РГПУ ИМ. А.И. ГЕРЦЕНА

PYTHON

Работу выполнила студентка ИКНиТО ИВТ: Шибаева Мария Дмитриевна
Преподаватель: Жуков Николай Николаевич

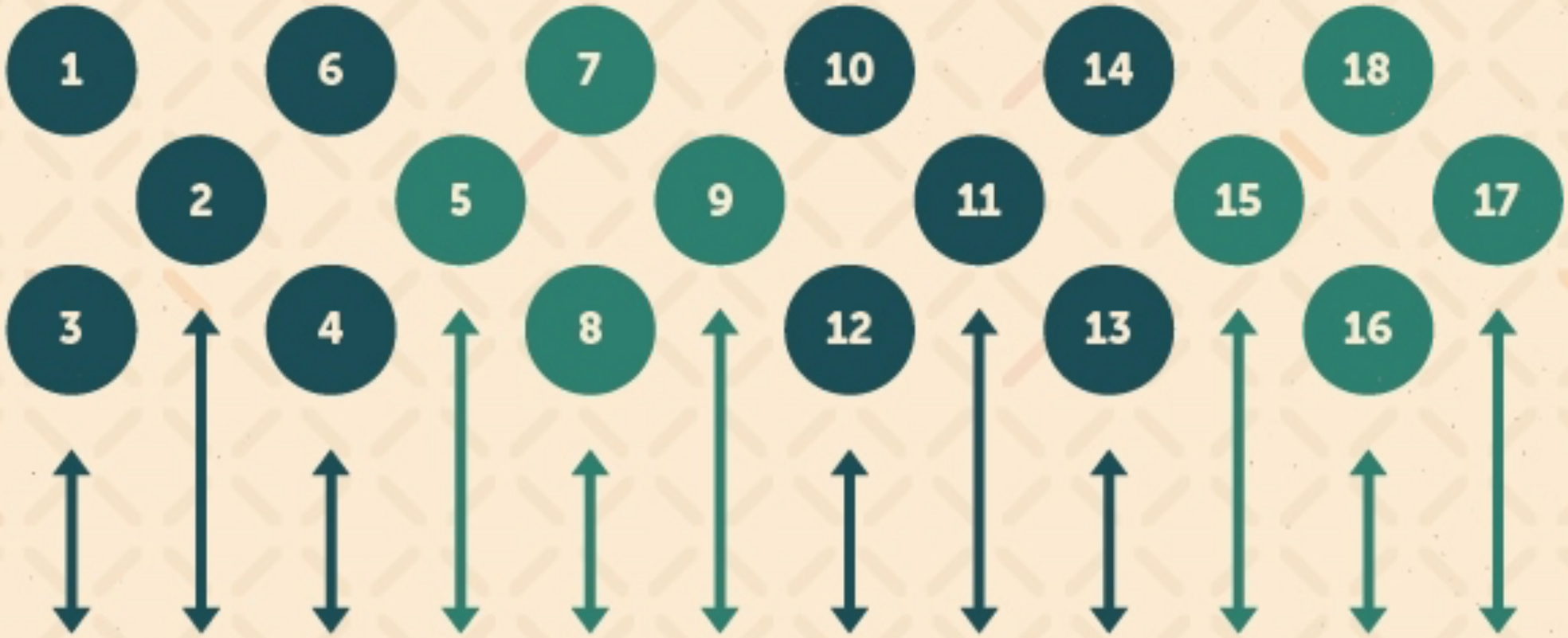
ШАБЛОН ФАСАД

2019 ГОД

ШАБЛОНЫ ПРОЕКТИРОВАНИЯ — ЭТО РУКОВОДСТВА ПО РЕШЕНИЮ ПОВТОРЯЮЩИХСЯ ПРОБЛЕМ.

