

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет
по производственной технологической практике

Место прохождения практики
«Софтверке»

Авторы: Герасимов Михаил

Группа: М3335

Факультет: ФИТиП



УНИВЕРСИТЕТ ИТМО

Санкт-Петербург 2022

Ссылка на github: <https://github.com/Mizzza54/Internship>

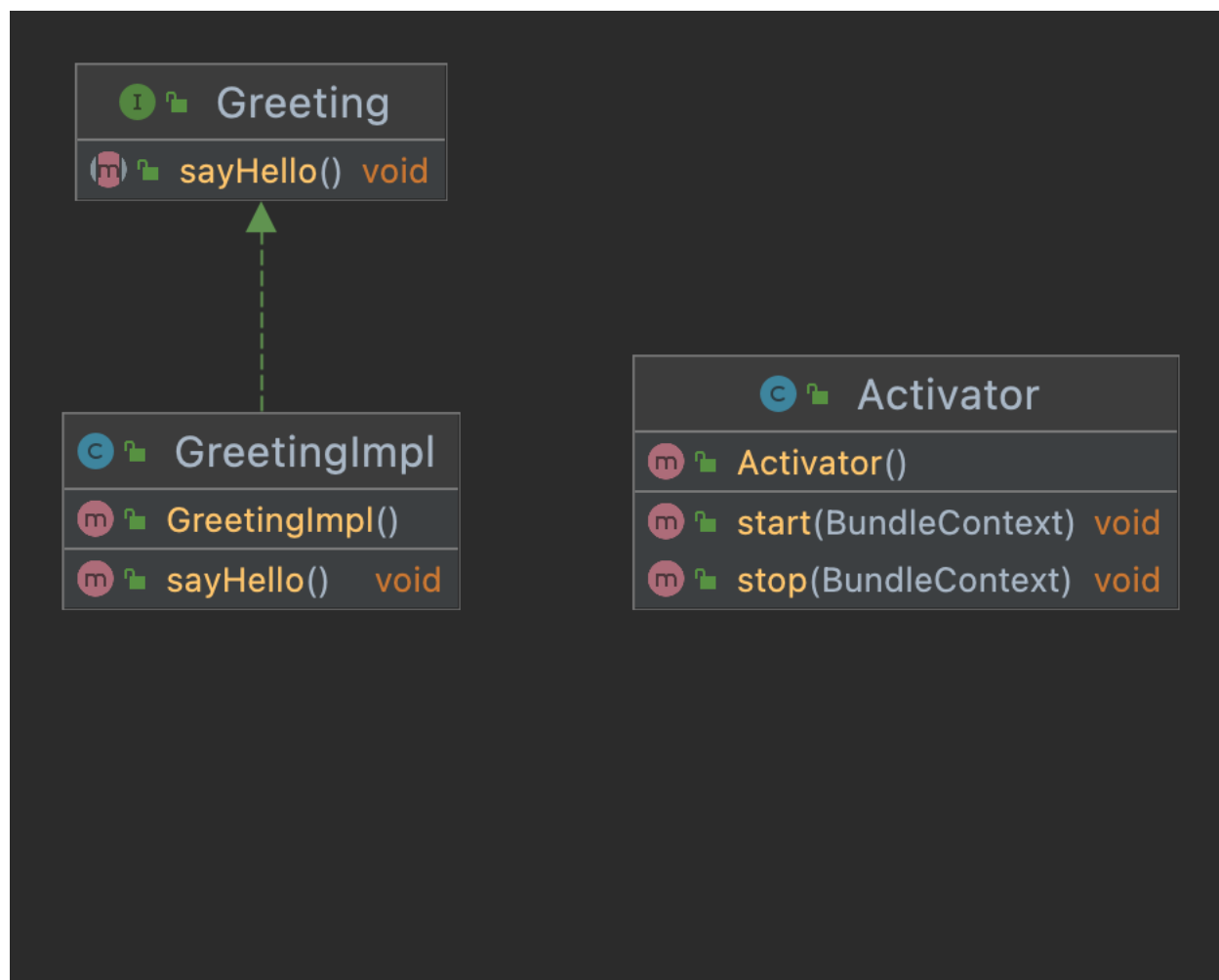
Описание результатов по каждому этапу

- **Этап 1: Подготовительный**

Были прочитаны первые 4 главы книги «OSGi in Action». В ходе прочтения были усвоены основные концепции спецификации OSGi. Так же был установлен Apache Felix - свободный фреймворк, являющийся реализацией спецификации OSGi Release 4. После этого ознакомился с основными командами Apache Felix и установил Felix Web Console.

