

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Навчально-науковий інститут комп'ютерного моделювання,
прикладної фізики та математики

Звіт

З лабораторної роботи №1

З дисципліни: “ Ефективність та якість архітектурних рішень
інформаційних систем ”

На тему:

«Паттерн Одинак»

Виконав:

студент групи ІКМ-М225В

Загорулько Віталій Олегович

Харків 2025

Вступ

Одинак (Singleton), також відомий як Singleton, є одним із найбільш фундаментальних породжувальних патернів проєктування. Його сутність полягає в тому, що він гарантує, що клас має лише один екземпляр, та надає глобальну точку доступу до нього.

Патерн Одинак є унікальним, оскільки він вирішує відразу дві проблеми, при цьому, як зазначається, порушуючи принцип єдиного обов'язку класу:

1. Гарантія єдиного екземпляра класу: Найчастіше це необхідно для доступу до якогось спільного ресурсу, такого як налаштування конфігурації, журнал подій або, як у нашому випадку, менеджер підключень. Звичайна поведінка конструктора полягає в тому, що він завжди повертає новий об'єкт. Однак, у ситуаціях, коли потрібно отримати вже існуючий об'єкт замість створення нового, звичайний конструктор не може цього забезпечити. Одинак бере на себе контроль над цим процесом.

2. Надання глобальної точки доступу: Забезпечує контрольований та централізований спосіб доступу до єдиного об'єкта, що є безпечнішим, ніж використання глобальних змінних. Глобальні змінні не захищені від перезапису, тоді як Одинак контролює життєвий цикл об'єкта та гарантує, що жоден інший код не зможе замінити створений екземпляр.

Таким чином, у даній лабораторній роботі Одинак застосовується для створення єдиного об'єкта `StorageManager`, який повинен централізовано керувати конфігурацією та підключеннями сховищ, що відповідає вимозі гарантувати існування єдиного об'єкта в системі.

Реалізація Завдання та Структура Проєкту

Завдання полягало у реалізації системи управління файлами користувача, де кожен користувач повинен підключатися до одного з кількох сховищ (наприклад, локальний диск або Amazon S3), причому список сховищ може зростати, а обране сховище для кожного користувача задається окремо. Для забезпечення централізованого управління доступними сховищами та гарантії єдиного конфігураційного об'єкта було обрано реалізацію Одинака в класі `StorageManager`.

Проєктна структура складається з трьох основних компонентів:

1. Інтерфейс `IStorage`: визначає загальний контракт для всіх сховищ (`uploadFile`, `downloadFile`, `deleteFile`).
2. Конкретні класи сховищ (`LocalDiskStorage`, `AmazonS3Storage`): реалізують `IStorage` і містять специфічну логіку роботи з відповідними ресурсами.
3. Клас-Одинак `StorageManager`: виконує функцію централізованого диспетчера, зберігаючи зв'язок між ідентифікатором користувача та призначеним йому об'єктом сховища.

Детальна Технічна Реалізація Одинака

Для коректної реалізації Одинака в Java було дотримано класичних кроків:

1. Приватний конструктор: Забороняє пряме створення екземплярів через `new StorageManager()`.
2. Приватне статичне поле: `private static volatile StorageManager instance;` зберігає єдиний екземпляр.
3. Публічний статичний метод `getInstance()`: Слугує єдиною точкою доступу та реалізує ліниву ініціалізацію (створення об'єкта під час першого виклику методу).

Оскільки додаток потенційно працюватиме в багатопотоковому середовищі, критично важливо було забезпечити багатопотокову безпеку. Цього досягнуто за допомогою патерну `Double-Checked Locking`.

Цей механізм забезпечує як безпеку, так і високу продуктивність, оскільки блокування використовується лише під час первинної ініціалізації.

Аналіз Патерну: Переваги та Недоліки

Реалізація патерна Одинак надає значні переваги:

- Гарантія єдиного екземпляра та глобальна точка доступу.
- Відкладена ініціалізація, що економить ресурси системи.

Однак, Одинак має й деякі недоліки:

- Порушення принципу єдиного обов'язку класу.
- Маскування поганого дизайну та ускладнення тестування.
- Вимагає особливої уваги до проблем багатопоточності.

Висновок

В ході роботи було засвоєно теоретичні знання про природу Одинака як породжувального патерну та його критичну роль у забезпеченні унікальності об'єкта. Практична реалізація на мові Java із застосуванням класу `StorageManager` продемонструвала всі ключові елементи: приватний конструктор, статичний метод доступу (`getInstance`) та ліниву ініціалізацію.

Особлива увага була приділена багатопотоковій безпеці. Впровадження механізму `Double-Checked Locking` було ключовим технічним рішенням, яке дозволило гарантувати, що єдиний екземпляр `StorageManager` буде коректно створений навіть при одночасному доступі з кількох потоків, що є необхідною умовою для розробки стійких сучасних систем.

Крім того, ця робота дозволила оцінити Одинак не лише з точки зору його переваг (глобальний доступ, єдиний екземпляр), але й з точки зору його архітектурних ризиків, зокрема, порушення принципу єдиного обов'язку та потенційного ускладнення модульного тестування.

Клас `StorageManager` успішно функціонує як Одинак, ефективно вирішуючи завдання централізованого управління доступом користувачів до різномірних сховищ. Здобуті навички щодо забезпечення багатопотокової безпеки та застосування Одинака як механізму контролю над життєвим циклом об'єкта є критично важливими для подальшої роботи у сфері об'єктно-орієнтованого проектування.