Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Лабораторна робота № 4

Технології розроблення програмного забезпечення

«Вступ до паттернів проектування»

20. Mind-mapping software

Виконав:
студент групи ІА–33:
Хитрова НА

Перевірив:
Мягкий МЮ

Київ 2025

Тема: Вступ до паттернів проектування.

Мета: Вивчити структуру шаблонів «Singleton», «Iterator», «Proxy», «State», «Strategy» та навчитися застосовувати їх в реалізації програмної системи.

4.1. 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.

4.2. Теоретичні відомості

Будь-який патерн проєктування, використовуваний при розробці інформаційних систем, являє собою формалізований опис, який часто зустрічається в завданнях проєктування, вдале рішення даної задачі, також рекомендації по застосуванню цього рішення в різних ситуаціях [5]. Крім того, патерн проєктування обов’язково має загальновживане найменування. Правильно сформульований патерн проєктування дозволяє, відшукавши одного разу вдале рішення, користуватися ним знову і знову. Варто підкреслити, що важливим початковим етапом при роботі з патернами є адекватне моделювання розглянутої предметної області. Це є необхідним як для отримання належним чином формалізованої постановки задачі, так і для вибору відповідних патернів проєктування.

Відповідне використання патернів проєктування дає розробнику ряд незаперечних переваг. Наведемо деякі з них. Модель системи, побудована в межах патернів проєктування, фактично є структурованим виокремленням тих елементів і зв’язків, які значимі при вирішенні поставленого завдання. Крім цього, модель, побудована з використанням патернів проєктування, більш проста і наочна у вивченні, ніж стандартна модель. Проте, не дивлячись на простоту і наочність, вона дозволяє глибоко і всебічно

опрацювати архітектуру розроблюваної системи з використанням спеціальної мови.

Застосування патернів проєктування підвищує стійкість системи до зміни вимог та спрощує неминуче подальше доопрацювання системи. Крім того, важко переоцінити роль використання патернів при інтеграції інформаційних систем організації. Також слід зазначити, що сукупність патернів проєктування, по суті, являє собою єдиний словник проєктування, який, будучи уніфікованим засобом, незамінний для спілкування розробників один одним.

Таким чином шаблони представляють собою, підтверджені роками розробок в різних компаніях і на різних проєктах, «ескізи» архітектурних рішень, які зручно застосовувати у відповідних обставинах.

4.2.6. Шаблон «Strategy»

Призначення: Шаблон «Strategy» (Стратегія) дозволяє змінювати деякий алгоритм поведінки об'єкта іншим алгоритмом, що досягає ту ж мету іншим способом. Прикладом можуть служити алгоритми сортування: кожен алгоритм має власну реалізацію і визначений в окремому класі; вони можуть бути взаємозамінними в об'єкті, який їх використовує [6].

Даний шаблон дуже зручний у випадках, коли існують різні «політики» обробки даних. По суті, він дуже схожий на шаблон «State» (Стан), проте використовується в абсолютно інших цілях – незалежно від стану об'єкта відобразити різні можливі поведінки об'єкта (якими досягаються одні й ті самі або схожі цілі).

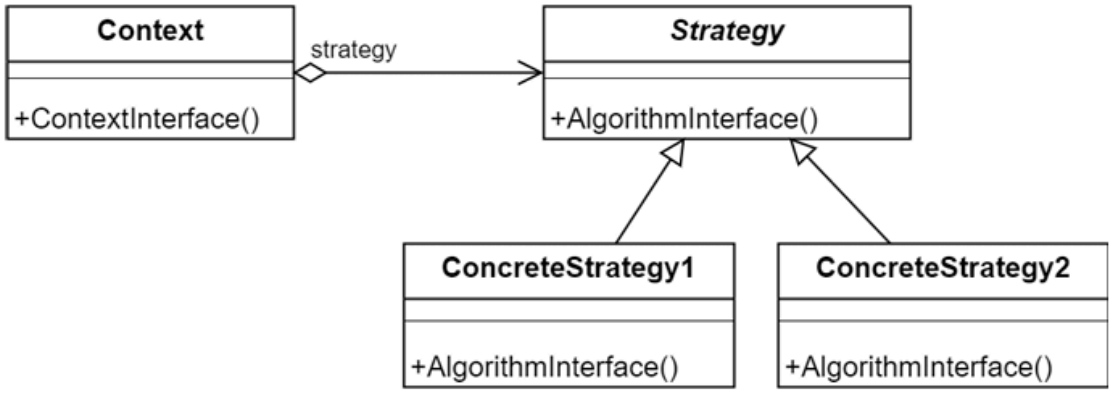


Рисунок 4.5. Структура патерну Стратегія.

Рішення: Коли ви використовуєте патерн «Стратегія», то схожі алгоритми виносяться з класа контекста в конкретні стратегії, за рахунок чого клас контексту стає чистіше і його легше супроводжувати. Також одні

і тіж самі стратегії можна використати з різними контекстами, що значно збільшує гнучкість вашої системи та зменшує кількість дублювань у коді.