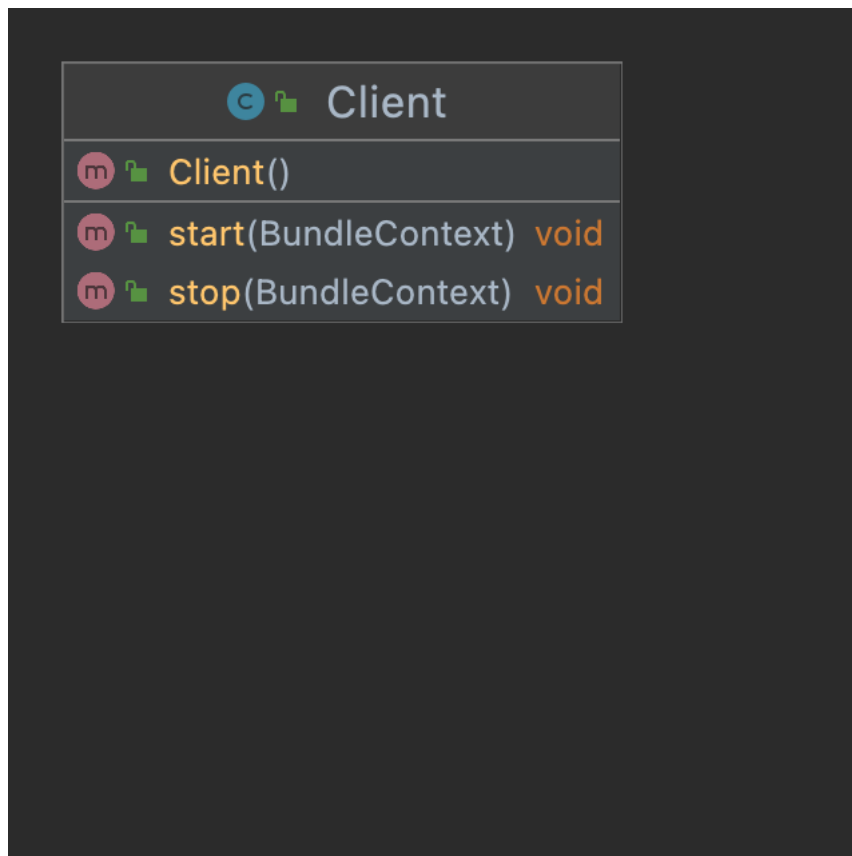
- **Этап 2. Реализация OSGi-сервиса**

Был создан бандл сервиса содержащий интерфейс, активатор бандла и реализацию. Созданный сервис выполняет только одну функцию – вывод на консоль «Hello OSGi World!». Был также создан бандл-клиент, содержащий только активатор. Бандл-клиент потреблял бандл-сервис. Все созданные бандлы были установлены в Apache Felix.



модуль ru.ifmo.gerasimov

Модуль *ru.ifmo.gerasimov* содержит реализацию сервиса. Бандл-сервис зависит только от *org.osgi.framework*.



