
        public string Name { get; set; }
        Ссылок: 9
        public string Surname { get; set; }
        Ссылок: 7
        public string Email { get; set; }
        Ссылок: 4
        public string? Phone { get; set; }
        Ссылок: 3
        public string? ProfilePicture { get; set; }
        Ссылок: 4
        public string PasswordHash { get; set; }
        Ссылок: 2
        public string Role { get; set; }
        Ссылок: 7
        public ICollection<Appointment> ClientAppointments { get; set; }
        Ссылок: 3
        public ICollection<DoctorReview> ClientReviews { get; set; }

        Ссылок: 4
        public virtual void AcceptVisitor(IUserVisitor visitor)
        {

        }
    }
}
```

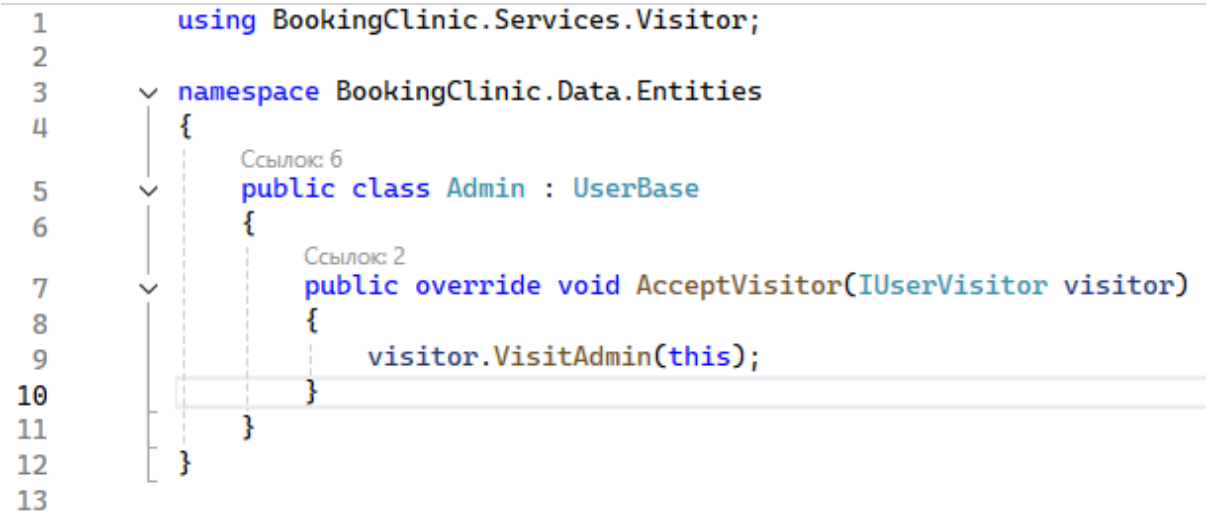
Рис.8 - Код класу UserBase

```
1   using BookingClinic.Services.Visitor;
2
3   namespace BookingClinic.Data.Entities
4   {
        Ссылок: 9
5       public class Patient : UserBase
6       {
            Ссылок: 2
7           public override void AcceptVisitor(IUserVisitor visitor)
8           {
9               visitor.VisitPatient(this);
10          }
11      }
12  }
13
```

Рис.9 - Код класу Patient

```
1   using BookingClinic.Services.Visitor;
2
3   namespace BookingClinic.Data.Entities
4   {
        Ссылок: 24
5       public class Doctor : UserBase
6       {
            Ссылок: 4
7           public ICollection<DoctorReview> DoctorReviews { get; set; }
            Ссылок: 8
8           public ICollection<Appointment> DoctorAppointments { get; set; }
            Ссылок: 2
