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

Лабораторна робота №1

Тема**:** **«**Патер проектування Одинак»

Дисципліна: «Ефективність та якість архітектурних рішень інформаційних систем»

Виконав: студент групи ІКМ-224а
Нестеренко Владислав Валентинович

Перевірив:
Хорошун Андрій Сергійович

Харків - 2024р.

Мета: Здобути навички з реалізацією патерна проектування Одинак.

Завдання:

1. Ознайомитися з патерном Одинак.
2. Виконати всі пункти лабораторної роботи.
3. Надати звіт про виконану роботу.

Хід роботи:

1. Реалізувати систему управління файлами користувача
2. Користувач має підключатися до одного із сховищ з списку сховищ.

   На початку проекту це:

   - локальний диск системи
   - сховище Amazon S3

3. Список доступних сховищ може бути збільшено у майбутньому
4. Для кожного користувача обране сховище задається окремо.

Необхідно створити структуру класів та методів яка буде демонструвати реалізацію патерну Одинак і буде вирішувати описанне завдання.

У рамках виконання лабораторної роботи не потрібно описувати деталі реалізації самих методів! Достатньо вказати сам метод та параметри який він приймає та повертає.

Виконання:

Патерн Одинак:

Одинак — це породжувальний патерн проектування, який гарантує, що клас має лише один екземпляр, та надає глобальну точку доступу до нього.

Кроки реалізації:

1. Додайте до класу приватне статичне поле, котре міститиме одиночний об’єкт.
2. Оголосіть статичний створюючий метод, що використовуватиметься для отримання Одинака.
3. Додайте «ліниву ініціалізацію» (створення об’єкта під час першого виклику методу) до створюючого методу одинака.
4. Зробіть конструктор класу приватним.
5. У клієнтському коді замініть прямі виклики конструктора одинака на виклики його створюючого методу.

Реалізація на Java:

Клас User представляє користувача, який може взаємодіяти з різними сховищами:

```
public class User {
    private String name;
    private Storage storage;

    public User(String name) {
        this.name = name;
    }

    public void setStorage(Storage storage) {
        this.storage = storage;
        this.storage.connect();
    }

    public void uploadFile(String fileName, byte[] data) {
        this.storage.uploadFile(fileName, data);
    }

    public byte[] downloadFile(String fileName) {
        return this.storage.downloadFile(fileName);
    }
}
```

Абстрактний клас Storage визначає загальний інтерфейс для всіх типів сховищ.

```
public abstract class Storage {
    public abstract void connect();
    public abstract void uploadFile(String fileName, byte[] data);
    public abstract byte[] downloadFile(String fileName);
}
```

Клас LocalDiskStorage з патерном Одинак - реалізує сховище на локальному диску.

```
public class LocalDiskStorage extends Storage {

// Статичне приватне поле для збереження єдиного екземпляра класу
    private static LocalDiskStorage instance;


// Приватний конструктор запобігає створенню екземплярів класу ззовні
    private LocalDiskStorage() {}


// Статичний метод, який повертає єдиний екземпляр класу
    public static synchronized LocalDiskStorage getInstance() {
        if (instance == null) {
            instance = new LocalDiskStorage();
        }
        return instance;
    }

    @Override
    public void connect() {
    }

    @Override
    public void uploadFile(String fileName, byte[] data) {
    }

    @Override
    public byte[] downloadFile(String fileName) {
        return null;
    }
}
```

Клас AmazonS3 застосовується патерн Одинак - реалізує сховище на Amazon S3.

```
public class AmazonS3 extends Storage {

// Статичне приватне поле для збереження єдиного екземпляра класу
    private static AmazonS3 instance;

// Приватний конструктор запобігає створенню екземплярів класу ззовні
    private AmazonS3() {}

// Статичний метод, який повертає єдиний екземпляр класу
    public static synchronized AmazonS3 getInstance() {
        if (instance == null) {
            instance = new AmazonS3();
        }
        return instance;
    }

    @Override
    public void connect() {
    }

    @Override
    public void uploadFile(String fileName, byte[] data) {
    }

    @Override
    public byte[] downloadFile(String fileName) {
        return null;
    }
}
```

Клас Main - використання системи.

```
public class Main {
    public static void main(String[] args) {
        User user = new User("Vlad");

        // Використання локального диска
        Storage localDisk = LocalDiskStorage.getInstance();
        user.setStorage(localDisk);
        user.uploadFile("file.txt", new byte[] {1, 2, 3});
        byte[] data = user.downloadFile("file.txt");

        // Використання Amazon S3
        Storage amazonS3 = AmazonS3.getInstance();
        user.setStorage(amazonS3);
        user.uploadFile("file.txt", new byte[] {1, 2, 3});
        data = user.downloadFile("file.txt");
    }
}
```

Висновки

Під час виконання даної лабораторної роботи я ознайомився з патерном Одинак. А також створили систему управління файлами користувача з використанням даного патерну без використання вбудованих інструментів або сторонніх бібліотек.