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 № 7**

із дисципліни «Технології розроблення програмного забезпечення»

Тема: «Патерни проектування»

Варіант - 17

Виконав
Студент групи ІА-31:
Лисенко В. Є

Перевірив:
Мягкий М. Ю.

Київ 2025

# Зміст

## 1. Мета:

Вивчити структуру шаблонів «Mediator», «Facade», «Bridge», «Template method» та навчитися застосовувати їх в реалізації програмної системи.

## 2. Теоретичні відомості:

Mediator (Посередник): Шаблон для централізації взаємодії між об'єктами через посередника, замість прямих зв'язків. Кожен об'єкт зберігає лише посилання на медіатор. Застосовується в складних формах з великою кількістю взаємодіючих компонентів (наприклад, чекбокси, блоки, що ховаються/показуються). Переваги: Зменшення зв'язаності, простота розширення, легке тестування. Недоліки: Медіатор може стати «God Object».

Facade (Фасад): Надає спрощений уніфікований інтерфейс до складної підсистеми, приховуючи її внутрішню структуру. Використовується для роботи з різними протоколами (HTTP, TCP) або складними бібліотеками. Переваги: Інкапсуляція складності, простіший API, легке оновлення внутрішньої реалізації. Недоліки: Зменшення гнучкості.

Bridge (Міст): Розділяє абстракцію та її реалізацію, дозволяючи змінювати їх незалежно. Утворює дві ієрархії. Застосовується в графічних редакторах (фігури + драйвери: екран, принтер, bitmap). Переваги: Незалежний розвиток абстракції та реалізації, гнучкість. Недоліки: Збільшення складності.

Template Method (Шаблонний метод): Визначає скелет алгоритму в базовому класі, залишаючи окремі кроки для перевизначення в похідних. Використовується для обробки різних форматів відео (MPEG-4, MPEG-2) або компіляції веб-сторінок. Переваги: Повторне використання коду, чітка структура алгоритму. Недоліки: Жорсткий скелет, можливе порушення принципу Лісков, складність при багатьох кроках.

## 3. Завдання:

1) Ознайомитись з короткими теоретичними відомостями.
2) Реалізувати частину функціоналу робочої програми у вигляді класів та їхньої взаємодії для досягнення конкретних функціональних можливостей.
3) Реалізувати один з розглянутих шаблонів за обраною темою.
4) Реалізувати не менше 3-х класів відповідно до обраної теми.
5) 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.

**4. Хід роботи:**
Мій варіант:

17. **System activity monitor** (iterator, command, abstract factory, bridge, visitor,

SOA) Монітор активності системи повинен зберігати і запам'ятовувати статистику використовуваних компонентів системи, включаючи навантаження на процесор, обсяг займаної оперативної пам'яті, натискання клавіш на клавіатурі, дії миші (переміщення, натискання), відкриття вікон і зміна вікон; будувати звіти про використання комп'ютера за різними критеріями (% часу перебування у веб-браузері, середнє навантаження на процесор по годинах, середній час роботи комп'ютера по днях і т.д.); правильно поводитися з «простоюванням» системи – відсутністю користувача.

Паттерн Bridge дозволяє:

1. Розділити абстракцію (логіку підготовки даних звіту) та реалізацію (логіку рендерингу в конкретний формат).
2. Змінювати формат звіту динамічно під час виконання програми (через ComboBox на UI), не змінюючи клас самого звіту.
3. Зменшити зв'язність коду: клас DailyReport займається лише даними, а клас HtmlRenderer — лише HTML-тегами.