9           public Guid SpecialityId { get; set; }
            Ссылок: 7
10          public Speciality Speciality { get; set; }
            Ссылок: 1
11          public Guid ClinicId { get; set; }
            Ссылок: 13
12          public Clinic Clinic { get; set; }
13
            Ссылок: 2
14          public override void AcceptVisitor(IUserVisitor visitor)
15          {
16              visitor.VisitDoctor(this);
17          }
18      }
19  }
20
```

Рис.10 - Код класу Doctor



```
using BookingClinic.Services.Visitor;

namespace BookingClinic.Data.Entities
{
    Ссылок: 6
    public class Admin : UserBase
    {
        Ссылок: 2
        public override void AcceptVisitor(IUserVisitor visitor)
        {
            visitor.VisitAdmin(this);
        }
    }
}
```

Рис.11 - Код класу Admin

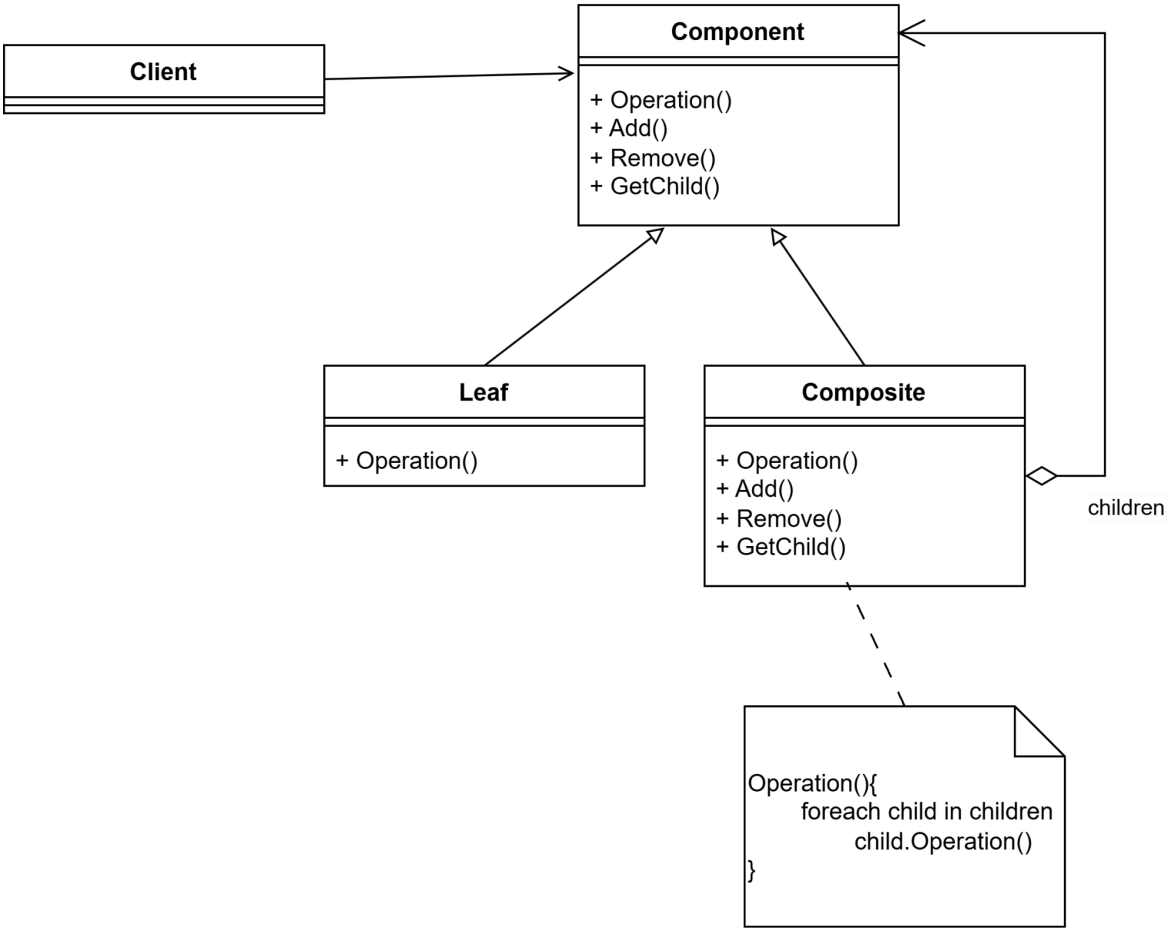
**Висновки:** У цій лабораторній роботі я ознайомився з новими шаблонами проектування. “Композит” дозволяє будувати структури складних та простих об’єктів із загальним інтерфейсом, “Пристосуванець” дозволяє виділити спільний стан для декількох об’єктів економлячи пам’ять, “Інтерпретатор” дозволяє інтерпретацію деревовидної мови з контекстом, “Відвідувач” дозволяє обходити структури класів та визначати різні операції над ними. Я обрав та реалізував шаблон “Відвідувач” на прикладі обходу даних користувачів та занесення в PDF-документ. Вивчене покращило моє розуміння патернів проектування та знадобиться в подальшій розробці.

**Контрольні питання**:

1. Яке призначення шаблону «Композит»?