Контекст при цьому містить посилання на конкретну стратегію, а коли стратегію потрібно замінити, то замінюється об’єкт стратегії в полі Context.strategy.

Важливою умовою є відносна простота інтерфейсу для алгоритмів стратегій. Якщо алгоритмам стратегій прийдеться передавати десятки параметрів, то це, скоріш за все, приведе до ускладнення системи та заплутаності коду. Якщо стратегії на вході будуть приймати об’єкт контексту, щоб отримувати з нього всі необхідні дані, то такі стратегії будуть прив’язані до конкретного контексту і їх не можна буде використати з іншим типом контексту.

Приклад з життя: Ви їдете на роботу. Можна доїхати на автомобілі, на метро, або йти пішки. Тут алгоритм, як ви добираєтеся на роботу, є стратегією. В залежності від поточної ситуації ви вибираєте стратегію, що найбільше підходить в цій ситуації, наприклад, на дорогах великі пробки тоді ви їдете на метро, або метро тимчасово не ходить тоді ви їдете на таксі, або ви знаходитеся в 5 хвилинах ходьби від місця роботи і простіше добратися пішки.

Переваги та недоліки:

+ Використовувані алгоритми можна змінювати під час виконання.

+ Реалізація алгоритмів відокремлюється від коду, що його використовує.

+ Зменшує кількість умовних операторів типу switch та if в контексті.

- Надмірна складність, якщо у вас лише кілька невеликих алгоритмів.

- Під час виклику алгоритму, клієнтський код має враховувати різницю між стратегіями.

Хід роботи

У нашій системі є логічна операція “Експортувати мапу”, але форматів може бути кілька, такі як PNG, JPG, PDF. По суті, це одна команда з різними алгоритмами всередині. І це точне визначення задачі для патерну Strategy.

Виділяємо інтерфейс ExportStrategy, який описує контракт для всіх алгоритмів експорту. Для кожного формату створюється окремий клас (PngExportStrategy, JpgExportStrategy, PdfExportStrategy), що реалізує цей інтерфейс. Клас ExportService виступає в ролі контексту: він отримує від

інтерфейсу користувача вибір формату, підставляє відповідну стратегію і викликає її метод export().