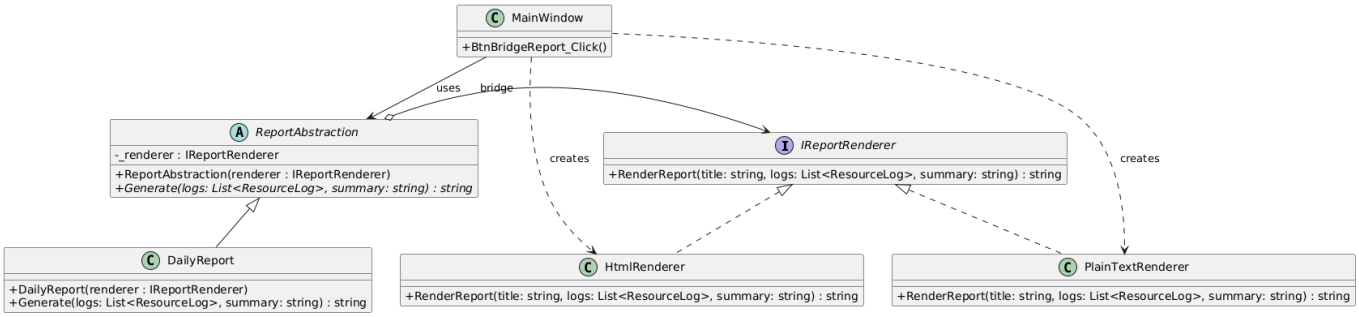



Рисунок 1 - Структура патерну Bridge

| | | |
|---|---|---|
| IReportRenderer | Implementor | Інтерфейс, що визначає низькорівневі операції рендерингу (методи RenderReport). Він не прив'язаний до бізнес-логіки звіту. |
| HtmlRenderer / PlainTextRenderer | Concrete Implementors | Конкретні реалізації інтерфейсу. Перший формує рядок з тегами <html><table>, другий — звичайний текст з розділювачами. |
| ReportAbstraction | Abstraction | Базовий клас звіту. Він зберігає посилання на об'єкт _renderer (Міст) і делегує йому роботу з візуалізації. |

| | | |
|---|---|---|
| DailyReport | Refined Abstraction | Уточнена абстракція. Це клас конкретного типу звіту (Щоденний), який може додавати специфічну бізнес-логіку (наприклад, заголовок, фільтрацію даних), але для фінального виводу використовує метод _renderer.RenderReport(). |
| MainWindow | Клієнт | Зв'язує абстракцію з конкретною реалізацією на основі вибору користувача (через ComboBox) під час виконання програми. |

**Програмна реалізація:**

IReportRenderer.cs:

```
using System.Collections.Generic;
using SystemActivityMonitor.Data.Entities;

namespace SystemActivityMonitor.Data.Patterns.Bridge
{
    public interface IReportRenderer
    {
        string RenderReport(string title, List<ResourceLog> logs, string summary = "");
    }
}
```

ReportAbstraction.cs:

```
using System.Collections.Generic;
using SystemActivityMonitor.Data.Entities;

namespace SystemActivityMonitor.Data.Patterns.Bridge
{
    public abstract class ReportAbstraction
    {
        protected IReportRenderer _renderer;

        public ReportAbstraction(IReportRenderer renderer)
        {
            _renderer = renderer;
        }

        public abstract string Generate(List<ResourceLog> logs, string summary = "");
    }
}
```

## DailyReport.cs:

```
using System.Collections.Generic;

using SystemActivityMonitor.Data.Entities;


namespace SystemActivityMonitor.Data.Patterns.Bridge

{

    public class DailyReport : ReportAbstraction

    {

        public DailyReport(IReportRenderer renderer) : base(renderer)

        {

        }


        public override string Generate(List<ResourceLog> logs, string summary = "")

        {

            return _renderer.RenderReport("Щоденний Звіт Системи", logs, summary);

        }

    }

}
```

## MainWindow.xaml.cs:

```
private void BtnBridgeReport_Click(object sender, RoutedEventArgs e)

        {

            List<ResourceLog> logs;

            using (var db = new MonitorDbContext())

            {

                logs = db.ResourceLogs.OrderByDescending(l => l.CreatedAt).Take(10).ToList();

            }


            var elements = new List<IMetricElement>();

            foreach (var log in logs)

            {

                elements.Add(new CpuMetric(log.CpuLoad, log.CreatedAt.ToShortTimeString(), log.ActiveWindow,
log.IsSystemIdle));

                elements.Add(new RamMetric(log.RamUsage));

            }


            var analyzer = new AnalysisVisitor();

            foreach (var el in elements) el.Accept(analyzer);


            string analyticsResult = analyzer.GetStats();


            IReportRenderer renderer = cmbReportFormat.SelectedIndex == 0

                ? new PlainTextRenderer()
```

```
        : new HtmlRenderer();

    ReportAbstraction report = new DailyReport(renderer);
    string finalOutput = report.Generate(logs, analyticsResult);

    MessageBox.Show(finalOutput, "Повний інтегрований звіт");
}
```

