



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 №4

Технології розроблення програмного забезпечення

Тема проєкту ‘Shell (total commander)’

Виконала: студентка групи ІА-33

Маляревич Анжела Валентинівна

Перевірив:

Мягкий Михайло Юрійович

Київ - 2025

Зміст

Теоретичні відомості.....	3
Шаблон проектування.....	3
Singleton (Одинак).....	3
Iterator (Ітератор).....	4
Proxy (Проксі, Замісник).....	4
State (Стан).....	4
Strategy (Стратегія).....	4
Хід роботи.....	5
Діаграма класів.....	5
OperationState.java.....	6
OperationType.java.....	6
InProgressState.java.....	7
OperationRequest.java.....	7
OperationContext.java.....	8
Форма пошуку.....	9
Висновок.....	11
Питання до лабораторної роботи.....	11

Тема: Вступ до паттернів проектування.

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

Тема роботи:

18. Shell (total commander) (state, prototype, factory method, template method, interpreter, client-server) Оболонка повинна вміти виконувати основні дії в системі – перегляд файлів папок в файлової системі, перемикання між дисками, копіювання, видалення, переміщення об'єктів, пошук.

Теоретичні відомості

Шаблон проектування

Шаблон проектування – це типові рішення поширеної проблеми під час розробки програмного забезпечення. Він описує структуру класів і їх взаємодію. Використання шаблонів робить код гнучким, зрозумілим, зменшує дублювання та полегшує супровід програми.

Singleton (Одинак)

Забезпечує існування лише одного екземпляра класу і надає глобальний доступ до нього.

Приклад: конфігураційний файл або єдине підключення до бази даних.

Переваги: гарантує один об'єкт у системі, спрощує доступ.

Недоліки: порушує принцип єдиної відповідальності, ускладнює тестування та супровід.

Iterator (Ітератор)

Дозволяє послідовно обходити елементи колекції без розкриття її внутрішньої структури.

Основні методи: First(), Next(), IsDone(), CurrentItem().

Переваги: забезпечує єдиний інтерфейс обходу для різних колекцій, підтримує різні способи проходження.

Недоліки: може бути надмірним для простих структур.

Proxy (Проксі, Замісник)

Додає проміжний об'єкт, який контролює доступ до іншого об'єкта або додає додаткову поведінку.

Приклад: кешування, перевірка доступу, робота з віддаленим сервісом.

Переваги: дозволяє змінювати логіку без зміни клієнтського коду, покращує безпеку.

Недоліки: може знизити швидкодію, ускладнює структуру.

State (Стан)

Дозволяє змінювати поведінку об'єкта залежно від його стану.

Приклад: банківська картка може бути активною, заблокованою або закритою.

Переваги: спрощує код контексту, легко додавати нові стани.

Недоліки: збільшує кількість класів, ускладнює переходи між станами.

Strategy (Стратегія)

Дозволяє змінювати алгоритм роботи програми під час виконання.

