Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота № 4

з дисципліни «Технології розроблення програмного забезпечення»

Тема: «Вступ до паттернів проектування»

Виконала:
студентка групи ІА-31
Кизим Є. К.

Київ 2025

Зміст

Мета: Вивчити структуру шаблонів «Singleton», «Iterator», «Proxy», «State», «Strategy» 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.

Моя тема: 27. Особиста бухгалтерія (state, prototype, decorator, bridge, flyweight, SOA) Програма повинна бути наочним засобом для ведення особистих фінансів: витрат і прибутку; з можливістю встановлення періодичних витрат / прибутку (зарплата і орендна плата); введення сканованих чеків з відповідними статтями витрат; побудова статистики; експорт/імпорт в Excel, реляційні джерела даних; різні рахунки; ведення єдиного фонду на всі рахунки (всією сім'єю) – на особливі потреби (ремонт, автомобіль, відпустка); можливість введення вкладів / кредитів для контролю банківських рахунків (звірка нарахованих відсотків з необхідними і т.д.).

Теоретичні відомості

1. Поняття шаблону проєктування

Шаблон (патерн) проєктування — це формалізований опис часто зустріжуваної задачі проєктування, вдале рішення цієї задачі, а також рекомендації щодо застосування цього рішення в різних ситуаціях. Сукупність патернів являє собою єдиний словник проєктування, який полегшує спілкування між розробниками та дозволяє використовувати перевірені архітектурні рішення.

Використання шаблонів підвищує стійкість системи до змін вимог та спрощує її подальше доопрацювання.

2. Огляд досліджуваних шаблонів

В рамках лабораторної роботи було розглянуто наступні породжувальні, структурні та поведінкові патерни:

- **Singleton (Одинак)** Цей породжувальний шаблон гарантує, що клас має лише один екземпляр, і надає глобальну точку доступу до нього.
    - Застосування: Використовується, коли в системі має бути рівно один екземпляр деякого класу (наприклад, менеджер налаштувань або підключення до БД) .
    - Особливості: Часто вважається «анти-патерном», оскільки вводить глобальний стан, що ускладнює тестування та підтримку коду.
- **Iterator (Ітератор)** Поведінковий шаблон, що надає спосіб послідовного доступу до елементів колекції без розкриття її внутрішнього представлення.
    - Принцип роботи: Виносить логіку перебору елементів (обходу) із самої колекції в окремий об'єкт-ітератор.
    - Переваги: Дозволяє реалізувати різні алгоритми обходу однієї й тієї ж структури даних, не змінюючи код самої колекції.
- **Proxy (Замісник)** Структурний шаблон, що надає об'єкт-замінник для контролю доступу до іншого об'єкта.
    - Призначення: Використовується для «лінивої» ініціалізації (створення важких об'єктів тільки коли вони потрібні), контролю доступу, логування запитів або кешування результатів.
    - Структура: Клієнт працює з інтерфейсом Subject, який реалізують і реальний об'єкт (RealSubject), і замісник (Proxy).
- **State (Стан)** Поведінковий шаблон, який дозволяє об'єкту змінювати свою поведінку при зміні його внутрішнього стану.

    - Принцип роботи: Поведінка, що залежить від стану, виноситься в окремі класи. Об'єкт-контекст зберігає посилання на поточний об'єкт-стан і делегує йому виконання роботи.
    - Застосування: Доречний, коли об'єкт може перебувати в різних станах (наприклад, замовлення: «нове», «обробляється», «відправлено»), і в кожному стані він має поводитися по-різному.
- **Strategy (Стратегія)** Поведінковий шаблон, який визначає сімейство алгоритмів, інкапсулює кожен з них і робить їх взаємозамінними.
    - Призначення: Дозволяє змінювати алгоритми (стратегії) незалежно від клієнтів, які їх використовують.
    - Структура:
        - **Context (Контекст):** Клас, що використовує алгоритм.
        - **Strategy (Стратегія):** Спільний інтерфейс для всіх алгоритмів.
        - **ConcreteStrategy:** Конкретні реалізації алгоритмів .
    - Переваги: Дозволяє уникнути складних умовних операторів (if/switch) та легко додавати нові варіанти поведінки (наприклад, нові формати експорту даних), не змінюючи основний код програми.