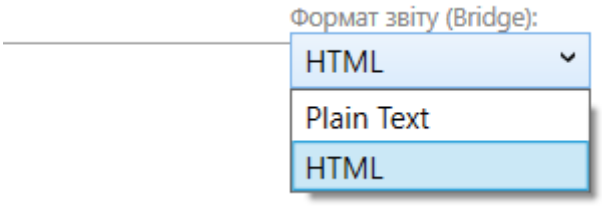



Рисунок 2 – Обрання формату звіту

```
<html><body>
<h1>Щоденний Звіт Системи</h1>
<div style='border: 2px solid #007ACC; padding: 10px; background-color: #e6f7ff; margin-bottom: 20px;'>
<h3> Аналітика (Visitor)</h3>
<p>Аналіз завершено:<br> - Середнє CPU: 94,70%<br> - Час у браузері: 60,00%<br> - Час простою (Idle): 0,00%<br> - Мінімум RAM: 294 MB</p>
</div>
<table border='1' cellpadding='5' cellspacing='0'>
<tr style='background-color: #f2f2f2;'><th>Time</th><th>CPU Load</th><th>RAM Usage</th></tr>
<tr>
<td>13:39</td>
<td style='color:red; font-weight:bold;'>96%</td>
<td>352 MB</td>
</tr>
<tr>
<td>13:39</td>
<td style='color:red; font-weight:bold;'>96%</td>
<td>294 MB</td>
</tr>
<tr>
<td>13:39</td>
<td style='color:red; font-weight:bold;'>96%</td>
<td>459 MB</td>
</tr>
<tr>
<td>13:39</td>
<td style='color:red; font-weight:bold;'>92%</td>
<td>496 MB</td>
</tr>
<tr>
<td>13:39</td>
<td style='color:red; font-weight:bold;'>93%</td>
<td>368 MB</td>
</tr>
<tr>
<td>13:39</td>
<td style='color:red; font-weight:bold;'>99%</td>
<td>489 MB</td>
</tr>
<tr>
<td>13:39</td>
<td style='color:red; font-weight:bold;'>92%</td>
<td>310 MB</td>
</tr>
<tr>
<td>13:39</td>
<td style='color:red; font-weight:bold;'>99%</td>
<td>339 MB</td>
</tr>
<tr>
<td>13:39</td>
```

Рисунок 3 – Результат роботи HTML-рендерера, підключеного через патерн Bridge.

## 5. Висновок

У ході виконання лабораторної роботи було досліджено та імплементовано структурний патерн проєктування Bridge (Міст) . Цей підхід було використано для вирішення задачі експорту звітів моніторингу в різні формати (Текст, HTML).

Реалізація патерну дозволила розділити систему на дві незалежні ієрархії:

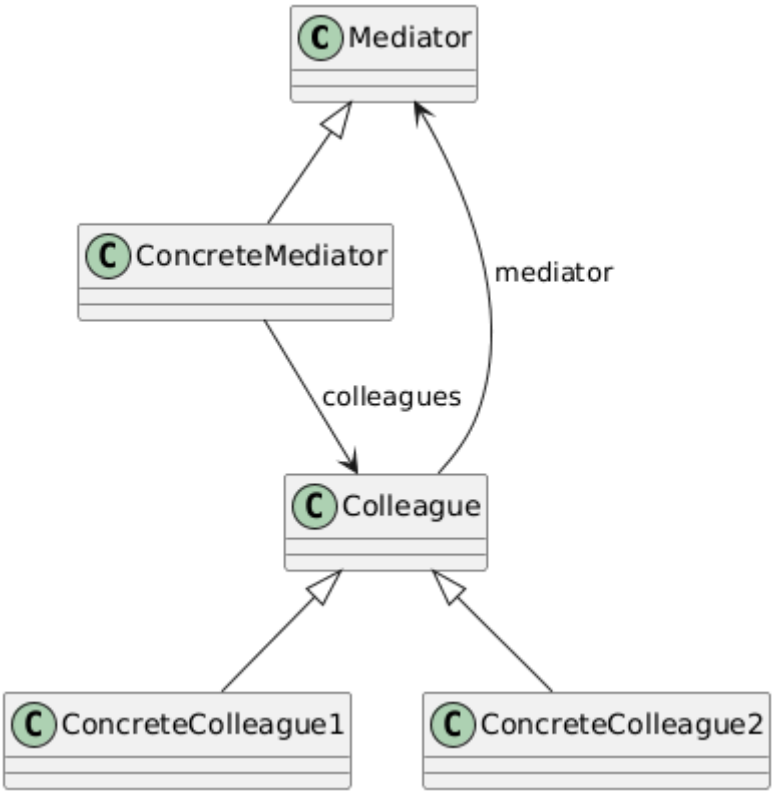
1. Абстракція: Ієрархія типів звітів (базовий клас ReportAbstraction та конкретний DailyReport), яка відповідає за бізнес-логіку підготовки даних.
2. Реалізація: Ієрархія рендерерів (інтерфейс IReportRenderer та класи HtmlRenderer, PlainTextRenderer), яка відповідає виключно за формат представлення даних.