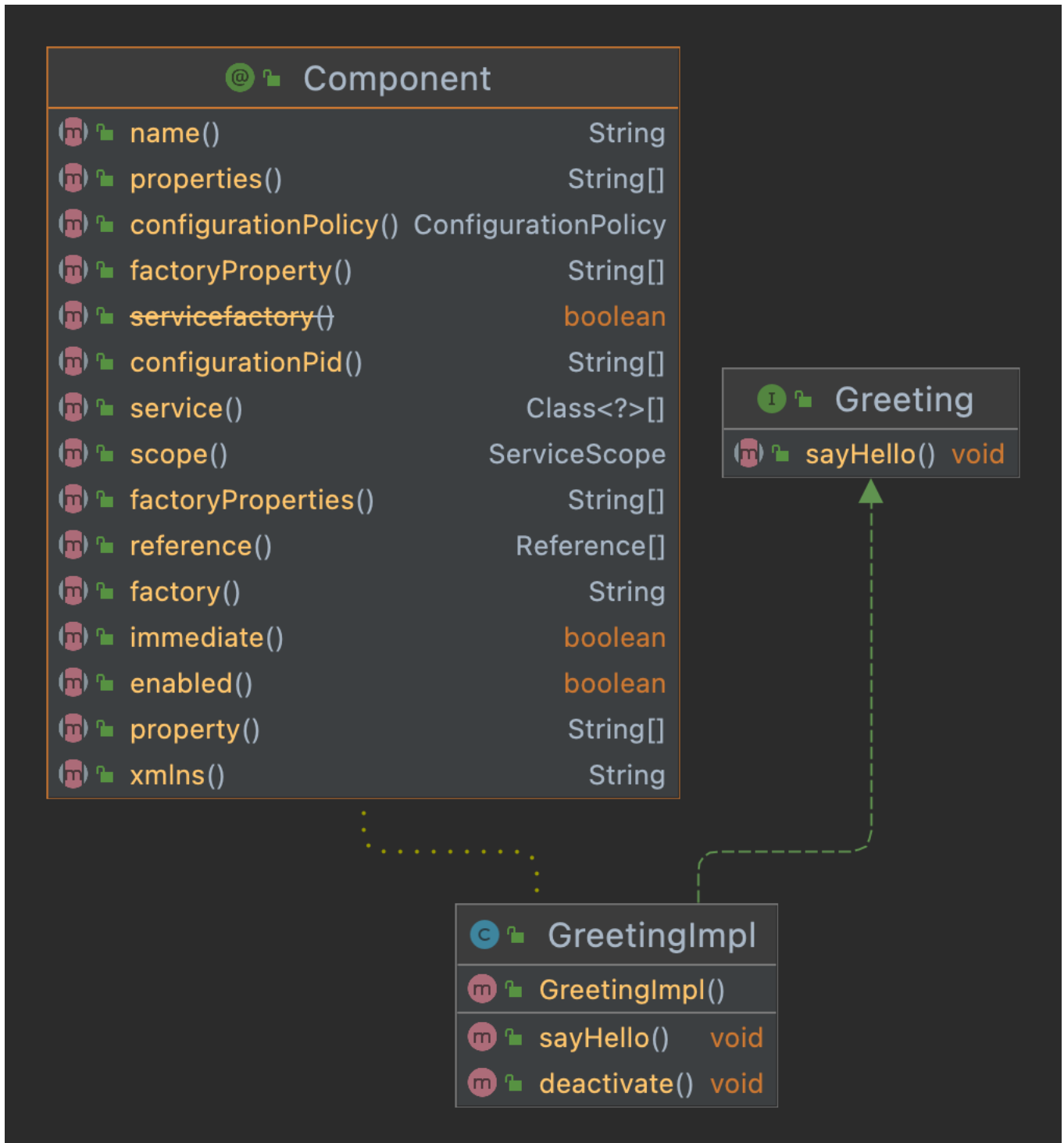
модуль `ru.ifmo.gerasimov.client`

Модуль `ru.ifmo.gerasimov.client` содержит реализацию клиента. Бандл-сервис зависит от `org.osgi.framework` и `ru.ifmo.gerasimov`.