Саме шаблон **Strategy** було обрано для практичної реалізації в даній лабораторній роботі для забезпечення гнучкості механізму експорту звітів.

Хід роботи

Діаграма класів реалізації патерну Strategy

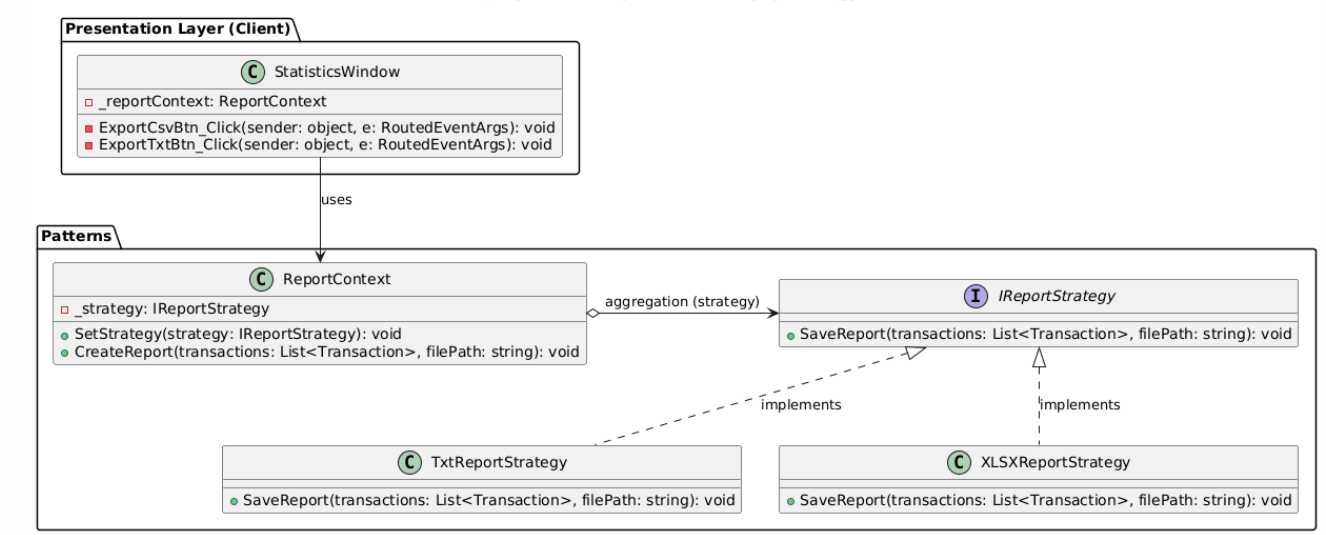


Рисунок 1 – Діаграма класів реалізації патерну Strategy

Опис реалізації патерну «Strategy»:

Для реалізації функціоналу експорту статистичних звітів у системі «Особиста бухгалтерія» було обрано поведінковий шаблон проєктування **Strategy (Стратегія)**. Цей вибір обумовлений необхідністю підтримувати різні формати збереження даних (Excel, Text) та можливістю легко додавати нові формати в майбутньому без зміни основного коду програми.

Реалізація складається з наступних елементів:

1. **Strategy (Інтерфейс Стратегії):** Роль цього елемента виконує інтерфейс **IReportStrategy**. Він визначає єдиний контракт для всіх алгоритмів генерації звітів — метод void SaveReport(...). Це гарантує, що контекст може працювати з будь-якою конкретною стратегією однаково.
2. **Concrete Strategies (Конкретні Стратегії):** Розроблено два класи, що реалізують інтерфейс IReportStrategy та містять специфічну бізнес-логіку:
    - **XLSXReportStrategy**: Реалізує алгоритм експорту даних у формат Microsoft Excel (.xlsx). Для роботи з таблицями використовується