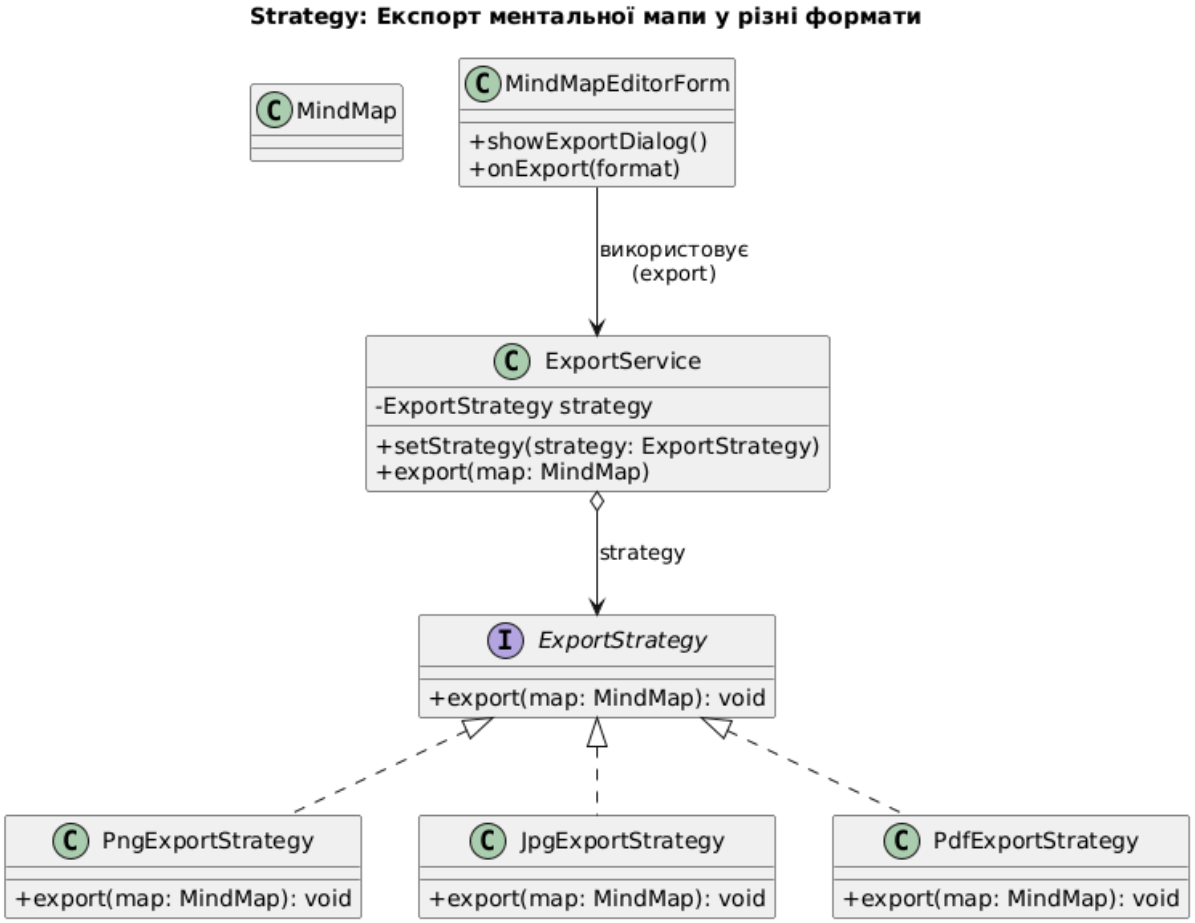


Рис. 1. UML-діаграму класів для Strategy

На даній діаграмі зображено структуру класів, яка демонструє застосування шаблону проектування Strategy у функціоналі експорту ментальних карт у різні графічні формати. Основною метою використання цього шаблону є забезпечення гнучкості та розширюваності системи при додаванні нових способів експорту без необхідності змінювати існуючий код інтерфейсу чи бізнес-логіки.

Інтерфейс ExportStrategy – це загальний контракт стратегій. Цей метод описує абстрактну операцію експорту ментальної мапи у файл. Інтерфейс не визначає, як саме буде виконано експорт, лише що він має бути виконаний. Таким чином, конкретні алгоритми експорту (PNG, JPG, PDF) можуть бути реалізовані незалежно один від одного.

Інтерфейс ExportStrategy реалізують три спеціалізовані класи:

- PngExportStrategy
- JpgExportStrategy

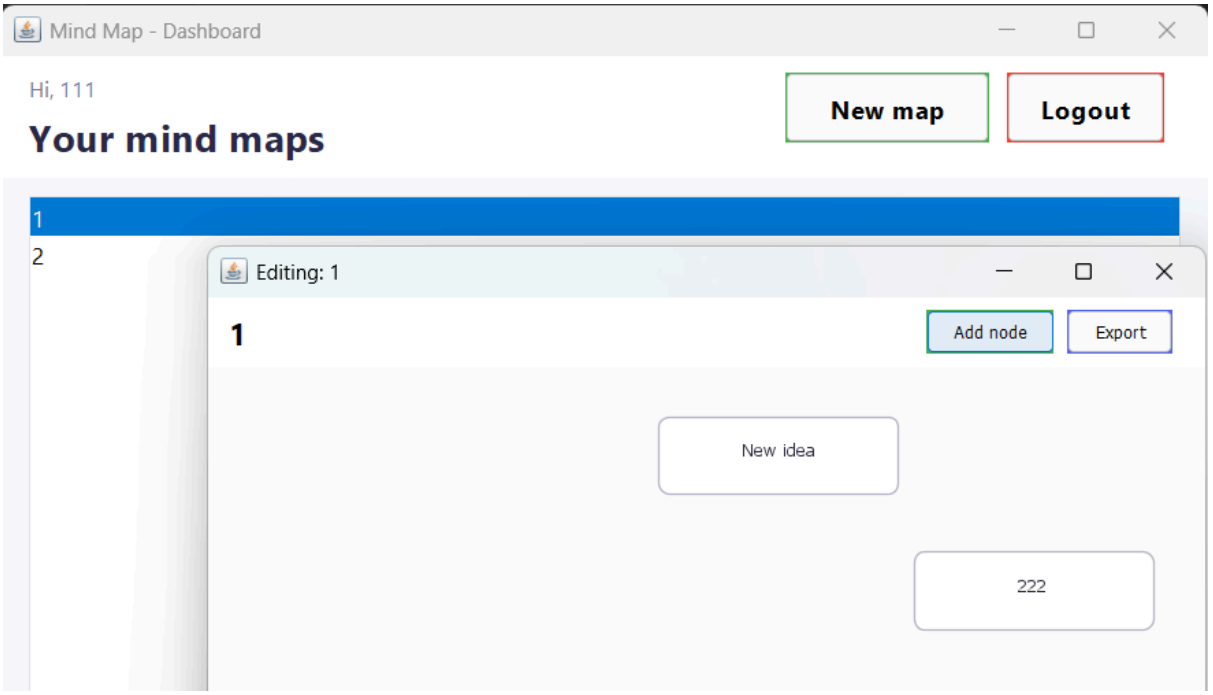
- PdfExportStrategy

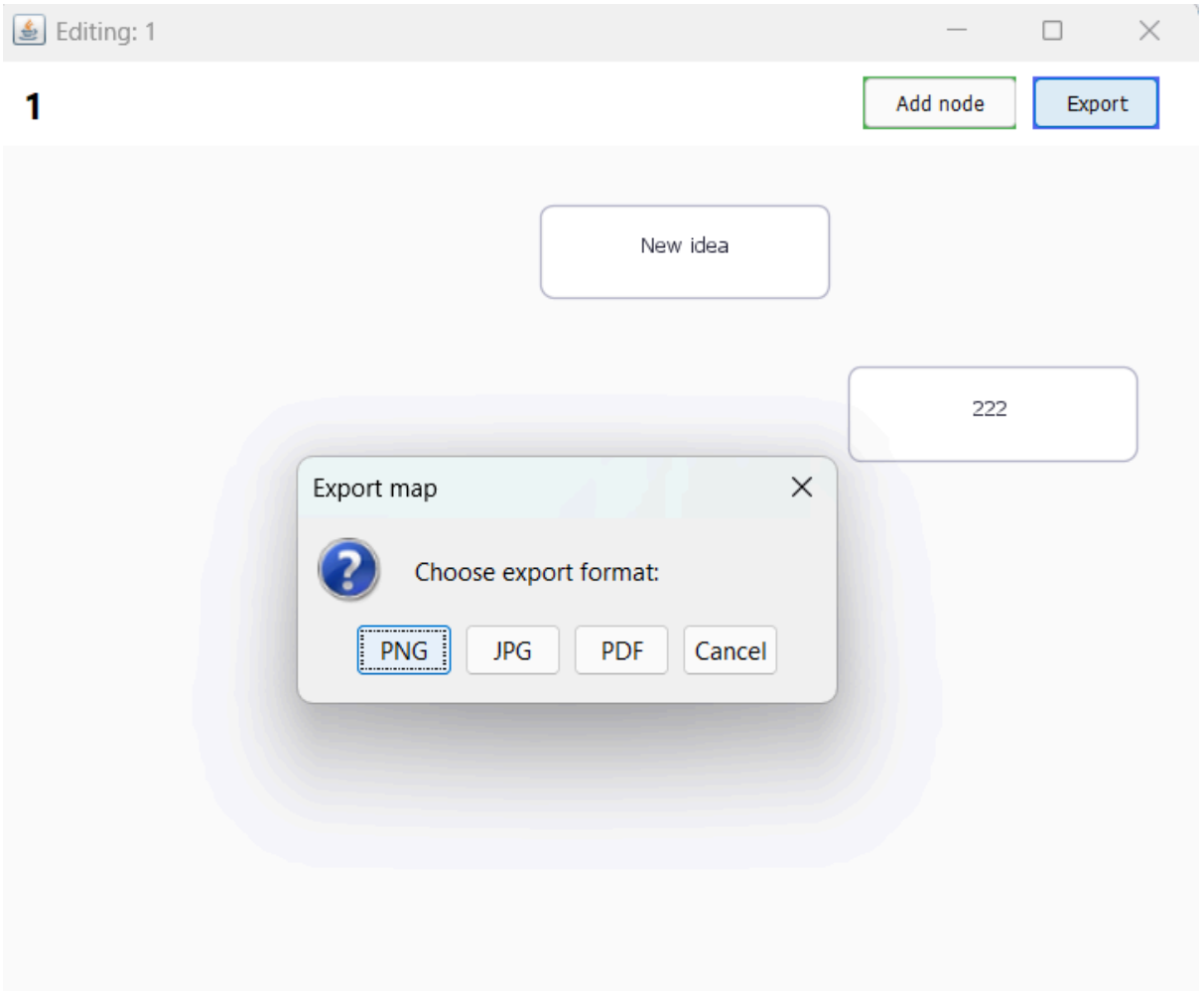
Кожна з цих реалізацій містить власний варіант алгоритму експорту.