Github: <https://github.com/Mizzza54/Internship/tree/main/Stage2>

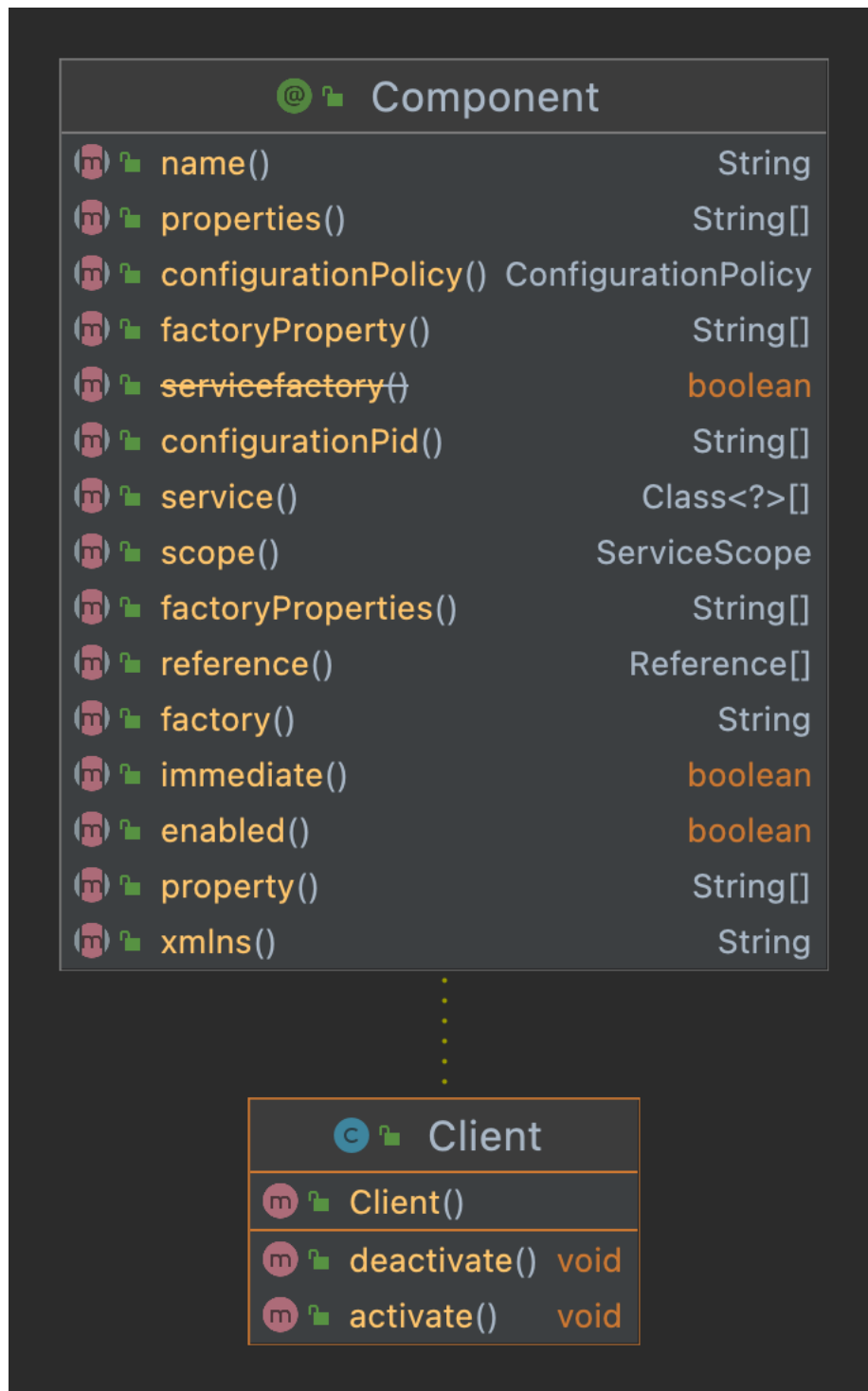
- **Этап 3. Apache Felix Service Component Runtime**

Был создан бандл сервиса содержащий интерфейс и реализацию с использованием аннотаций (*org.osgi.service.component.annotations*). Созданный сервис при вызове метода выводит в консоль «Hello OSGi World!», а при деактивации «Goodbye OSGi world =((by Service)». Был также создан бандл-клиент, содержащий класс, использующий созданный сервис и аннотации. Аннотации используются для активации бандла и для создания ссылки на сервис. Бандл-клиент потребляет бандл-сервис. Все созданные бандлы были установлены в Apache Felix.



модуль *ru.ifmo.gerasimov*

Модуль *ru.ifmo.gerasimov* содержит реализацию сервиса.