зовнішня бібліотека ClosedXML, що дозволяє формувати професійно оформлені звіти зі стилями та заголовками.
    - **TxtReportStrategy**: Реалізує алгоритм експорту у простий текстовий файл (.txt). Використовує StringBuilder для формування рядкового представлення транзакцій.
3. **Context (Контекст):** Клас **ReportContext** виступає посередником між клієнтським кодом та стратегіями.
    - Він містить посилання на поточну стратегію (_strategy).
    - Метод SetStrategy() дозволяє динамічно змінювати алгоритм експорту під час виконання програми.
    - Метод CreateReport() делегує виконання роботи конкретній стратегії, викликаючи її метод SaveReport().
4. **Client (Клієнт):** Роль клієнта виконує форма **StatisticsWindow**. В залежності від натиснутої користувачем кнопки («Експорт в Excel» або «Експорт в Текст»), клієнт створює екземпляр відповідної стратегії (new XLSXReportStrategy або new TxtReportStrategy) та передає його в Контекст.

Переваги застосованого підходу:

- **Відкритість/Закритість (Open/Closed Principle):** Ми можемо додати нову стратегію (наприклад, PdfReportStrategy), просто створивши новий клас, не змінюючи код ReportContext або інших стратегій.
- **Усунення умовних операторів:** Використання поліморфізму дозволило уникнути громіздких конструкцій if/else або switch при виборі формату збереження.
- **Розділення відповідальності:** Логіка створення файлів повністю відокремлена від логіки інтерфейсу користувача.

Фрагменти програмного коду:

Інтерфейс стратегії (IReportStrategy.cs):

```
using System.Collections.Generic;
using personalAccounting.Models;

namespace personalAccounting.Patterns
{
    public interface IReportStrategy
    {
        void SaveReport(List<Transaction> transactions, string filePath);
    }
}
```

Конкретна стратегія для Excel (XLSXReportStrategy.cs):

```
using System.Collections.Generic;
using personalAccounting.Models;
using ClosedXML.Excel;

namespace personalAccounting.Patterns
{
    public class XLSXReportStrategy : IReportStrategy
    {
        public void SaveReport(List<Transaction> transactions, string filePath)
        {
            var workbook = new XLWorkbook();
            var worksheet = workbook.Worksheets.Add("Звіт");
```

```
        worksheet.Cell(1, 1).Value = "ID";
        worksheet.Cell(1, 2).Value = "Дата";
        worksheet.Cell(1, 3).Value = "Сума";
        worksheet.Cell(1, 4).Value = "Категорія";
        worksheet.Cell(1, 5).Value = "Тип";
        worksheet.Cell(1, 6).Value = "Опис";

        var headerRange = worksheet.Range("A1:F1");
        headerRange.Style.Font.Bold = true;
        headerRange.Style.Fill.BackgroundColor = XLColor.LightGray;

        int row = 2;
        foreach (var t in transactions)
        {
            worksheet.Cell(row, 1).Value = t.TransactionId;
            worksheet.Cell(row, 2).Value = t.Date;
            worksheet.Cell(row, 3).Value = t.Amount;
            worksheet.Cell(row, 4).Value = t.Category;
            worksheet.Cell(row, 5).Value = t.Type.ToString();
            worksheet.Cell(row, 6).Value = t.Description;

            worksheet.Cell(row, 2).Style.DateFormat.Format = "dd.MM.yyyy";
            row++;
        }

        worksheet.Columns().AdjustToContents();
        workbook.SaveAs(filePath);
    }
}
```

```
}
```