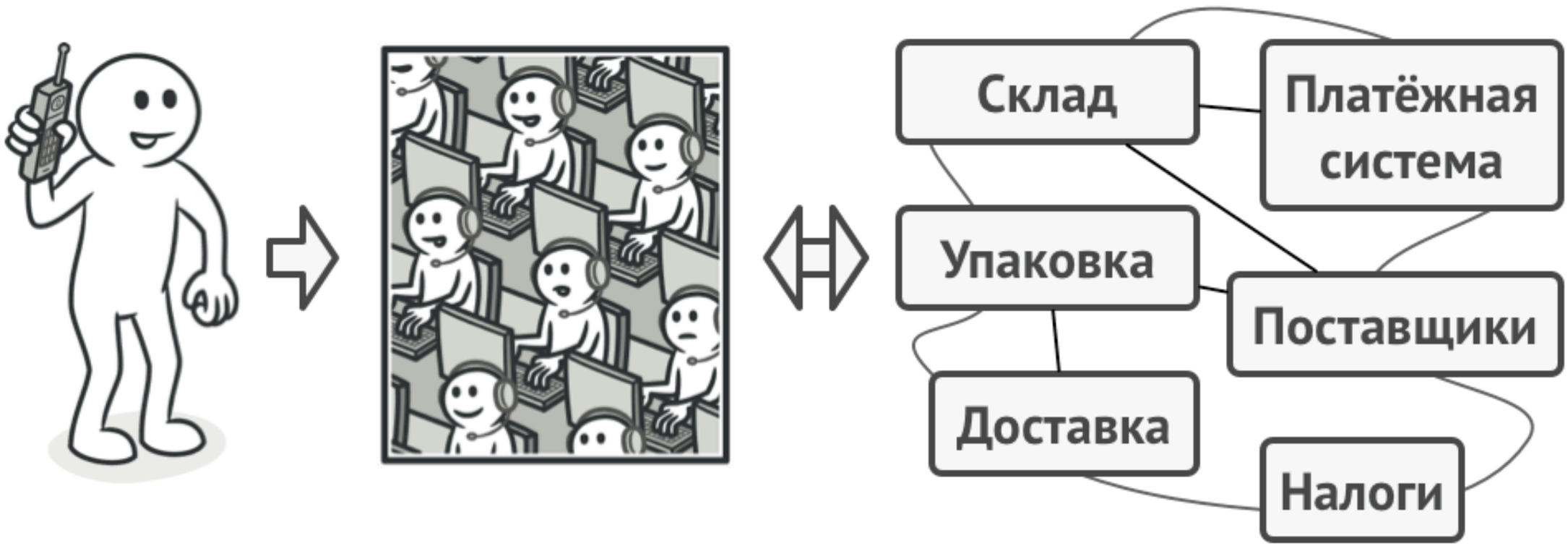
ФАСАД — СТРУКТУРНЫЙ ШАБЛОН ПРОЕКТИРОВАНИЯ, ПОЗВОЛЯЮЩИЙ СКРЫТЬ СЛОЖНОСТЬ СИСТЕМЫ ПУТЕМ СВЕДЕНИЯ ВСЕХ ВОЗМОЖНЫХ ВНЕШНИХ ВЫЗОВОВ К ОДНОМУ ОБЪЕКТУ, ДЕЛЕГИРУЮЩЕМУ ИХ СООТВЕТСТВУЮЩИМ ОБЪЕКТАМ СИСТЕМЫ.

*Структурные шаблоны – шаблоны проектирования, в которых рассматривается вопрос о том, как из классов и объектов образуются более крупные структуры. Т.е. занимаются объединением отдельных классов и объектов в сложные группы.



FACADE

CLIENT



Пример телефонного заказа.

ПРИМЕНИМОСТЬ

- Паттерн часто встречается в клиентских приложениях, написанных на Python, которые используют классы-фасады для упрощения работы со сложными библиотеки или API.

ПРИЗНАКИ ПРИМЕНЕНИЯ ПАТТЕРНА

- Фасад угадывается в классе, который имеет простой интерфейс, но делегирует основную часть работы другим классам. Чаще всего, фасады сами следят за жизненным циклом объектов сложной системы.