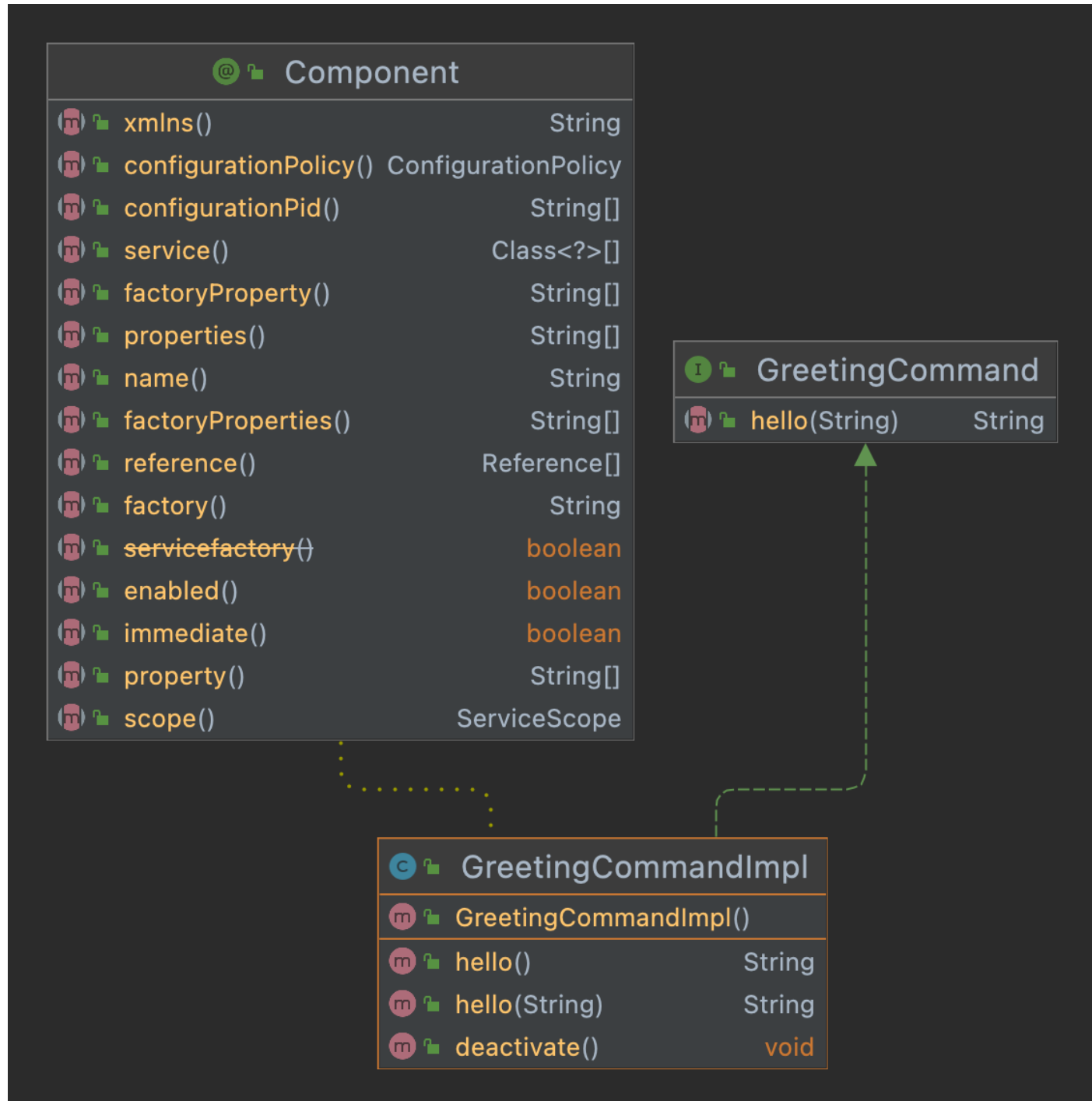
модуль *ru.ifmo.gerasimov.client*

Модуль *ru.ifmo.gerasimov.client* содержит реализацию клиента. Бандл-сервис зависит от *ru.ifmo.gerasimov*.

Github: <https://github.com/Mizzza54/Internship/tree/main/Stage3>

- Этап 4. Создание собственной команды для Apache Felix Gogo

Была создана собственную команда «practice:hello» с одним параметром, которая при вызове печатает на консоль «Hello, <param>», где <param> - введенный пользователем параметр. Был создан один бандл, включающий в себя один интерфейс (API команды) и реализация команды с использованием аннотаций.



модуль ru.ifmo.gerasimov

Модуль ru.ifmo.gerasimov содержит реализацию команды. При активации бандла в консоль выводиться «Hello, Gogo Shell», при деактивации бандла в консоль выводиться «Goodbye, Gogo Shell». Если команде не был передан параметр, то на консоль выводится «Hello,».