Конкретна стратегія для тексту (TxtReportStrategy.cs):

```
using System.Collections.Generic;
using System.IO;
using System.Text;
using personalAccounting.Models;

namespace personalAccounting.Patterns
{
    public class TxtReportStrategy : IReportStrategy
    {
        public void SaveReport(List<Transaction> transactions, string filePath)
        {
            var sb = new StringBuilder();
            sb.AppendLine("ЗВІТ ПРО ВИТРАТИ");
            sb.AppendLine($"Дата генерації: {System.DateTime.Now}");
            sb.AppendLine("-----------------------------------------------");

            decimal total = 0;

            foreach (var t in transactions)
            {
                sb.AppendLine($"{t.Date.ToShortDateString()} | {t.Category}");
                sb.AppendLine($"   Сума: {t.Amount} UAH");
                sb.AppendLine($"   Опис: {t.Description}");
                sb.AppendLine("- - - - - - - - - - - - - - -");

                total += t.Amount;
```

```
            }

            sb.AppendLine("-----------------------------------------------");
            sb.AppendLine($"ВСЬОГО ВИТРАЧЕНО: {total} UAH");

            File.WriteAllText(filePath, sb.ToString(), Encoding.UTF8);
        }
    }
}
```

Клас Контекст (ReportContext.cs):

```
using System.Collections.Generic;
using personalAccounting.Models;

namespace personalAccounting.Patterns
{
    public class ReportContext
    {
        private IReportStrategy _strategy;

        public void SetStrategy(IReportStrategy strategy)
        {
            _strategy = strategy;
        }

        public void CreateReport(List<Transaction> transactions, string filePath)
        {
            if (_strategy == null)
            {
```

```
            _strategy = new TxtReportStrategy();
        }

        _strategy.SaveReport(transactions, filePath);
    }
  }
}
```

Використання в клієнтському коді (StatisticsWindow.xaml.cs):

```
using System.Windows;
using Microsoft.Win32;
using personalAccounting.Patterns;
using personalAccounting.Repositories;
using personalAccounting.Services;

namespace personalAccounting
{
  public partial class StatisticsWindow : Window
  {
    private readonly StatisticsService _statisticsService;
    private readonly TransactionRepository _transactionRepository;
    private readonly ReportContext _reportContext;

    public StatisticsWindow()
    {
      InitializeComponent();
      _statisticsService = new StatisticsService();
      _transactionRepository = new TransactionRepository();
      _reportContext = new ReportContext();
```

```
    LoadData();
}

private void LoadData()
{
    var data = _statisticsService.GetExpensesByCategory();
    if (data.Count == 0) MessageBox.Show("Витрат поки немає.", "Інфо");
    StatsGrid.ItemsSource = data;
}

private void ExportXlsxBtn_Click(object sender, RoutedEventArgs e)
{
    SaveFileDialog saveFileDialog = new SaveFileDialog
    {
        Filter = "Excel XLSX (*.xlsx)|*.xlsx",
        FileName = "Report.xlsx"
    };

    if (saveFileDialog.ShowDialog() == true)
    {
        _reportContext.SetStrategy(new XLSXReportStrategy());

        var transactions = _transactionRepository.GetAll();

        _reportContext.CreateReport(transactions, saveFileDialog.FileName);

        MessageBox.Show("Звіт збережено у форматі Excel (XLSX)!", "Успіх");
    }
}
```

```
    private void ExportTxtBtn_Click(object sender, RoutedEventArgs e)
    {
        SaveFileDialog saveFileDialog = new SaveFileDialog
        {
            Filter = "Text File (*.txt)|*.txt",
            FileName = "Report.txt"
        };

        if (saveFileDialog.ShowDialog() == true)
        {
            _reportContext.SetStrategy(new TxtReportStrategy());

            var transactions = _transactionRepository.GetAll();

            _reportContext.CreateReport(transactions, saveFileDialog.FileName);

            MessageBox.Show("Текстовий звіт збережено!", "Успіх");
        }
    }

    private void CloseBtn_Click(object sender, RoutedEventArgs e)
    {
        this.Close();
    }
  }
}
```

Контрольні питання

1. **Що таке шаблон проєктування?**

   Це перевірене архітектурне рішення для типових проблем, що виникають при розробці ПЗ. Це не готовий код для копіювання, а концептуальна схема взаємодії об'єктів, яку адаптують під конкретну задачу.