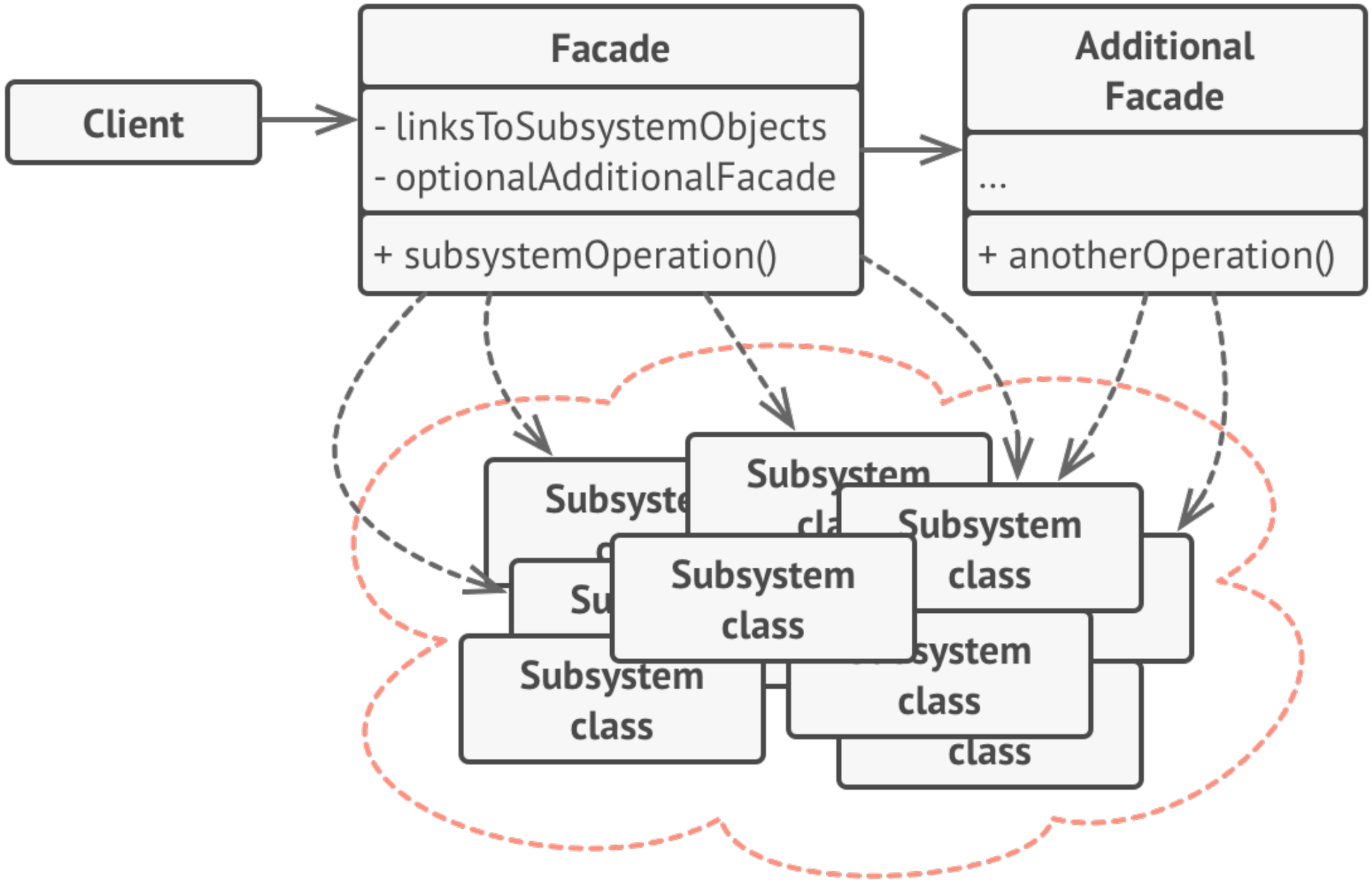
ПРОБЛЕМА

- Вашему коду приходится работать с большим количеством объектов некой сложной библиотеки или фреймворка. Вы должны самостоятельно инициализировать эти объекты, следить за правильным порядком зависимостей и так далее.
- В результате бизнес-логика ваших классов тесно переплетается с деталями реализации сторонних классов. Такой код довольно сложно понимать и поддерживать.

РЕШЕНИЕ

- Фасад – это простой интерфейс для работы со сложной подсистемой, содержащей множество классов. Фасад может иметь урезанный интерфейс, не имеющий 100% функциональности, которой можно достичь, используя сложную подсистему напрямую.
- Фасад полезен, если вы используете какую-то сложную библиотеку со множеством подвижных частей, но вам нужна только часть её возможностей.
- К примеру, программа, заливающая видео котиков в социальные сети, может использовать профессиональную библиотеку сжатия видео. Но все, что нужно клиентскому коду этой программы – простой метод #encode(filename, format)#. Создав класс с таким методом, вы реализуете свой первый фасад.

СТРУКТУРА


Client
Facade
- linksToSubsystemObjects
- optionalAdditionalFacade
+ subsystemOperation()
Additional Facade
...
+ anotherOperation()
Subsystem class
Subsystem class
Subsystem class
Subsystem class
Subsystem class
Subsystem class

ПРИЗНАКИ ПРИМЕНЕНИЯ ПАТТЕРНА

- Фасад угадывается в классе, который имеет простой интерфейс, но делегирует основную часть работы другим классам. Чаще всего, фасады сами следят за жизненным циклом объектов сложной системы.

1.py

```
class VideoFile
// ...
class OggCompressionCodec
// ...
class MPEG4CompressionCodec
// ...
class CodecFactory
// ...
class BitrateReader
// ...
class AudioMixer
// ...

class VideoConverter is
    method convert(filename, format):File is
        file = new VideoFile(filename)
        sourceCodec = new CodecFactory.extract(file)
        if (format == "mp4")
            destinationCodec = new MPEG4CompressionCodec()
        else
            destinationCodec = new OggCompressionCodec()
        buffer = BitrateReader.read(filename, sourceCodec)
        result = BitrateReader.convert(buffer, destinationCodec)
        result = (new AudioMixer()).fix(result)
        return new File(result)

class Application is
    method main() is
        convertor = new VideoConverter()
        mp4 = convertor.convert("youtubevideo.ogg", "mp4")
        mp4.save()

```

ОТНОШЕНИЯ С ДРУГИМИ ПАТТЕРНАМИ

- **Фасад** задаёт новый интерфейс, тогда как **Адаптер** повторно использует старый. *Адаптер* оборачивает только один класс, а *Фасад* оборачивает целую подсистему. #Кроме того, *Адаптер* позволяет двум существующим интерфейсам работать сообща, вместо того, чтобы задать полностью новый.
- **Абстрактная фабрика** может быть использована вместо **Фасада** для того, чтобы скрыть платформо-зависимые классы.
- **Посредник** и **Фасад** похожи тем, что пытаются организовать работу множества существующих классов.
- **Фасад** можно сделать **Одиночкой**, так как обычно нужен только один объект-фасад.
- **Фасад** похож на **Заместитель** тем, что замещает сложную подсистему и может сам её инициализировать. Но в отличие от *Фасада*, *Заместитель* имеет тот же интерфейс, что его служебный объект, благодаря чему их можно взаимозаменять.

СПАСИБО ЗА ВНИМАНИЕ!