Java Design Patterns

Bridge

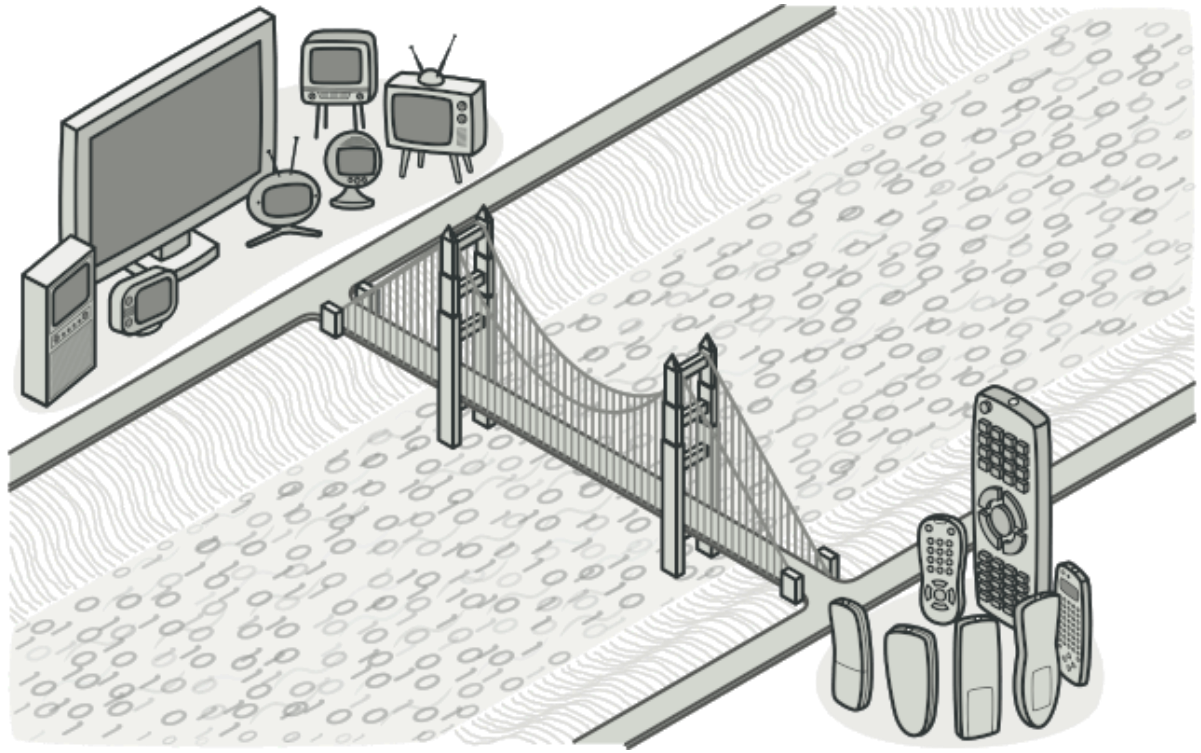
Тема

Bridge

Суть паттерна

Мост

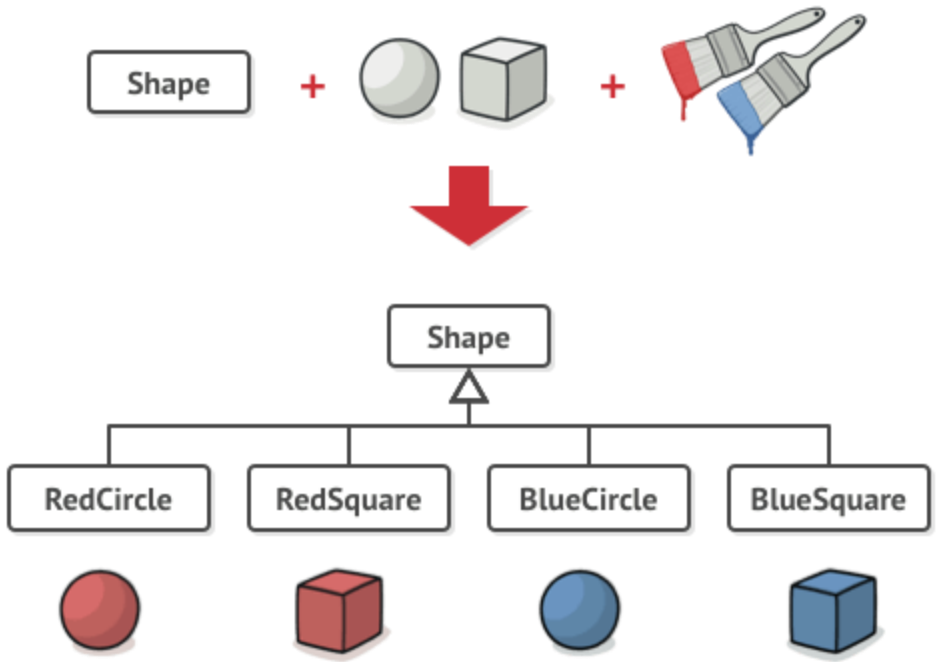
Мост — это структурный паттерн проектирования, который разделяет один или несколько классов на две отдельные иерархии — абстракцию и реализацию, позволяя изменять их независимо друг от друга.