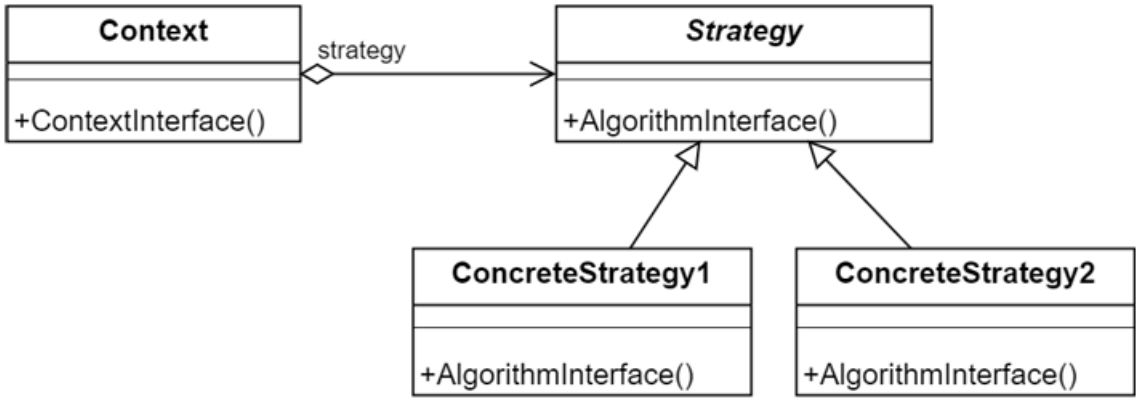
2. **Навіщо використовувати шаблони проєктування?**

   Вони дозволяють застосовувати надійні архітектурні рішення, роблячи код зрозумілішим та гнучкішим до змін. Також шаблони спрощують підтримку системи та покращують комунікацію в команді завдяки уніфікованій термінології.

3. **Яке призначення шаблону «Стратегія»?**

   Він визначає групу алгоритмів, інкапсулює кожен в окремий клас і робить їх взаємозамінними. Це дозволяє клієнту змінювати логіку виконання (наприклад, алгоритм сортування) динамічно, не змінюючи власний код.

4. **Нарисуйте структуру шаблону «Стратегія».**



5. **Які класи входять в шаблон «Стратегія», та яка між ними взаємодія?**

Context (Контекст): Клас, якому потрібно виконати певну роботу. Він зберігає посилання на інтерфейс Strategy і спілкується з ним, не знаючи конкретної реалізації.

Strategy (Стратегія): Спільний інтерфейс для всіх алгоритмів.

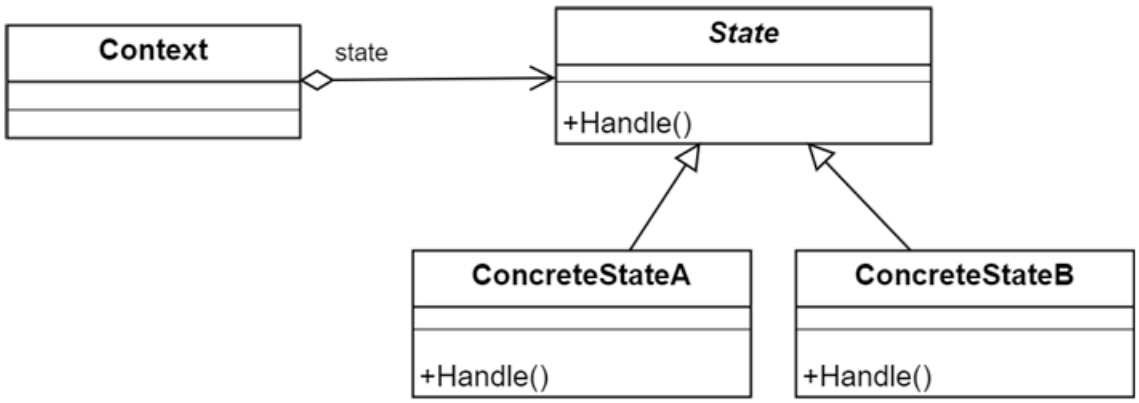
ConcreteStrategy (Конкретна Стратегія): Класи, що реалізують конкретний алгоритм.

Взаємодія: Клієнт створює об'єкт конкретної стратегії та передає його в Контекст. Контекст делегує виконання роботи об'єкту стратегії.