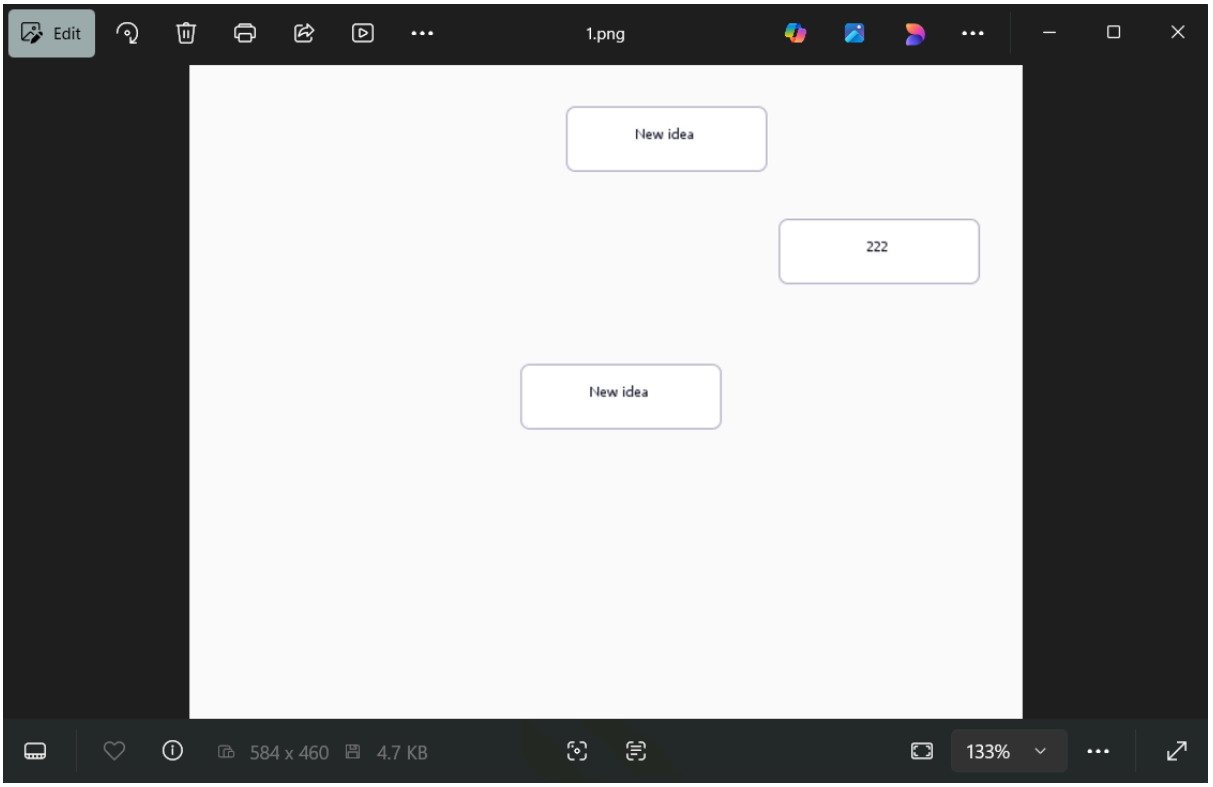
Клас ExportService виконує роль контексту, який працює зі стратегією. Він містить посилання на об’єкт типу ExportStrategy, має метод setStrategy(), що дозволяє вибирати потрібний алгоритм експорту під час виконання, реалізує метод export(map), який просто делегує роботу поточній стратегії. Контекст нічого не знає про деталі формування PNG, JPG або PDF, він лише викликає метод відповідної стратегії. Це дозволяє змінювати поведінку системи динамічно, у момент взаємодії з користувачем.

Форма редагування ментальної мапи MindMapEditorForm взаємодіє з ExportService. Вона показує користувачу діалог вибору формату експорту, на основі вибору встановлює в ExportService відповідну стратегію, викликає метод export(), який і запускає процес збереження. Таким чином, інтерфейс користувача залишається максимально простим і не залежить від технічних деталей експорту, UI лише визначає, що саме хоче користувач, а як це зробити вирішує стратегія.

Реалізація експорту в нашому додатку:


Mind Map - Dashboard
Hi, 111
Your mind maps
New map
Logout
1
2
Editing: 1
1
Add node
Export
New idea
222


Editing: 1
1
Add node
Export
New idea
222
Export map
Choose export format:
PNG
JPG
PDF
Cancel



Лістинг вихідного код системи, який було додано в цій лабораторній роботі:

https://github.com/ch1fir/trpz-lab-4

ExportService.java:

```
package com.example.mindmap.export;

import com.example.mindmap.entities.MindMap;

import javax.swing.*;
import java.io.File;
import java.io.IOException;

public class ExportService {

   private ExportStrategy strategy;
```

```
    public void setStrategy(ExportStrategy strategy) {
        this.strategy = strategy;
    }

    public void export(MindMap map, JComponent canvas, File targetFile)
throws IOException {
        if (strategy == null) {
            throw new IllegalStateException("Export strategy is not set");
        }
        strategy.export(map, canvas, targetFile);
    }
}
```

ExportStrategy.java:

```
package com.example.mindmap.export;

import com.example.mindmap.entities.MindMap;

import javax.swing.*;
import java.io.File;
import java.io.IOException;

public class ExportService {

    private ExportStrategy strategy;

    public void setStrategy(ExportStrategy strategy) {
        this.strategy = strategy;
    }

    public void export(MindMap map, JComponent canvas, File targetFile)
throws IOException {
        if (strategy == null) {
            throw new IllegalStateException("Export strategy is not set");
        }
        strategy.export(map, canvas, targetFile);
    }
}
```

