(Слайд 1)

В программировании очень часто приходится использовать что-то, что уже было создано другими разработчиками, что столкнулись с похожей проблемой. Сегодня же, речь пойдёт о такой вещи, как паттерн мост.

(Слайд 2)

Мост (Bridge) – это структурный паттерн проектирования, который разделяет один или несколько классов на две отдельные составляющие — абстракцию и реализацию.

Взаимодействие между слоями осуществляется через композицию в результате чего слои можно изменять независимо друг от друга.

Слой реализации (она же платформа) – слой, реализующий бизнес-логику (логику, направленную на решение поставленной задачи).

Слой абстракции (он же интерфейс) – данный слой является образным так как он не делает работу самостоятельно, а делегирует ее слою реализации. Необходим для работы с пользователем.

(Слайд 3)

Abstraction – стандартный абстрактный слой. Данный слой содержит в себе управляющую логику. Код абстракции делегирует работу объекту, который реализует интерфейс Implementation.

Implementation – интерфейс для слоя реализации. Необходим для выделения общих методов для всех платформ. Именно через данные методы слой абстракции будет взаимодействовать со слоем реализации. Интерфейсы абстракции и реализации могут как совпадать, так и быть совершенно разными. Но обычно в реализации прописаны базовые элементарные операции, на которых строятся более сложные операции абстракции.

Concrete Implementations – реализации платформы. Реализуют общий интерфейс слоя реализации, содержат код который зависит от платформы

Refined Abstraction – расширенные абстракции, содержат различные вариации управляющей логики. Как и родитель, работает с реализациями только через общий интерфейс. Может не быть в коде если нет нужды в нескольких реализациях абстракции

(Слайд 4)

Паттерн мост можно применять при следующих ситуациях:

* Когда необходимо разделить монолитный класс содержащий в себе несколько реализаций какой-либо функциональности
* Когда необходимо расширять класс в двух независимых плоскостях
* Когда необходимо изменить слой реализации не трогая интерфейс и наоборот

(Слайд 5)

У паттерна мост можно выделить следующие шаги построения:

1. Определить, существует ли в классах два непересекающихся направлений (Например: фронт-энд и бэк-энд логика)
2. Выделить общие операции платформ. На основе данных операций создать общий интерфейс для слоя реализации
3. Для каждой платформы создать класс, реализующий базовый интерфейс слоя реализации.

(Слайд 6)

1. Выделить общие операции, необходимые пользователям, и на их основе создать общий базовый класс для абстрактной части. Добавить в данный класс ссылку на объект реализации.
2. Если имеется несколько вариаций абстракции, то они должны наследоваться от базового класса для абстрактной части.

(Слайд 7)

Задача:

Давайте представим, что в один не очень хороший день перед вам поручили написать бота для скачивания видео с различных площадок.

Решение:

Получив задачу, вы как ответственный программист перед написанием кода решили изучить предметную область. В ходе изучения предметной области выяснилось, что в коде отвечающим за работу с видео есть 2 независимые части - часть, которая работает с социальными сетями и та, что работает с пользователем.

(Слайд 8)

В ходе свих размышлений вы поняли, что необходимо использовать такой паттерн как мост, где в качестве абстрактной части будет работа с пользователем, а в качестве слоя реализации – работа с соц. сетями. Продолжив анализ области по итогу вы построили следующию UML диаграмму:

На данной диаграмме вы отобразили общий интерфес IPlatform для слоя реализации, классы YouTube и TikTok реализующие выше описанный интерфейс. Так же вы отобразили класс BaseUser для работы с пользователем

(Слайд 9)

После реализации диаграммы вы приступили к написанию интерфейса IPlatform. В результате получилось следующее (подождать пока люди осмыслят происходящее)

(Слайд 10)

(Слайд 11)

После выделения общего интерфейса вы решили приступить к реализации платформ. В результате сего действия вы написали такие классы как TikTok и YouTube

(Слайд 12)

После реализации слоя реализации вы вспомнили что необходимо бы реализовать базовый класс для абстрактной части. Собственно этим вы занялись и в результате вы реализовали такой класс BaseUser.

(Слайд 13)

После реализации абстрактного слоя паттерн можно использовать в клиентском коде. А вот, собственно, и пример как можно использовать этот паттерн в клиентском коде.

(Слайд 14)

Реализовав работу в клиентском коде, вы с чувством выполненного долга вы реквестите свой код и сообщаете работодателю что работа выполнена.

Спустя какое-то время ваш бот становится популярным из-за чего вам дают новую задачу: добавить особых пользователей с расширенными возможностями. Вы принимаете задание и после небольших раздумий вы реализуете класс SuperUser в результате чего у вас получается следующее

(Слайд 15)

Подводя все вышесказанное о паттерне мост можно выделить следующие преимущества:

1. Возможность строить платформо-независимое ПО
2. Скрытие реализации платформы от клиентского кода
3. Реализует принцип открытости/закрытости
4. Возможность изменения одного слоя без обязятельных изменений в другом

Помимо плюсов можно выделить и следующие недостатки:

1. Усложнение кода из-за введения новых интерфейсов и классов

(Слайд 16)