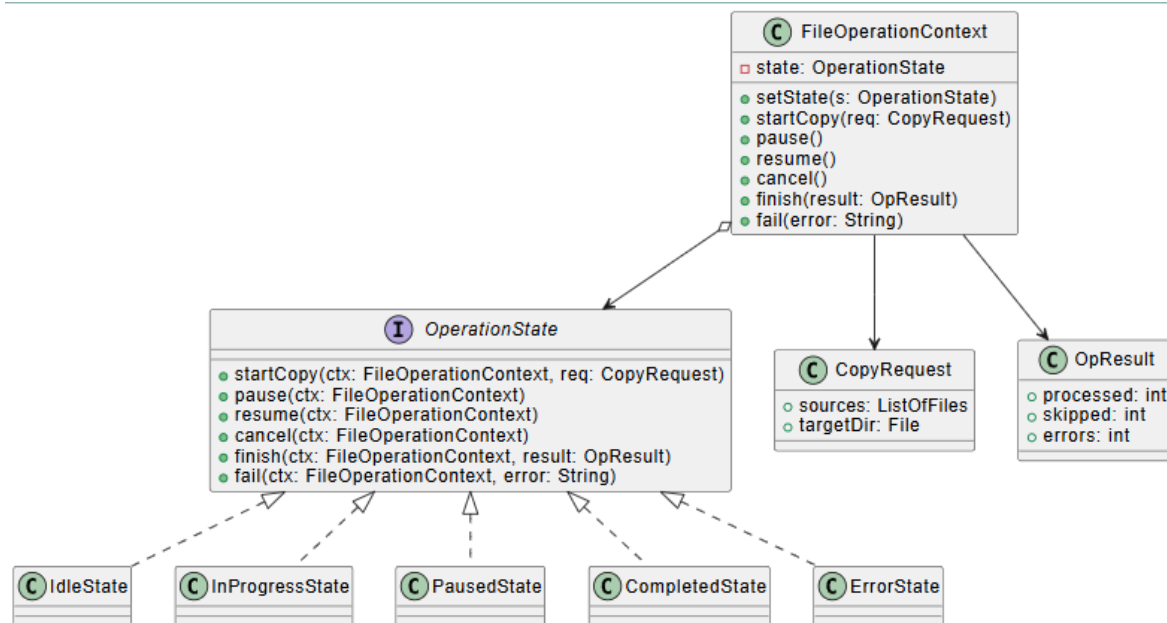
Приклад: вибір різних способів оплати або алгоритмів сортування.

Переваги: підвищує гнучкість, зменшує кількість умовних операторів, розділяє реалізацію алгоритмів.

Недоліки: створює додаткові класи, потребує знань про різні стратегії.

Хід роботи

Діаграма класів



У проєкті я застосувала State для керування довгими операціями файлового менеджера (пошук/копіювання/переміщення/видалення). Контекст (OperationContext) інкапсулює стан і службові поля прогресу/повідомлень, а конкретні стани (Idle/Preparing/InProgress/Completed/Failed/Canceled) визначають поведінку та переходи. Такий підхід прибрав “if-else по типах” із контролерів, дозволив централізовано відстежувати прогрес і обробляти скасування, а також спростив розширення новими операціями (дати тип + стан без зміни існуючих класів)

OperationState.java

```
1 package ua.kpi.ua33.shellweb.patterns.state.op;
2
3 public interface OperationState { 9 usages 6 implementations
4     void start(OperationContext ctx, OperationRequest req); 6 implementations
5     void cancel(OperationContext ctx); 1 usage 6 implementations
6
7     String name(); 6 implementations
8     boolean isTerminal(); 1 usage 6 implementations
9
10    int progress(OperationContext ctx); 6 implementations
11    String statusMessage(OperationContext ctx); no usages 6 implementations
12 }
13
```

визначає контракт для всіх станів у шаблоні State. Саме цей інтерфейс задає методи start(), cancel(), progress(), statusMessage(), а також isTerminal()

OperationType.java

```
1 package ua.kpi.ua33.shellweb.patterns.state.op;
2
3 public enum OperationType { 13 usages
4     SEARCH, COPY, MOVE, DELETE 2 usages
5 }
6
```

продемонстровано перелік типів операцій файлової системи (SEARCH, COPY, MOVE, DELETE). Перерахування використовується у машині станів для визначення того, яку дію потрібно виконати, і передається у OperationRequest

InProgressState.java

```
35         switch (req.getType()) {
36             case SEARCH -> {
37                 User user = req.getUser();
38                 if (user == null) throw new IllegalStateException("User is required for SEARCH");
39
40                 SearchQuery q = ctx.searchService().createQuery(user, req.getNameMask(), req.getExt());
41                 ctx.setProgress(100);
42                 ctx.setMessage("Пошук завершено: запит #" + q.getId());
43             }
44         }
```

Саме у цьому стані відбувається фактичне виконання довгої операції:

- пошук файлів — через `SearchService.createQuery(...)`;
- копіювання/переміщення — через `FileOpsService` з оновленням прогресу;
- обробка скасування (cancelled).

Після завершення стан переходить у `Completed` або `Failed`.