PngExportStrategy.java:

```
package com.example.mindmap.export;

import com.example.mindmap.entities.MindMap;

import javax.imageio.ImageIO;
import javax.swing.*;
import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
```

```
public class PngExportStrategy implements ExportStrategy {

   @Override
   public void export(MindMap map, JComponent canvas, File targetFile)
throws IOException {
       int w = canvas.getWidth();
       int h = canvas.getHeight();

       if (w <= 0 || h <= 0) {
           throw new IOException("Canvas has invalid size");
       }

       BufferedImage image = new BufferedImage(w, h,
BufferedImage.TYPE_INT_ARGB);
       Graphics2D g2 = image.createGraphics();
       canvas.printAll(g2);
       g2.dispose();

       ImageIO.write(image, "png", targetFile);
   }
}
```

JpgExportStrategy.java:

```
package com.example.mindmap.export;

import com.example.mindmap.entities.MindMap;

import javax.imageio.ImageIO;
import javax.swing.*;
import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;

public class JpgExportStrategy implements ExportStrategy {

   @Override
   public void export(MindMap map, JComponent canvas, File targetFile)
throws IOException {
       int w = canvas.getWidth();
       int h = canvas.getHeight();

       if (w <= 0 || h <= 0) {
           throw new IOException("Canvas has invalid size");
       }

       BufferedImage image = new BufferedImage(w, h,
BufferedImage.TYPE_INT_RGB);
       Graphics2D g2 = image.createGraphics();
       // Білий фон, щоб не було чорного фону прозорості
       g2.setColor(Color.WHITE);
       g2.fillRect(0, 0, w, h);
```

```
        canvas.printAll(g2);
        g2.dispose();

        ImageIO.write(image, "jpg", targetFile);
    }
}
```

PdfExportStrategy.java:

```
package com.example.mindmap.export;

import com.example.mindmap.entities.MindMap;

import javax.swing.*;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;

public class PdfExportStrategy implements ExportStrategy {

    @Override
    public void export(MindMap map, JComponent canvas, File targetFile)
throws IOException {
        // Спрощена реалізація: текстовий псевдо-PDF.
        // Для реального PDF потрібна окрема бібліотека.
        try (FileWriter writer = new FileWriter(targetFile)) {
            writer.write("Mind Map Export (pseudo PDF)\n");
            writer.write("Title: " + map.getTitle() + "\n");
            writer.write("Canvas size: " + canvas.getWidth() + "x" +
canvas.getHeight() + "\n");
            writer.write("Note: For real PDF export a PDF library should be
used.\n");
        }
    }
}
```

Висновок: У цій лабораторній роботі ми додали можливість експортувати мапу у різні формати за допомогою шаблону Strategy, а конкретно ми реалізували три стратегії, PngExportStrategy, яка робить “скріншот” CanvasPanel у BufferedImage і зберігає як .png. JpgExportStrategy, те саме, але з білим фоном і форматом .jpg та PdfExportStrategy, але спрощена стратегія, відбувається створення текстового файлу з розширенням .pdf, записуємо туди назву мапи і розміри полотна. (Для реального PDF потрібна окрема бібліотека, але для демонстрації патерну цього достатньо.) Контекст ExportService інкапсулює логіку вибору алгоритму, а самі алгоритми знаходяться в окремих класах. Поза лабораторною роботою ми додали можливість додали окреме вікно

редагування мапи та панель для відмальовування вузлів та зробили так, щоб наповнення мап зберігалося.

4.3. Питання до лабораторної роботи

1. Що таке шаблон проєктування?

Шаблон проєктування – це перевірене рішення типової задачі проектування, яке можна повторно використовувати при створенні програмних систем.

2. Навіщо використовувати шаблони проєктування?

Шаблони допомагають писати більш гнучкий, зрозумілий і підтримуваний код, уникати типових помилок і створювати архітектуру, яку легко розширювати.

3. Яке призначення шаблону «Стратегія»?

Стратегія дозволяє визначити кілька алгоритмів та підміняти їх під час виконання, не змінюючи код клієнта. Вона розділяє логіку вибору алгоритму і його реалізацію.

4. Нарисуйте структуру шаблону «Стратегія».

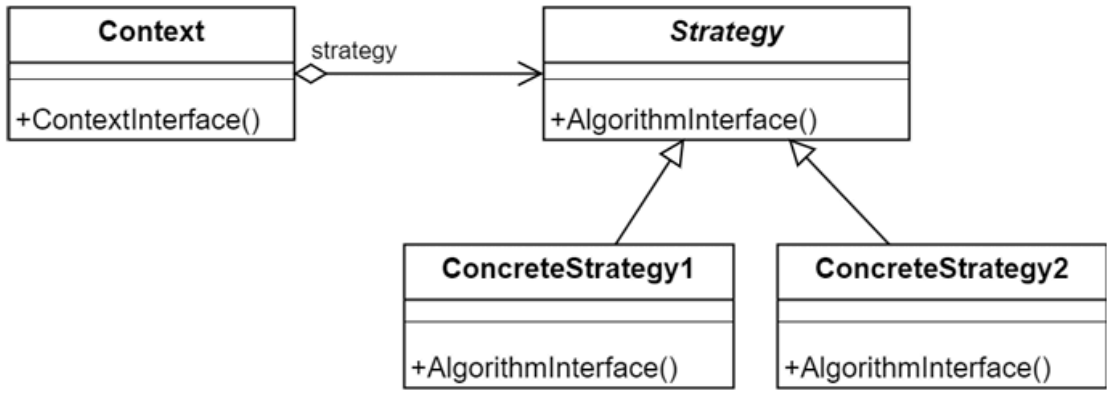


Рисунок 4.5. Структура патерну Стратегія.