Проблема

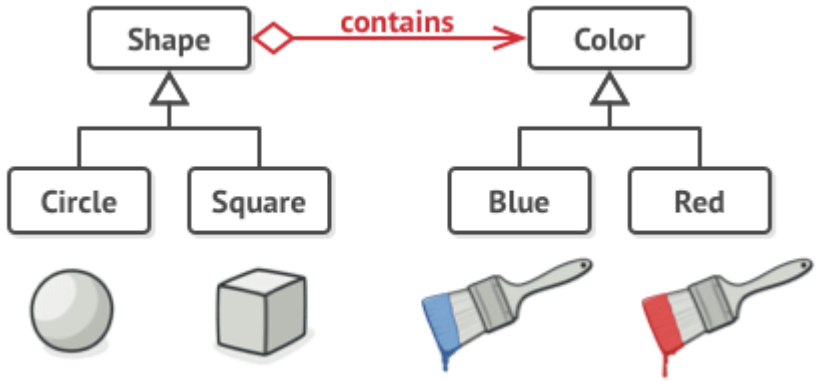
Постановка задачи

У вас есть класс геометрических Фигур, который имеет подклассы Круг и Квадрат. Вы хотите расширить иерархию фигур по цвету, то есть иметь Красные и Синие фигуры. Но чтобы всё это объединить, вам придётся создать 4 комбинации подклассов вроде Синие Круги и Красные Квадраты.



Решение задачи

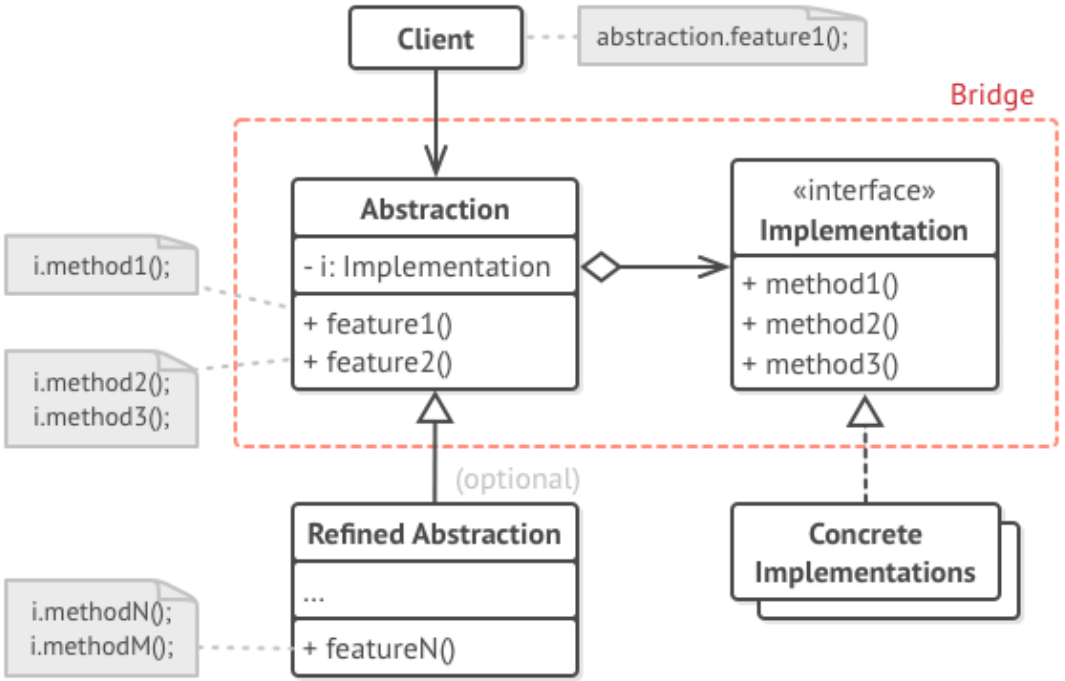
Корень проблемы заключается в том, что мы пытаемся расширить классы сразу в двух независимых плоскостях — по виду и по цвету. Именно это приводит к разрастанию дерева классов.