6. Яке призначення шаблону «Стан»?

Дозволяє об'єкту динамічно змінювати свою поведінку залежно від зміни його внутрішнього стану, наче об'єкт змінив свій клас.

7. Нарисуйте структуру шаблону «Стан».



8. Які класи входять в шаблон «Стан», та яка між ними взаємодія?

Context (Контекст): Об'єкт, поведінка якого змінюється. Зберігає посилання на поточний Стан.

State (Стан): Інтерфейс для інкапсуляції поведінки.

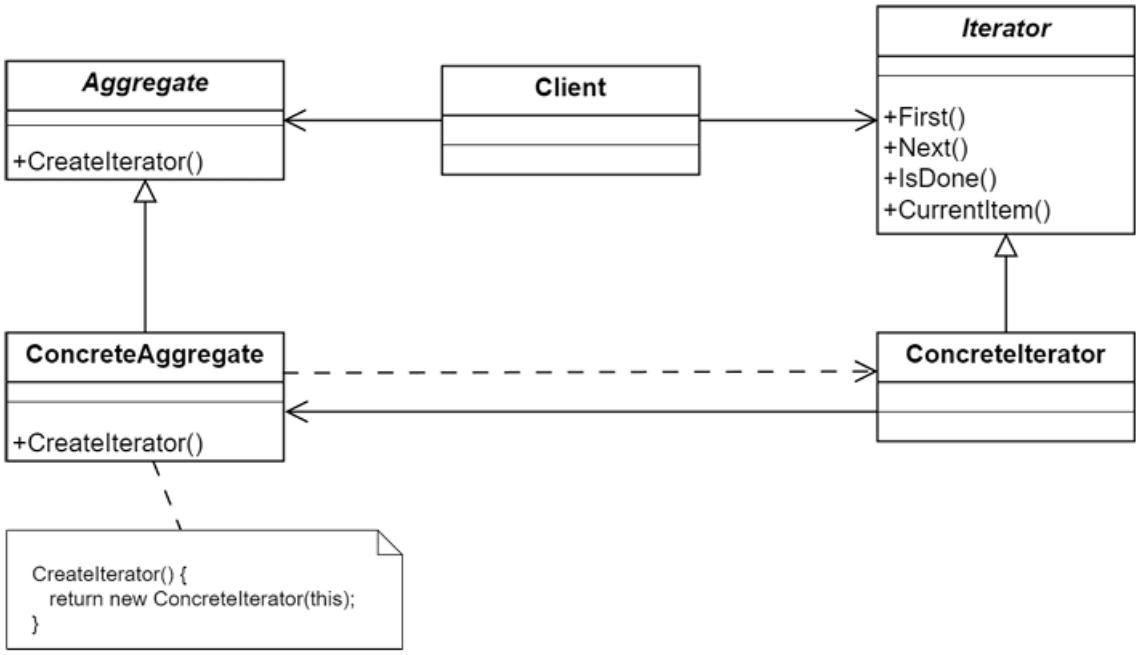
ConcreteState (Конкретний стан): Реалізує поведінку, специфічну для певної стадії життя об'єкта.

Взаємодія: Контекст делегує виконання операцій об'єкту поточного стану. Самі стани можуть ініціювати зміну стану в Контексті (перехід від A до B).

9. Яке призначення шаблону «Ітератор»?

Забезпечує послідовний доступ до елементів колекції без розкриття її внутрішньої структури (чи це список, чи стек, чи дерево).

10. Нарисуйте структуру шаблону «Ітератор».



11. Які класи входять в шаблон «Ітератор», та яка між ними взаємодія?

Aggregate (Колекція): інтерфейс для створення ітератора.

Iterator: визначає методи для перебору елементів та відстеження поточної позиції.

Взаємодія: Клієнт отримує ітератор від колекції та використовує його методи для циклічного проходу по елементах.

12. В чому полягає ідея шаблона «Одинак»?

Гарантувати, що клас має лише один екземпляр у системі, та надати до нього глобальну точку доступу.