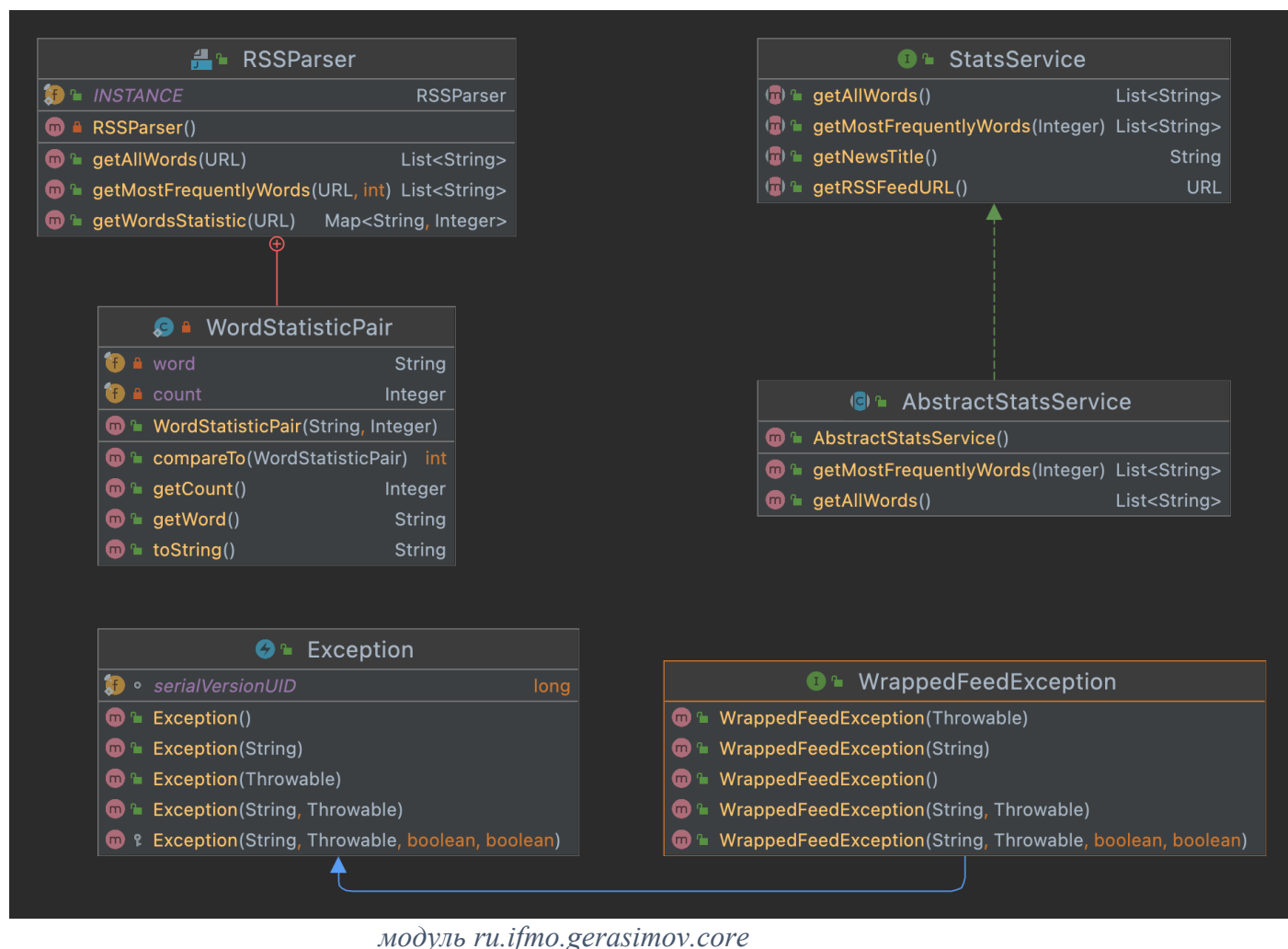
Github: <https://github.com/Mizzza54/Internship/tree/main/Stage4>

- **Этап 5. Создание приложения**

Было создано приложение, которое получает через API новостных порталов список актуальных новостей и выводит на консоль 10 самых часто встречающихся слов из заголовков новостей.