Дає можливість створювати деревовидні структури класів для подання ієрархій типу “частина цілого”. Дозволяє уніфіковано обробляти обробляти поодинокі об’єкти та об’єкти з вкладеністю. Якщо у нас є умовна коробка та продукти що можуть в ній лежати, то можна ввести для них загальний інтерфейс з методом, який наприклад обраховує загальну вагу. Тоді звичайний продукт повертатиме свою вагу, а коробка опитає кожний об’єкт (інші коробки та продукти всередині) з цим самим інтерфейсом і поверне суму ваг.

2. Нарисуйте структуру шаблону «Композит».




3. Які класи входять в шаблон «Композит», та яка між ними взаємодія?

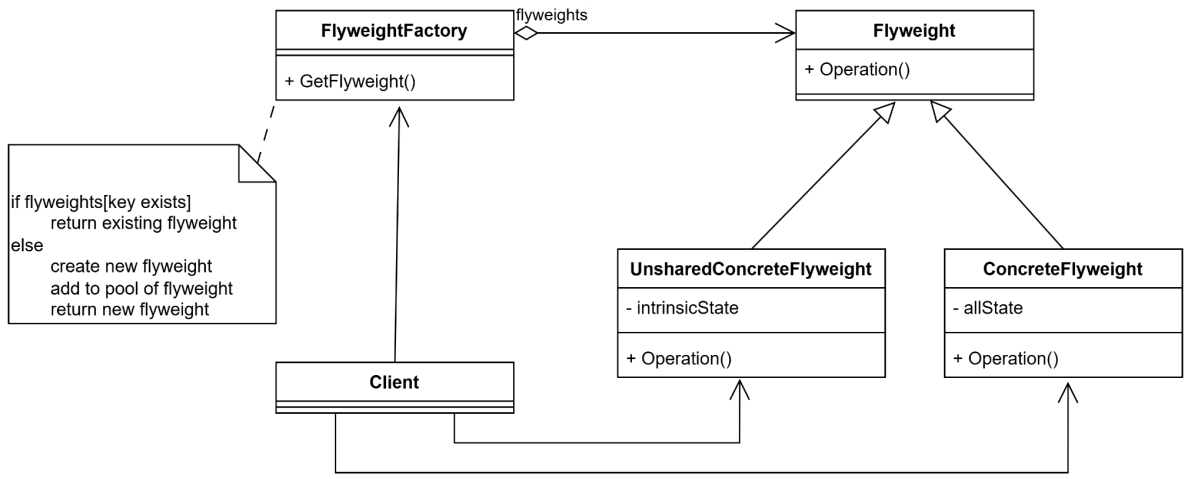
Client - клієнтський клас, який використовує клас-композит.
Component - Базовий клас-композит, який описує операції, що властиві окремому простому або комплексному елементу дерева.
Leaf - Клас простого предмета. Не делегує виконання далі і виконує реальну операцію.
Composite - Клас комплексного нащадка. Делегує виконання операції іншим компонентам.

4. Яке призначення шаблону «Легковаговик»?

Призначений для зменшення кількості об’єктів та економії пам’яті. Якщо у декількох об’єктів є певний зовнішній стан, який несе інформацію про

застосування в додатку, то є сенс розділити його між усіма як окремий об’єкт.

5. Нарисуйте структуру шаблону «Легковаговик».




6. Які класи входять в шаблон «Легковаговик», та яка між ними взаємодія?

Client - клієнтський клас, який використовує фабрику легковаговиків для їх створення та самих легковаговиків.
FlyweightFactory - фабрика для створення легковаговиків з поширеним зовнішнім та непоширеним внутрішнім станом.
Flyweight - базовий клас легковаговика.
UnsharedConcreteFlyweight - клас легковаговика з непоширеним внутрішнім станом.
ConcreteFlyweight - клас легковаговика з поширеним зовнішнім станом.