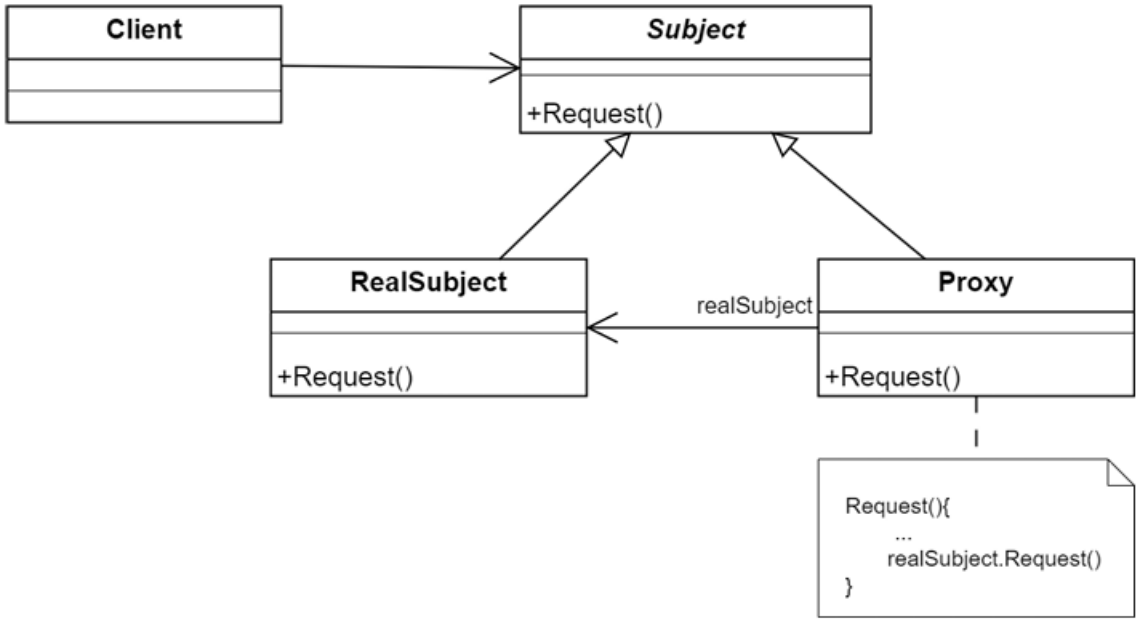
13. Чому шаблон «Одинак» вважають «анти-шаблоном»?

Він створює глобальний стан, що ускладнює відстеження помилок, та формує жорсткі залежності, які заважають модульному тестуванню. Також він порушує принцип єдиної відповідальності (SRP), змішуючи бізнес-логіку з контролем власного створення.

14. Яке призначення шаблону «Проксі»?

Надає об'єкт-замінник для контролю доступу до іншого об'єкта. Використовується для кешування, перевірки прав, логування або «лінивої» ініціалізації «важких» об'єктів.

15. Нарисуйте структуру шаблону «Проксі».



16. Які класи входять в шаблон «Проксі», та яка між ними взаємодія?

Subject: Спільний інтерфейс для реального об'єкта та замісника.

RealSubject: Об'єкт, який виконує реальну роботу.

Proxy: Зберігає посилання на RealSubject. Реалізує той самий інтерфейс Subject.

Клієнт працює з Proxy як з реальним об'єктом. Proxy може виконати щось "до" або "після" виклику (наприклад, перевірити права) і, якщо потрібно, передає виклик RealSubject.

Висновки: Під час виконання лабораторної роботи я дослідила структурні та поведінкові шаблони проєктування, зокрема «Singleton», «Iterator», «Proxy» та «Strategy», а також набула практичних навичок їх застосування при розробці програмної системи «Особиста бухгалтерія». Для реалізації функціоналу експорту звітів було обрано та впроваджено шаблон «Стратегія» (Strategy). Це дозволило відокремити логіку збереження даних від клієнтського коду та створити гнучкий механізм генерації документів у різних форматах (Excel .xlsx за допомогою бібліотеки ClosedXML та текстовий формат .txt). Завдяки використанню інтерфейсу IReportStrategy та класу-контексту ReportContext, архітектура системи стала модульною та масштабованою, що дозволяє в майбутньому легко додавати нові типи звітів без необхідності внесення змін у вже існуючий та протестований код.

