Видишь меня?
Границы
зависимостей в Gradle.

КТО МЫ

PVS-Studio

Статический анализатор кода. Находит ошибки и потенциальные уязвимости до того, как они попадут в прод.

- Анализатор для C, C++, C# и Java
- Разрабатываем новые анализаторы для JavaScript/TypeScript и Go

pvs-studio.ru/training_gradle

СЕГОДНЯ

Что *разберём?*

СЕГОДНЯ

Что *разберём?*

01 java vs java-library

Какой плагин выбрать для библиотечного модуля

СЕГОДНЯ

Что *разберём?*

01 **java vs java-library**

Какой плагин выбрать для библиотечного модуля

02 **implementation vs api**

Где зависимость утекает наружу, а где - нет

СЕГОДНЯ

Что *разберём?*

01 **java vs java-library**

Какой плагин выбрать для библиотечного модуля

02 **implementation vs api**

Где зависимость утекает наружу, а где - нет

03 **Граф зависимостей**

`gradle :dependencies` - читаем и понимаем

Два плагина, *три конфигурации.*

ЧТО ДАЁТ	● **JAVA** · БАЗОВЫЙ	● **JAVA-LIBRARY** · РАСШИРЕННЫЙ

Два плагина, *три конфигурации.*

ЧТО ДАЁТ	● JAVA · БАЗОВЫЙ	● JAVA-LIBRARY · РАСШИРЕННЫЙ
implementation	✓ есть - внутренняя зависимость	✓ есть

build.gradle.kts

```
dependencies {
  implementation("com.google.guava:guava:32.0.0-jre")
}
```

Два плагина, *три конфигурации.*

ЧТО ДАЁТ	● JAVA · БАЗОВЫЙ	● JAVA-LIBRARY · РАСШИРЕННЫЙ
implementation	✓ есть - внутренняя зависимость	✓ есть
compileOnly / runtimeOnly	✓ есть	✓ есть

build.gradle.kts

```
dependencies {
  compileOnly("org.jetbrains:annotations:24.0.0")
  runtimeOnly("org.postgresql:postgresql:42.7.4")
}
```

Два плагина, *три конфигурации.*

ЧТО ДАЁТ	● JAVA · БАЗОВЫЙ	● JAVA-LIBRARY · РАСШИРЕННЫЙ
implementation	✓ есть - внутренняя зависимость	✓ есть
compileOnly / runtimeOnly	✓ есть	✓ есть
api	– нет - нечем экспонировать тип наружу	✓ есть - публичная зависимость

core/build.gradle.kts

```
plugins { `java-library` }

dependencies {
  api(project(":api"))  // утекает наружу
}
```

ЧАСТЬ 2 - ВИДИМОСТЬ

implementation vs *api.*

АСПЕКТ	● **IMPLEMENTATION** · СКРЫТО	● **API** · ПУБЛИЧНО

implementation vs *api*.

АСПЕКТ	● **IMPLEMENTATION** · СКРЫТО	● **API** · ПУБЛИЧНО
Видна потребителю?	– не видна на этапе компиляции	✓ транзитивно попадает в classpath

```
./gradlew :app:compileJava - implementation

> Task :app:compileJava FAILED
Main.java:3: error:
package shop.api does not exist
import shop.api.OrderService;
```

implementation vs *api*.

АСПЕКТ	● **IMPLEMENTATION** · СКРЫТО	● **API** · ПУБЛИЧНО
Видна потребителю?	– не видна на этапе компиляции	✓ транзитивно попадает в classpath
Утечка в публичный API	✓ нет - если типа нет в сигнатуре	– да - становится частью контракта

core/src/main/java/shop/core/OrderServiceImpl.java

```
public class OrderServiceImpl
  implements OrderService {
  // тип :api в публичной сигнатуре
}
```

implementation vs *api*.

АСПЕКТ	● IMPLEMENTATION · СКРЫТО	● API · ПУБЛИЧНО
Видна потребителю?	– не видна на этапе компиляции	✓ транзитивно попадает в classpath
Утечка в публичный API	✓ нет - если типа нет в сигнатуре	– да - становится частью контракта
Когда выбирать	· тип не появляется в сигнатурах	· тип в публичном API: return, параметры, супертипы

core/build.gradle.kts

```
plugins { `java-library` }

dependencies {
  api(project(":api"))
}

// ./gradlew :app:compileJava → BUILD SUCCESSFUL
```

Что *видит* :app?

```
./gradlew :app:dependencies --configuration compileClasspath

\--- project :core
    \--- project :api

// :api пришёл транзитивно - именно это нам и нужно
```

ПРАКТИКА

Идём на *лайвкодинг*

КОНЕЦ ВЕБИНАРА 02

Вопросы *и ответы*

Спрашиваете — отвечаем

конец