## 6. Контрольні питання:

1. Яке призначення шаблону «Посередник»?

Централізує взаємодію між об’єктами через єдиний об’єкт-посередник, усуваючи прямі зв’язки між ними. Зменшує зв’язність, спрощує логіку взаємодії.

2. Нарисуйте структуру шаблону «Посередник».




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

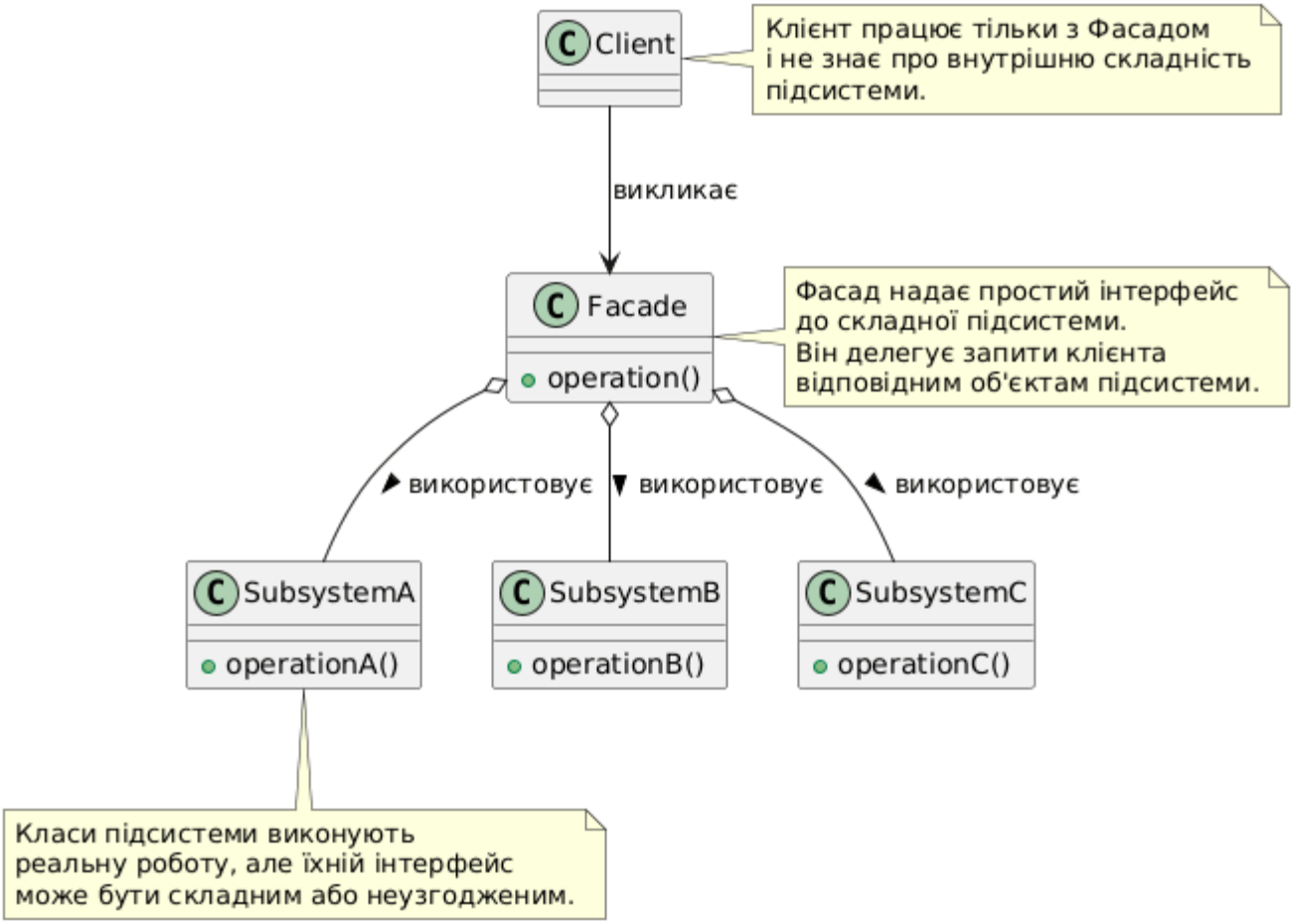
- Mediator — інтерфейс посередника

- ConcreteMediator — реалізація, координує взаємодію
- Colleague — базовий клас компонентів
- ConcreteColleague1, ConcreteColleague2 — конкретні компоненти

4. Яке призначення шаблону «Фасад»?