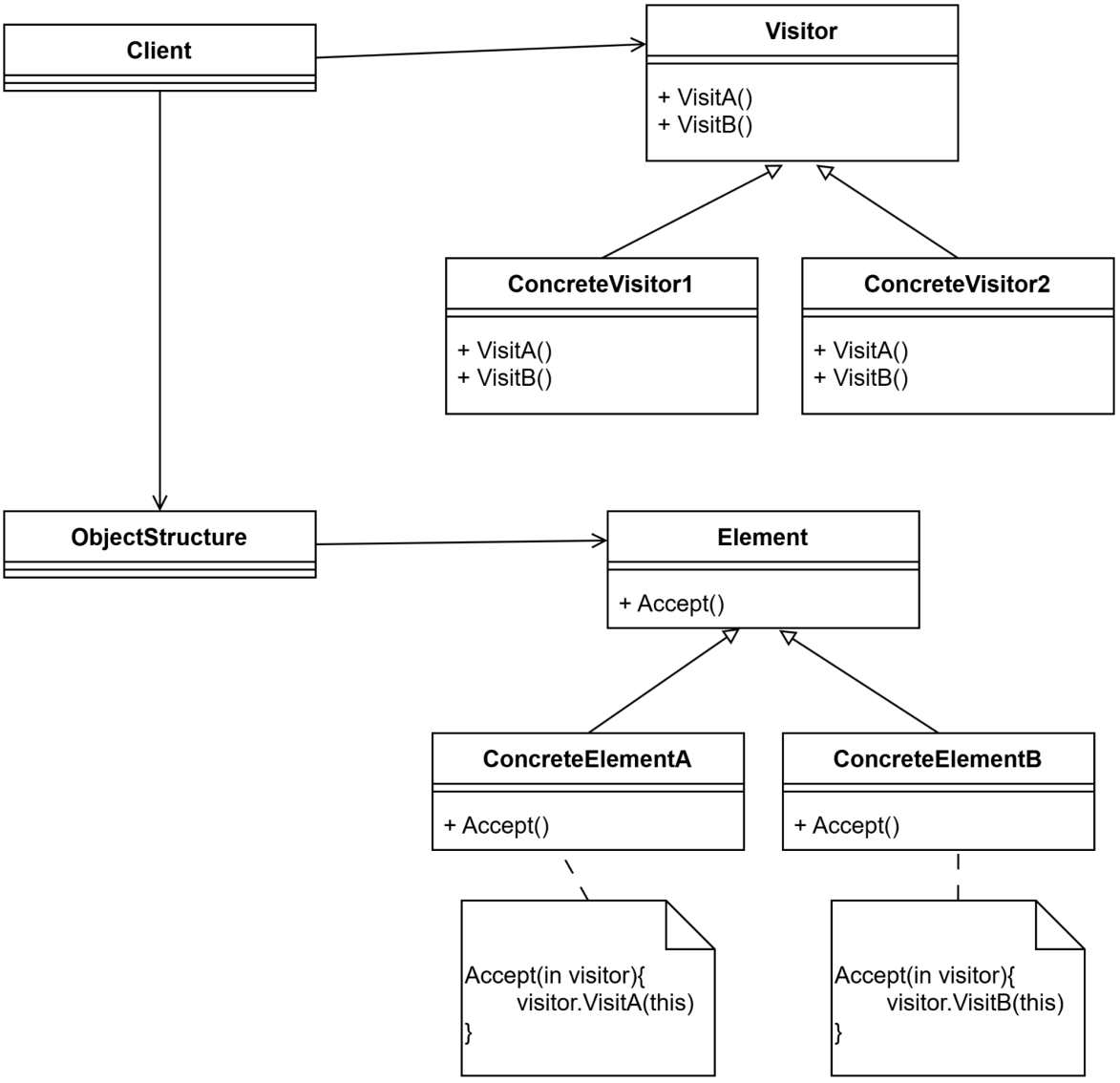
7. Яке призначення шаблону «Інтерпретатор»?

Використовується для подання граматики і інтерпретатора для вибраної мови (наприклад, скриптової), що включає термінальні та нетермінальні символи. Користувач передає контекст і вираз з деревоподібною структурою, яку рекурсивно обробляє інтерпретатор з урахуванням контексту.

8. Яке призначення шаблону «Відвідувач»?

Відвідувач дозволяє вказувати операції над елементами без зміни структури конкретних елементів. Кожен клас-відвідувач може проходити структуру об’єктів і виконувати над ними свої певні операції.

9. Нарисуйте структуру шаблону «Відвідувач».




10. Які класи входять в шаблон «Відвідувач», та яка між ними взаємодія?

Client - клієнтський клас, що використовує структуру об’єктів та відвідувача.
ObjectStructure - клас, що відображає певну структуру об’єктів.

Visitor - базовий клас відвідувача, якого будуть приймати елементи структури.
ConcreteVisitor1,2 - конкретні реалізації відвідувачів.
Element - об’єкт структури, який може прийняти в себе відвідувача для виконання певних дій.
ConcreteElementA,B - конкретні реалізації об’єктів структури.