OperationRequest.java

```
SearchOp.java  OperationContext.java  op\OperationType.java  op\OperationRequest.java x
1  package ua.kpi.ua33.shellweb.patterns.state.op;
2
3  import ua.kpi.ua33.shellweb.domain.User;
4  import java.nio.file.Path;
5
6  public class OperationRequest {
7      private final OperationType type; 2 usages
8
9      // SEARCH
10     private String nameMask; 2 usages
11     private String ext; 2 usages
12     private User user; 2 usages
13
14     // FILE OPS
15     private Path source; 2 usages
16     private Path target; 2 usages
17
18
19     public OperationRequest(OperationType type) {
20         this.type = type;
21     }
22
23
24     public OperationRequest nameMask(String nameMask) { no usages
25         this.nameMask = nameMask;
26         return this;
27     }
28
29     public OperationRequest ext(String ext) { no usages
30         this.ext = ext;
31         return this;
32     }
}
```

На рисунку зображено клас `OperationRequest`, який є універсальним контейнером параметрів операції.

Він містить поля для пошуку (маска, розширення, користувач) і для файлових операцій (source, target).

Саме цей об'єкт передається у контекст станів і визначає, яку операцію потрібно виконувати.

OperationContext.java

```
SearchOp.java  © OperationContext.java  ×  © op\OperationType.java  © op\OperationRequest.java  ⓘ OperationSt
1 package ua.kpi.ia33.shellweb.patterns.state.op;
2
3 import ua.kpi.ia33.shellweb.service.SearchService;
4 import ua.kpi.ia33.shellweb.service.FileOpsService;
5
6 import java.util.UUID;
7 import java.util.concurrent.atomic.AtomicInteger;
8 import java.util.concurrent.atomic.AtomicReference;
9
10 import ua.kpi.ia33.shellweb.patterns.prototype.fs.FsOperation;
11
12
13 public class OperationContext {
14     private final String id = UUID.randomUUID().toString(); 1 usage
15     private final AtomicReference<OperationState> state = new AtomicReference<>(); 5 usages
16     private final AtomicInteger progress = new AtomicInteger( initialValue: 0 ); 2 usages
17     private volatile String message; 2 usages
18     private volatile String error; 2 usages
19
20     private final SearchService searchService; 2 usages
21     private final FileOpsService fileOpsService; 2 usages
22
23     public OperationContext(SearchService searchService, FileOpsService fileOpsService) {
24         this.searchService = searchService;
25         this.fileOpsService = fileOpsService;
26         this.state.set(new IdleState());
27     }
28
29     public String id() { return id; } no usages
30     public OperationState getState() { return state.get(); }
31     public void setState(OperationState newState) { state.set(newState); }
32 }
```

Це центральний елемент шаблону State — він делегує поведінку поточному стану.

Форма пошуку

Старт операції

Тип:

Пошук

Маска імені:

report

Розширення:

pdf

Джерело (повний шлях):

C:\in\file.txt

Ціль (для copy/move):

D:\out\file.txt

Запустити

Старт операції

Тип:

Пошук

Пошук

Копіювання

Переміщення

Видалення

Старт операції

Тип:

Пошук

Маска імені:

report

Розширення:

pdf

Джерело (повний шлях):

C:\ztest_folder\wwwwwww

Ціль (для copy/move):

C:\ztest_folder\wwwwwww2

Запустити

Операція bb840d15-edef-4dee-8c0b-73c09d212849

Стан: **COMPLETED**

Повідомлення: Завершено

Прогрес: **100%**



Висновок

У ході роботи я реалізувала керування довготривалими операціями файлового менеджера за допомогою шаблону State. Було створено окремі стани для підготовки, виконання, завершення, скасування та обробки помилок. Завдяки цьому логіка виконання операцій (пошук, копіювання, переміщення та видалення) стала чітко структурованою, легко розширюваною та незалежною від контролерів. Використання контексту дозволило коректно обробляти прогрес, повідомлення та відображати актуальний статус у веб-інтерфейсі.