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

У шаблон входять:

Стратегія (інтерфейс) – описує спільний метод для всіх алгоритмів;

Конкретні стратегії – різні реалізації алгоритмів;

Контекст – клас, який зберігає посилання на стратегію і викликає її метод.

Контекст не знає деталей реалізації, а лише користується методом стратегії.

6. Яке призначення шаблону «Стан»?

Шаблон «Стан» дозволяє змінювати поведінку об’єкта в залежності від його внутрішнього стану. Об’єкт виглядає так, ніби він змінює свій клас під час роботи.

7. Нарисуйте структуру шаблону «Стан».

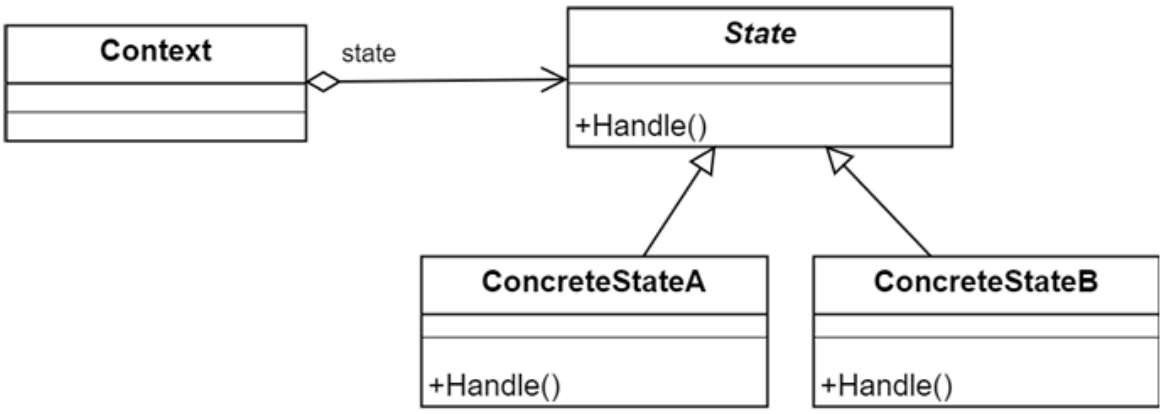


Рисунок 4.4. Структура патерну Стан

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

Компоненти:

Інтерфейс стану – описує спільні дії для всіх станів;

Конкретні стани – реалізують поведінку для певного стану;

Контекст – містить поточний стан і делегує йому виконання дій.

Контекст перемикає стани, коли логіка цього вимагає.

9. Яке призначення шаблону «Ітератор»?

Ітератор дає можливість послідовно обходити елементи колекції, не розкриваючи її внутрішню структуру.

10. Нарисуйте структуру шаблону «Ітератор».

