

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 патернів проектування»

Варіант: E-mail клієнт

Виконав:

студент групи ІА-32

Карповець Роман

Київ - 2025

Тема: Вступ до паттернів проектування.

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

Тема проєкту: E-mail клієнт

Опис: Поштовий клієнт повинен нагадувати функціонал поштових програм Mozilla Thunderbird, The Bat і т.д. Він повинен сприймати і коректно обробляти pop3/smtp/imap протоколи, мати функції автонастройки основних поштових провайдерів для України (gmail, ukr.net, i.ua), розділяти повідомлення на папки/категорії/важливість, зберігати чернетки незавершених повідомлень, прикріплювати і обробляти прикріплені файли.

Зміст

Теоретичні відомості	3
Завдання.	6
Хід роботи.	7
Висновок.	8

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, а також рекомендації по застосуванню цього рішення в різних ситуаціях. 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.

«**Singleton**» (Одинак) являє собою клас в термінах ООП, який може мати не більше одного об'єкта (звідси і назва «одинак»). Насправді, кількість об'єктів можна задати (тобто не можна створити більш n об'єктів даного класу). Даний об'єкт найчастіше зберігається як статичне поле в самому класі.

Переваги та недоліки: Однак слід зазначити, що в даний час патерн «Одинак» багато хто вважає т.зв. «анти-шаблоном», тобто поганою практикою проєктування. Це пов'язано з тим, що «одинаки» представляють собою глобальні дані (як глобальна змінна), що мають стан. Стан глобальних об'єктів важко

відслідковувати і підтримувати коректно; також глобальні об'єкти важко тестуються і вносять складність в програмний код (у всіх ділянках коду виклик в одне єдине місце з «одинаком»; при зміні підходу доведеться змінювати масу коду).

При цьому реалізація контролю доступу можлива за допомогою статичних змінних, замикань, мютексів та інших спеціальних структур.

- + Гарантує наявність єдиного екземпляра класу.
- + Надає до нього глобальну точку доступу.
- Порушує принцип єдиної відповідальності класу.
- Маскує поганий дизайн.

Призначення: «Iterator» (Ітератор) являє собою шаблон реалізації об'єкта доступу до набору (колекції, агрегату) елементів без розкриття внутрішніх механізмів реалізації. Ітератор виносить функціональність перебору колекції елементів з самої колекції, таким чином досягається розподіл обов'язків: колекція відповідає за зберігання даних, ітератор – за прохід по колекції.

При цьому алгоритм ітератора може змінюватися – при необхідності пройти в зворотньому порядку використовується інший ітератор. Можливо також написання такого ітератора, який проходить список спочатку по парних позиціях (2,4,6-й елементи і т.д.), потім по непарних. Тобто, шаблон ітератор дозволяє реалізовувати різноманітні способи проходження по колекції незалежно від виду і способу представлення даних в колекції.

Переваги та недоліки: Цей шаблон дозволяє уніфікувати операції проходження по наборам об'єктів для всіх наборів. Тобто, незалежно від реалізації (масив, зв'язаний список, незв'язаний список, дерево та ін.), кожен з наборів може використовувати будь-який з реалізованих ітераторів.

- + Дозволяє реалізувати різні способи обходу структури даних.
- + Спрощує класи зберігання даних.
- Не виправданий, якщо можна обійтися простим циклом.

Призначення: «Proxy» (Проксі) – об'єкти є об'єктами-заглушками або двійниками/замінниками для об'єктів конкретного типу. Зазвичай, проксі об'єкти вносять додатковий функціонал або спрощують взаємодію з реальними об'єктами.

Проксі об'єкти використовувалися в більш ранніх версіях інтернет-браузерів, наприклад, для відображення картинки: поки картинка завантажується, користувачеві відображається «заглушка» картинки.

Переваги та недоліки:

- + Легкість впровадження проміжного рівня без переробки клієнтського коду.
- + Додаткові можливості по керуванню життєвим циклом об'єкту.
- Існує ризик падіння швидкості роботи через впровадження додаткових операцій.
- Існує ризик неадекватної заміни відповіді клієнтському коду

Призначення: Шаблон «State» (Стан) дозволяє змінювати логіку роботи об'єктів у випадку зміни їх внутрішнього стану. Наприклад, відсоток нарахованих на картковий рахунок грошей залежить від стану картки: Visa Electron, Classic, Platinum і т.д. Або обсяг послуг, які надані хостинг компанією, змінюється в залежності від обраного тарифного плану (стану членства – бронзовий, срібний або золотий клієнт). Реалізація даного шаблону полягає в наступному: пов'язані зі станом поля, властивості, методи і дії виносяться в окремий загальний інтерфейс (State); кожен стан являє собою окремий клас (ConcreteStateA, ConcreteStateB), які реалізують загальний інтерфейс.