Питання до лабораторної роботи

1. Що таке шаблон проєктування?

Шаблон проєктування — це готове, перевірене рішення типових проблем проєктування програмного забезпечення, яке описує взаємодію класів і об'єктів без прив'язки до конкретної мови.

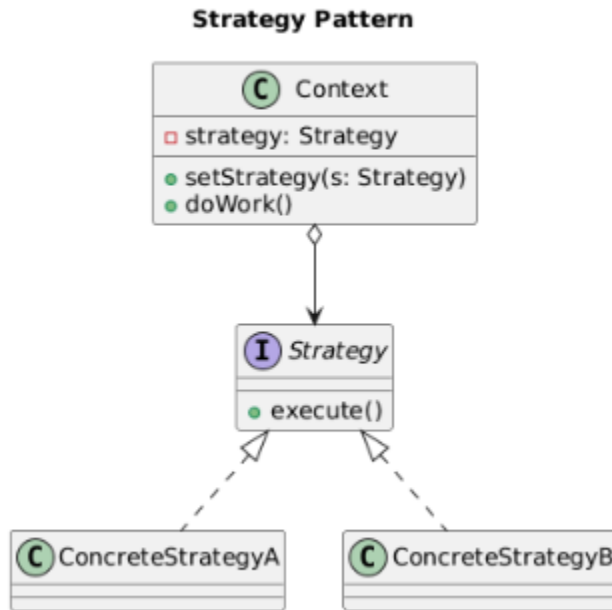
2. Навіщо використовувати шаблони проєктування?

Вони допомагають уникати помилок, підвищують гнучкість, повторне використання коду й полегшують підтримку.

3. Яке призначення шаблону «Стратегія»?

Шаблон «Стратегія» дозволяє інкапсулювати різні алгоритми в окремі класи й легко змінювати їх під час виконання програми, не змінюючи клієнтський код.

4. Нарисуйте структуру шаблону «Стратегія».



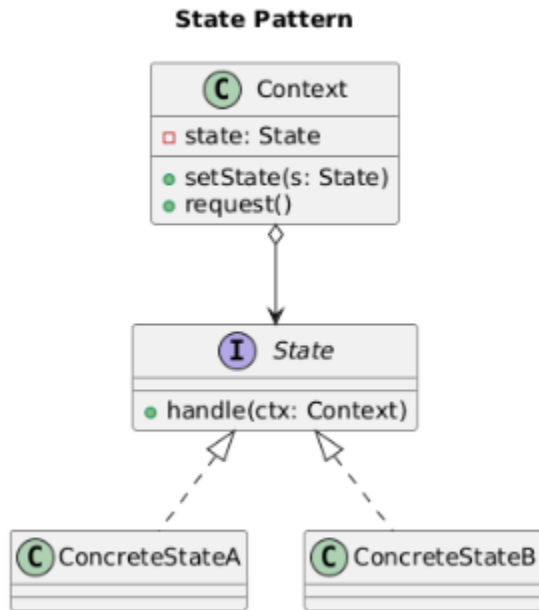
5. Які класи входять в шаблон «Стратегія», та яка між ними взаємодія?

- **Context** — використовує об'єкт стратегії.
 - **Strategy (інтерфейс)** — описує спільний метод алгоритму.
 - **ConcreteStrategyA / B** — реалізують різні варіанти алгоритмів.
- Context має посилання на Strategy і викликає її метод для виконання потрібної логіки.

6. Яке призначення шаблону «Стан»?

Шаблон «Стан» дозволяє об'єкту змінювати свою поведінку при зміні внутрішнього стану, тобто кожен стан реалізує власну поведінку.

7. Нарисуйте структуру шаблону «Стан».