Репозиторій: посилання

ДОДАТКИ

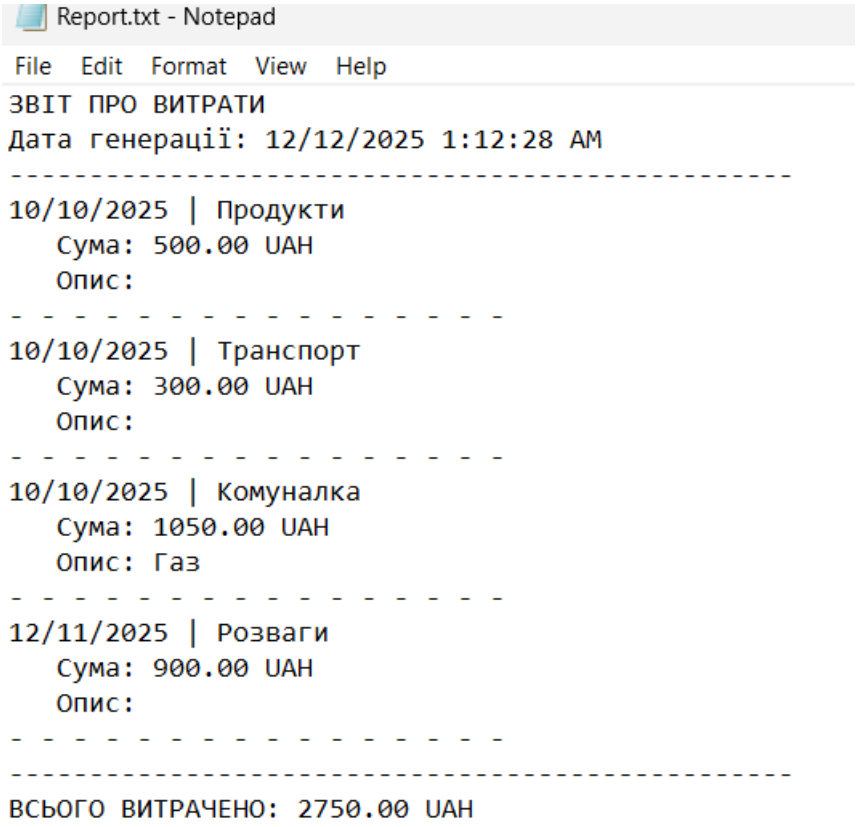
Додаток А

Результат експорту статистичного звіту у формат Microsoft Excel (.xlsx)

	A	B	C	D	E	F	G
1	**ID**	**Дата**	**Сума**	**Категорія**	**Тип**	**Опис**	
2	2	10.10.2025	500	Продукти	Expense		
3	3	10.10.2025	300	Транспорт	Expense		
4	4	10.10.2025	1050	Комуналка	Expense	Газ	
5	5	11.12.2025	900	Розваги	Expense		
6							
7							

Додаток Б

Результат експорту статистичного звіту у текстовий формат (.txt)


Report.txt - Notepad

File Edit Format View Help

```
ЗВІТ ПРО ВИТРАТИ
Дата генерації: 12/12/2025 1:12:28 AM
------------------------------------------------
10/10/2025 | Продукти
   Сума: 500.00 UAH
   Опис:
- - - - - - - - - - - - - - - -
10/10/2025 | Транспорт
   Сума: 300.00 UAH
   Опис:
- - - - - - - - - - - - - - - -
10/10/2025 | Комуналка
   Сума: 1050.00 UAH
   Опис: Газ
- - - - - - - - - - - - - - - -
12/11/2025 | Розваги
   Сума: 900.00 UAH
   Опис:
- - - - - - - - - - - - - - - -
------------------------------------------------
ВСЬОГО ВИТРАЧЕНО: 2750.00 UAH
```