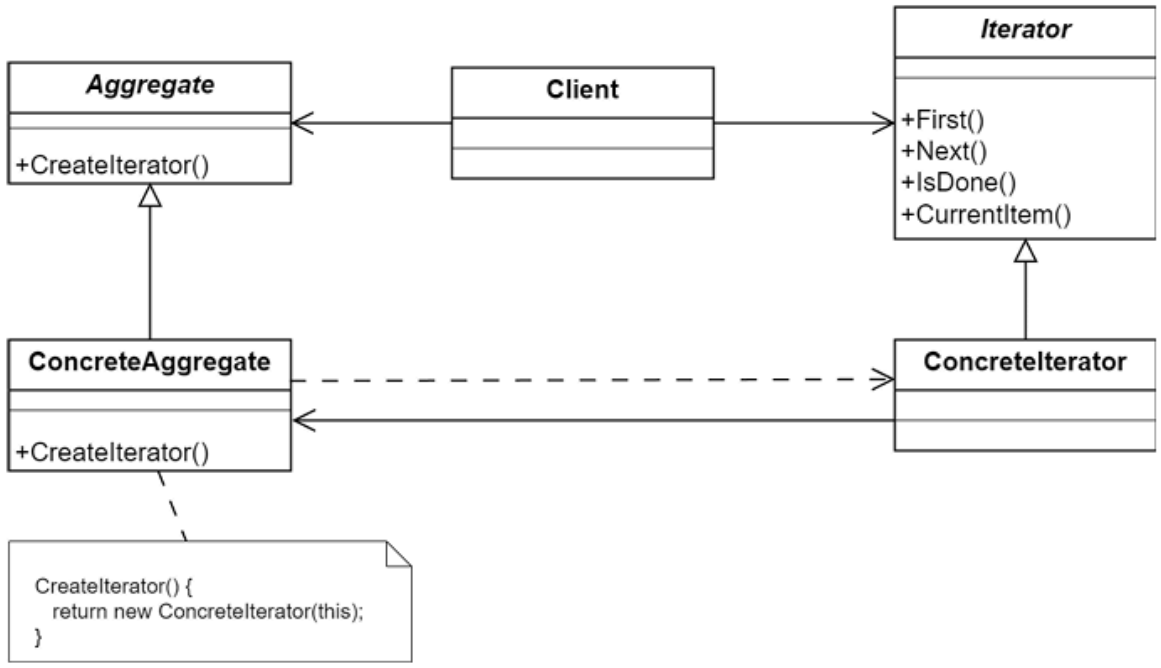


Рисунок 4.2. Структура патерну Ітератор

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

Входять:

Ітератор (інтерфейс) – описує методи переходу по елементах;

Конкретний ітератор – реалізує обходу конкретної колекції;

Колекція (aggregate) – надає метод для створення ітератора;

Конкретна колекція – зберігає елементи та створює свій ітератор.

Ітератор отримує доступ до елементів через колекцію, але не знає її структури.

12. В чому полягає ідея шаблона «Одинак»?

Одинак гарантує, що клас матиме лише один екземпляр, і забезпечує глобальний доступ до нього.

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

Тому що він створює приховані залежності, ускладнює тестування, порушує принципи SOLID і часто призводить до використання глобального стану, що робить систему менш гнучкою.

14. Яке призначення шаблону «Проксі»?

Проксі створює замінник або «посередника» для іншого об’єкта, контролюючи доступ до нього – наприклад, додає кешування, логування або ліниве завантаження.

15. Нарисуйте структуру шаблону «Проксі».

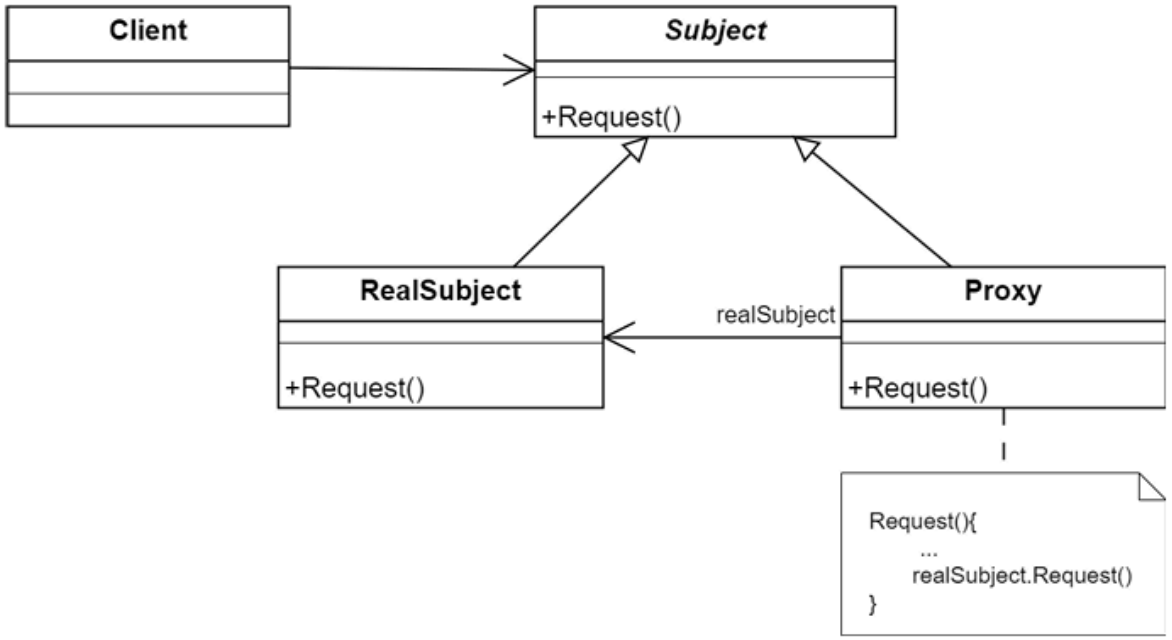


Рисунок 4.3. Структура патерну Proxy

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

У шаблон входять:

Суб’єкт (інтерфейс) – спільний інтерфейс для реального об’єкта й проксі;

Реальний суб’єкт – справжній об’єкт, до якого здійснюється доступ;

Проксі – містить посилання на реальний об’єкт і контролює доступ до нього.

Клієнт працює з проксі так само, як із реальним об’єктом.