Структура

Структура паттерна

1. Абстракция содержит управляющую логику. Код абстракции делегирует реальную работу связанному объекту реализации.
2. Реализация задаёт общий интерфейс для всех реализаций. Все методы, которые здесь описаны, будут доступны из класса абстракции и его подклассов.
3. Конкретные Реализации содержат платформо-зависимый код.
4. Расширенные Абстракции содержат различные вариации управляющей логики. Как и родитель, работает с реализациями только через общий интерфейс реализации.
5. Клиент работает только с объектами абстракции. Не считая первичного связывания абстракции с одной из реализаций, клиентский код не имеет прямого доступа к объектам реализации.



Применимость

Применение паттерна

1. Когда вы хотите разделить монолитный класс, который содержит несколько различных реализаций какой-то функциональности (например, может работать с разными системами баз данных).
2. Когда класс нужно расширять в двух независимых плоскостях.
3. Когда вы хотите, чтобы реализацию можно было бы изменять во время выполнения программы.

Шаги реализации

Алгоритм реализации паттерна

1. Определите, существует ли в ваших классах два непересекающихся измерения. Это может быть функциональность/платформа, предметная-область/инфраструктура, фронт-энд/бэк-энд или интерфейс/реализация.
2. Продумайте, какие операции будут нужны клиентам и опишите их в базовом классе *абстракции*.
3. Определите поведения доступные на всех платформах и выделите из них ту часть, которая будет нужная абстракции. На основании этого опишите общий интерфейс *реализации*.
4. Для каждой платформы создайте свой класс конкретной реализации. Все они должны следовать общему интерфейсу, который мы выделили перед этим.
5. Добавьте в класс абстракции ссылку на объект реализации. Реализуйте методы абстракции, делегируя основную работу связанному объекту реализации.
6. Если у вас есть несколько вариаций абстракции, создайте для каждой из них свой подкласс.
7. Клиент должен подать объект реализации в конструктор абстракции, чтобы связать их воедино. После этого он может свободно использовать объект абстракции, забыв о реализации.

Плюсы и недостатки

Плюсы:

- Позволяет строить платформо-независимые программы.
- Скрывает лишние или опасные детали реализации от клиентского кода.
- Реализует *принцип открытости/закрытости*.

Минусы:

- Усложняет код программы за счёт дополнительных классов.

Отношение с другими паттернами

- Мост проектируют заранее, чтобы развивать большие части приложения отдельно друг от друга. Адаптер применяется постфактум, чтобы заставить несовместимые классы работать вместе.
- Мост, Стратегия и Состояние (а также слегка и Адаптер) имеют схожие структуры классов — все они построены на принципе «композиции», то есть делегирования работы другим объектам. Тем не менее, они отличаются тем, что решают разные проблемы. Помните, что паттерны — это не только рецепт построения кода определённым образом, но и описание проблем, которые привели к данному решению.
- Абстрактная фабрика может работать совместно с Мостом. Это особенно полезно, если у вас есть абстракции, которые могут работать только с некоторыми из реализаций. В этом случае фабрика будет определять типы создаваемых абстракций и реализаций.
- Паттерн Строитель может быть построен в виде Моста: *директор* будет играть роль абстракции, а *строители* — реализации.

Информационный видеосервис для разработчиков программного обеспечения

ITVDN

IT VIDEO DEVELOPERS NETWORK

TestProvider PROMETRIC TEST CENTER CyberBionic systematics Microsoft Partner Silver Learning Windows Azure Cloud Partner EBA European Business Association