Об'єкти, що мають стан (Context), при зміні стану просто записують новий об'єкт в поле state, що призводить до повної зміни поведінки об'єкта.

Це дозволяє легко додавати в майбутньому і обробляти нові стани, відокремлювати залежні від стану елементи об'єкта в інших об'єктах, і відкрито проводити заміну стану (що має сенс у багатьох випадках).

Переваги та недоліки:

- + Код специфічний для окремого стану реалізується в класі стану.
- + Класи та об'єкти станів можна використовувати з різними контекстами, за рахунок чого збільшується гнучкість системи.
- + Код контексту простіше читати, тому що вся залежна від станів логіка винесена в інші класи.

- + Відносно легко додавати нові стани, головне правильно змінити переходи між станами.
- Клас контекст стає складніше через ускладнений механізм переключення станів.

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.

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).

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.

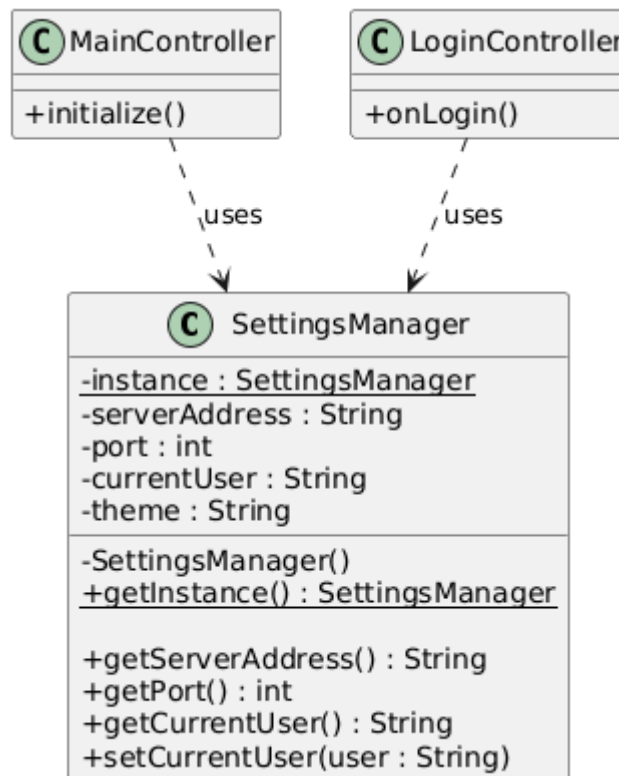
Завдання

- 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

Реалізація патерну Singleton



Рисю 1. Діаграма класів з реалізацією патерну Singleton.

Фрагмент коду реалізації патерну Singleton

```

package com.emailclient.config;

/**
 * Патерн: Singleton.
 * Зберігає глобальні налаштування програми та сесію користувача.
 */
public class SettingsManager {

    // 1. Приватне статичне поле для зберігання єдиного екземпляра
    private static SettingsManager instance;

    // Поля налаштувань
    private String serverAddress;
    private int serverPort;
    private String currentUserEmail;
    private String appTheme;
  
```

```

// 2. Приватний конструктор (забороняє new SettingsManager())
private SettingsManager() {
    // Імітація завантаження налаштувань з файлу
    this.serverAddress = "imap.ukr.net";
    this.serverPort = 993;
    this.appTheme = "Light";
    System.out.println("Налаштування завантажено.");
}

// 3. Публічний метод доступу
public static synchronized SettingsManager getInstance() {
    if (instance == null) {
        instance = new SettingsManager();
    }
    return instance;
}

// Гетери та сетери
public String getServerAddress() { return serverAddress; }
public int getServerPort() { return serverPort; }

public String getCurrentUserEmail() { return currentUserEmail; }
public void setCurrentUserEmail(String email) { this.currentUserEmail = email; }

public String getAppTheme() { return appTheme; }
public void setAppTheme(String theme) { this.appTheme = theme; }
}

```

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