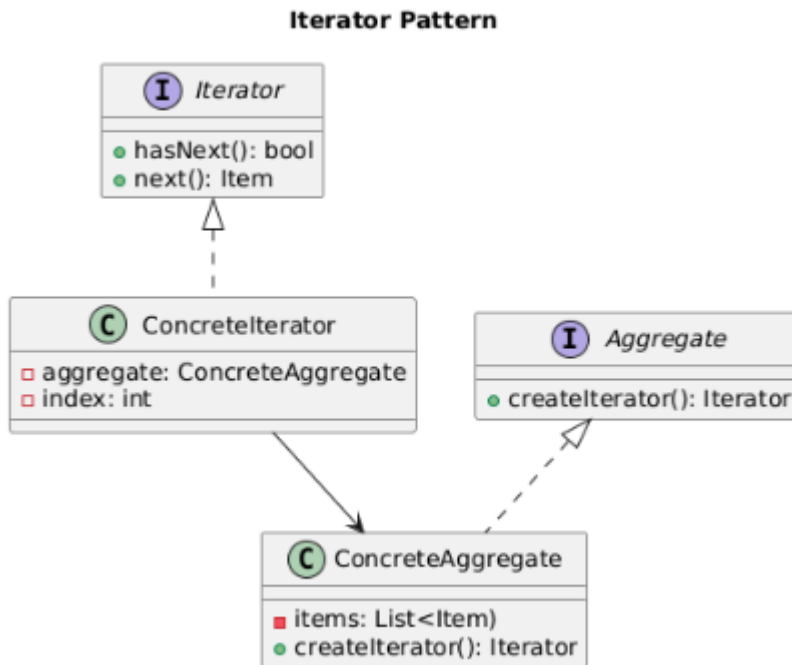
8. Які класи входять в шаблон «Стан», та яка між ними взаємодія?

- **Context** — зберігає поточний об'єкт стану.
 - **State (інтерфейс)** — оголошує методи для конкретних станів.
 - **ConcreteStateA / B** — реалізують різну поведінку для кожного стану.
- Context делегує виконання методів поточному об'єкту стану.

9. Яке призначення шаблону «Ітератор»?

Шаблон «Ітератор» забезпечує зручний і уніфікований спосіб послідовного доступу до елементів колекції без розкриття її внутрішньої структури.

10. Нарисуйте структуру шаблону «Ітератор».



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

- **Iterator (інтерфейс)** — визначає методи для обходу (`next()`, `hasNext()`).
 - **ConcreteIterator** — реалізує логіку обходу.
 - **Aggregate (інтерфейс)** — створює ітератор.
 - **ConcreteAggregate** — конкретна колекція, що повертає ітератор.
- Iterator взаємодіє з колекцією, обходячи її елементи.

12. В чому полягає ідея шаблону «Одинак»?

Шаблон «Одинак» (Singleton) гарантує, що клас має лише один екземпляр і надає глобальну точку доступу до нього.

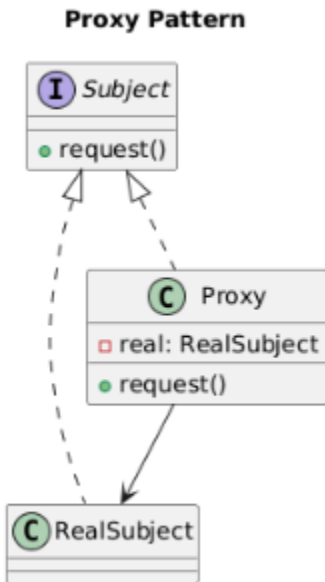
13. Чому шаблон «Одинак» вважають «анти-шаблоном»?

Бо він створює приховану глобальну залежність, ускладнює тестування, порушує принцип інкапсуляції та може викликати проблеми в багатопотокових програмах.

14. Яке призначення шаблону «Проксі»?

Шаблон «Проксі» використовується для контролю доступу до іншого об'єкта: він може додавати кешування, безпеку, логування чи віддалений виклик.

15. Нарисуйте структуру шаблону «Проксі».



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

- **Subject (інтерфейс)** — описує загальні методи.
- **RealSubject** — реальний об'єкт, що виконує основну роботу.
- **Proxy** — має посилання на `RealSubject` і контролює звернення до нього. Проксі вирішує, коли викликати `RealSubject` і може додавати додаткову поведінку.