Процесс подсчета инициируется пользователем приложения с помощью консольной команды «*news:stats*». Пользователь может передать источник в качестве параметра. Если команда вводится без параметров, пользователю предлагается выбрать источник данных (один из доступных в системе, или все сразу). Список источников выводится на консоль.

Если пользователь выбирает незарегистрированный (недоступный) источник данных, в системе не зарегистрировано ни одного источника или сеть недоступна, пользователю выводится соответствующее сообщение.



Модуль *ru.ifmo.gerasimov.core* содержит в себе класс *RSSParser*, который отвечает за парсинг RSS формата. Данный класс реализует паттерн проектирования синглтон. Для парсинга RSS используется библиотека *com.rometools*. Класс *WrappedFeedException* является классом-оберткой. Был создан чтобы оборачивать *com.rometools.rome.io.FeedException*. *StatsService* – это интерфейс (API) получения данных из источника. *AbstractStatsService* – абстрактный класс, созданный для избавления от копипасты, так как получения данных происходит везде одинаково и меняется только URL.

Бандл, созданный из этого модуля, зависит от *com.rometools*, *org.slf4j* и *jaxen*.
Так же данный бандл экспортирует все эти зависимости для других бандлов.

Github: <https://github.com/Mizzza54/Internship/tree/main/Stage5>

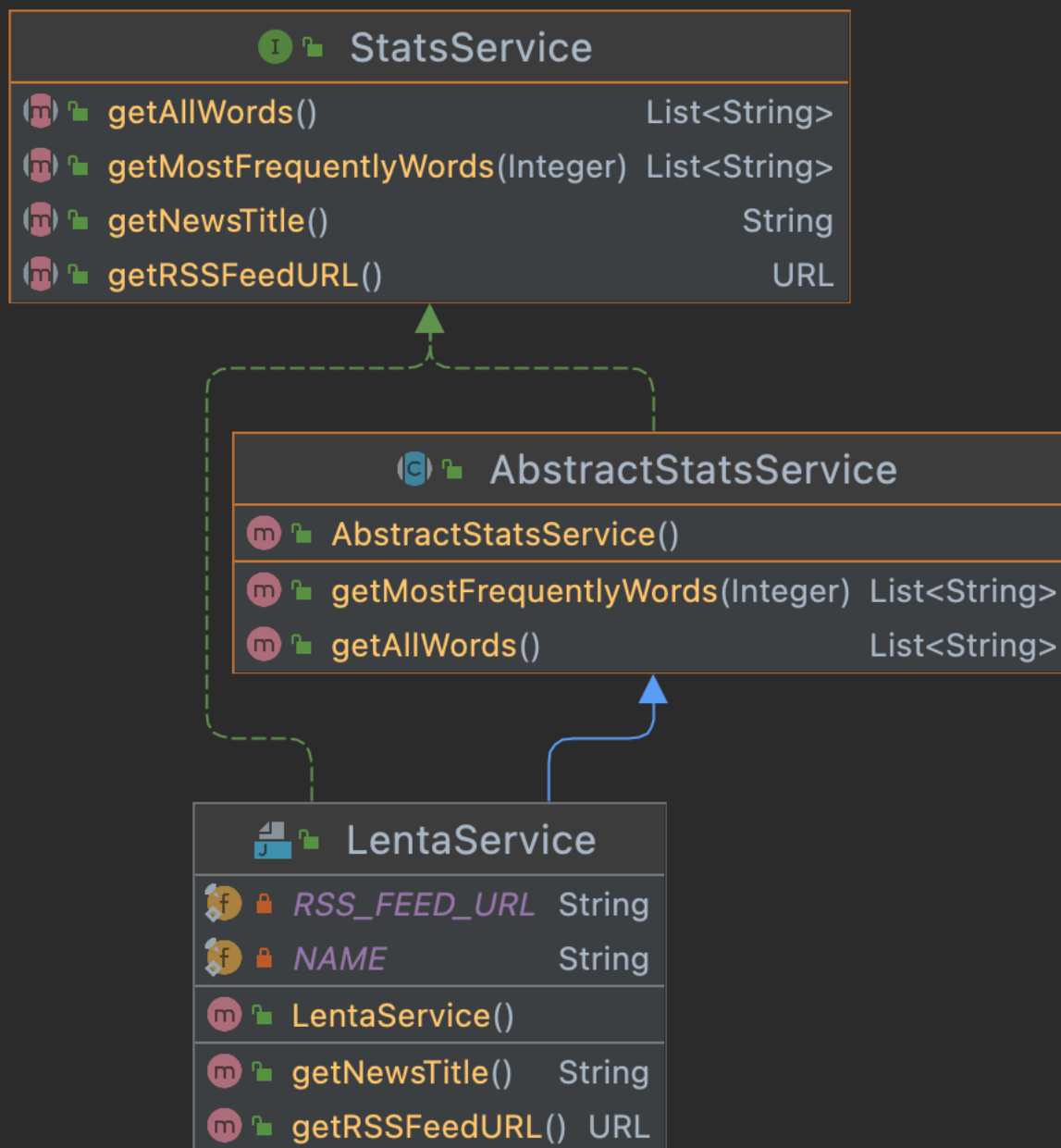
@ Component	
NAME	String
configurationPolicy()	ConfigurationPolicy
reference()	Reference[]
factoryProperties()	String[]
name()	String
factory()	String
configurationPid()	String[]
properties()	String[]
service()	Class<?>[]
property()	String[]
factoryProperty()	String[]
immediate()	boolean
enabled()	boolean
xmlns()	String
servicefactory()	boolean
scope()	ServiceScope

