Report

Zhaksybay Akhmet

Se-2416

Link to GitHub: https://github.com/AkhmetZh/4SDP.git

Introduction

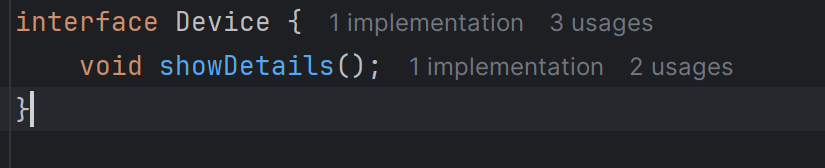
This report presents a Java implementation of the Bridge Pattern applied to the product mobile devices. The main idea of the Bridge Pattern is to decouple an abstraction from its implementation so that both can evolve independently. In this project, the abstraction is the concept of a Device, while the implementation is represented by different types of operating systems (AndroidOS and IOS).

In addition, the code demonstrates some principles from Clean Code, such as meaningful naming, separation of concerns, and immutability.

Code Explanation

1. Abstraction: Device

The Device interface defines the general behavior of any device, represented here by the showDetails() method.

2. Implementation Hierarchy: OperatingSystem

The OperatingSystem interface represents the bridge’s implementation side. It defines two methods: getName() and getVersion(). Concrete implementations include AndroidOS and IOS.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.This allows us to easily extend the system by adding new operating systems in the future (e.g., HarmonyOS, Windows Mobile).

3. Refined Abstraction: MobileDevice

The MobileDevice class represents a specific type of device. It contains properties such as brand, model, price, storage, and battery capacity. Importantly, it holds a reference to an OperatingSystem, which demonstrates the Bridge Pattern by connecting the abstraction (device) with its implementation (OS).

Изображение выглядит как текст, снимок экрана, программное обеспечение

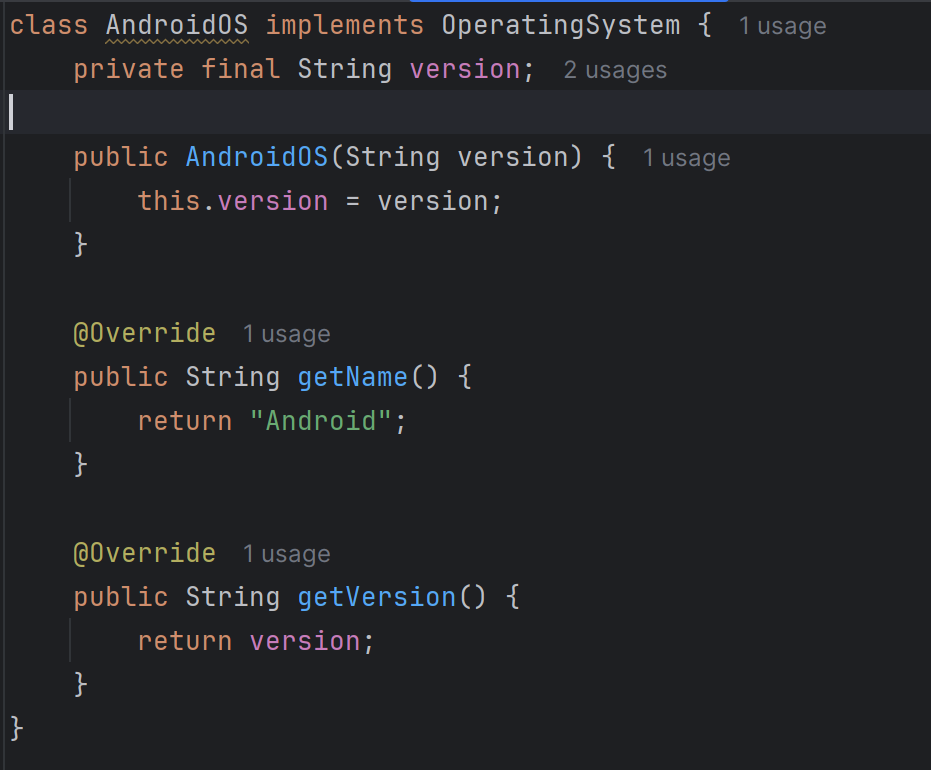
Содержимое, созданное искусственным интеллектом, может быть неверным.The showDetails() method prints out the specifications, including the operating system used.

4. Main Program

In the Main class, two devices are created: a Samsung phone running Android and an iPhone running iOS. Each device uses the same abstraction (Device) but can work with different OS implementations.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Содержимое, созданное искусственным интеллектом, может быть неверным.



Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Application of the Bridge Pattern

Abstraction → Device

Refined Abstraction → MobileDevice

Implementor → OperatingSystem

Concrete Implementors → AndroidOS, IOS

This separation allows us to add new devices (e.g., Tablet, SmartWatch) or new operating systems without modifying existing code. Thus, the Bridge Pattern increases flexibility and maintainability.

Conclusion

This project successfully demonstrates the Bridge Pattern in Java by decoupling devices from their operating systems. The implementation also reflects several Clean Code principles, resulting in readable, maintainable, and extensible code. The design makes it easy to expand with new devices or operating systems in the future, which is the main benefit of applying the Bridge Pattern.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.