Надає спрощений уніфікований інтерфейс до складної підсистеми, приховуючи її внутрішню складність.

5. Нарисуйте структуру шаблону «Фасад».



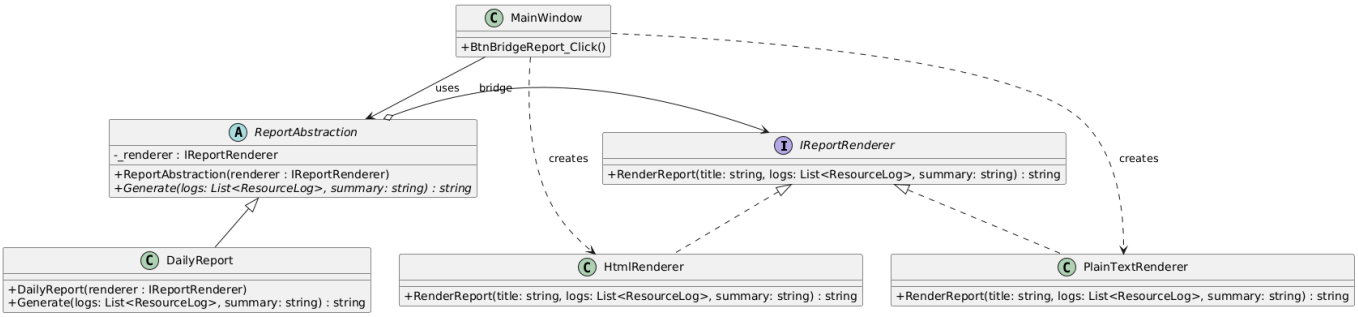

6. Які класи входять в шаблон «Фасад», та яка між ними взаємодія?

- Facade — єдиний інтерфейс
- SubsystemA, SubsystemB, SubsystemC — класи підсистеми

7. Яке призначення шаблону «Міст»?

Розділяє абстракцію та її реалізацію, дозволяючи змінювати їх незалежно (дві ієрархії).

8. Нарисуйте структуру шаблону «Міст».



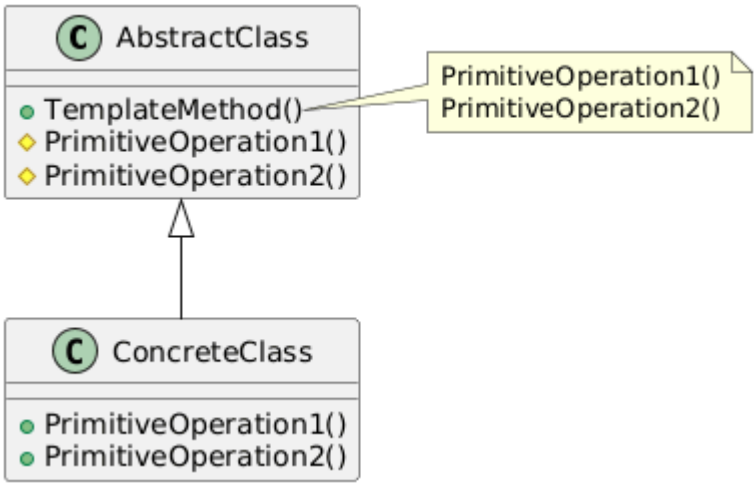

9. Які класи входять в шаблон «Міст», та яка між ними взаємодія?

- Abstraction — абстракція
- RefinedAbstraction — розширена абстракція
- Implementor — інтерфейс реалізації
- ConcreteImplementorA/B — конкретні реалізації

10. Яке призначення шаблону «Шаблонний метод»?

Визначає скелет алгоритму в базовому класі, залишаючи окремі кроки для перевизначення в похідних.

11. Нарисуйте структуру шаблону «Шаблонний метод».




12. Які класи входять в шаблон «Шаблонний метод», та яка між ними взаємодія?

- AbstractClass — містить TemplateMethod() і абстрактні методи
- ConcreteClass — реалізує абстрактні методи

13. Чим відрізняється шаблон «Шаблонний метод» від «Фабричного методу»?

Шаблонний метод» визначає скелет алгоритму, дозволяючи підкласам перевизначати окремі кроки. Фабричний метод» визначає інтерфейс створення об’єкта, залишаючи підкласам вибір конкретного класу.

14. Яку функціональність додає шаблон «Міст»?

Дозволяє незалежно розвивати абстракцію та реалізацію.

- Додає гнучкість
- Усуває експоненційне зростання підкласів
- Підтримує принцип розділення відповідальностей