StatsCommand	
stats(String[])	String

StatsCommandImpl	
USAGE	String
statsServices	Map<String, StatsService>
StatsCommandImpl()	
stats(String[])	String
bindService(StatsService)	void
unbindService(StatsService)	void

модуль *ru.ifmo.gerasimov.command*

Модуль *ru.ifmo.gerasimov.command* содержит интерфейс команды и реализует её.



модуль *ru.ifmo.gerasimov.lenta*

Модуль *ru.ifmo.gerasimov.lenta* реализует интерфейс *StatsService*, а именно предоставляет URL на RSS и названия новостного сайта, чтобы в дальнейшем получить заголовки с этого сайта. Бандл зависит только от *ru.ifmo.gerasimov.core*.

Все оставшиеся модули аналогичны модулю *ru.ifmo.gerasimov.lenta*.

Github: <https://github.com/Mizzza54/Internship/tree/main/Stage5>

Выводы

Для меня самым большим преимуществом использования OSGi является необходимость думать об архитектуре приложения. Мы должны более детально понимать о модулях и взаимодействии между ними. Это помогает создать более совершенную архитектуру, в которой каждый модуль отвечает за четко определенные задачи, а модули можно использовать повторно. Так же стоит упомянуть, что OSGi это не только стандарт построения модульных приложений. Он также определяет среду, в которой существуют и выполняются пакеты. Данная среда предоставляет полезные возможности, такие как динамическая загрузка новых бандлов или обновления уже существующих.

Из недостатков можно выделить необходимость создания большого количества модулей. Во-первых не всегда требуется делить приложения на маленькие модули, во-вторых со временем становится трудно поддерживать зависимости между модулями. Так же стоит упомянуть, что не все библиотеки с открытым исходным кодом совместимы с инфраструктурой OSGi. Их бывает трудно внедрить в окружение, завязанное на OSGi.

Я думаю, что OSGi полезен при создании модульных приложений. Это довольно жесткий стандарт, который не будет поощрять плохие решения, например нам желательно с самого начала разработки придерживаться модульной структуры приложения, мы должны постоянно поддерживать паттерн SOA, мы обязаны каждый раз прописывать *MANIFEST*, что иногда помогает лучше разобраться в зависимостях и при необходимости подгрузить только те, которые используются. Так же стоит упомянуть достаточно большое количество готовых сервисов для конфигурации приложения.

Мне кажется, не стоит реализовывать приложения, не адаптированные под архитектуру микросервисов. Например, тип приложений, которые не стоит писать, используя OSGi, это *серверные* приложения. Серверное приложение можно делать распределенным, состоящим из набора мелких приложений, бегающих под разными JVM, на разных компьютерах и, возможно, разнесенных географически, а следовательно, запустить это все в одной среде выполнения не самая тривиальная задача. Так же нет смысла писать не масштабные приложения, например этапы 2, 3, 4, 5. В данных приложениях не нужна сложная модульность, а иногда она только мешает.