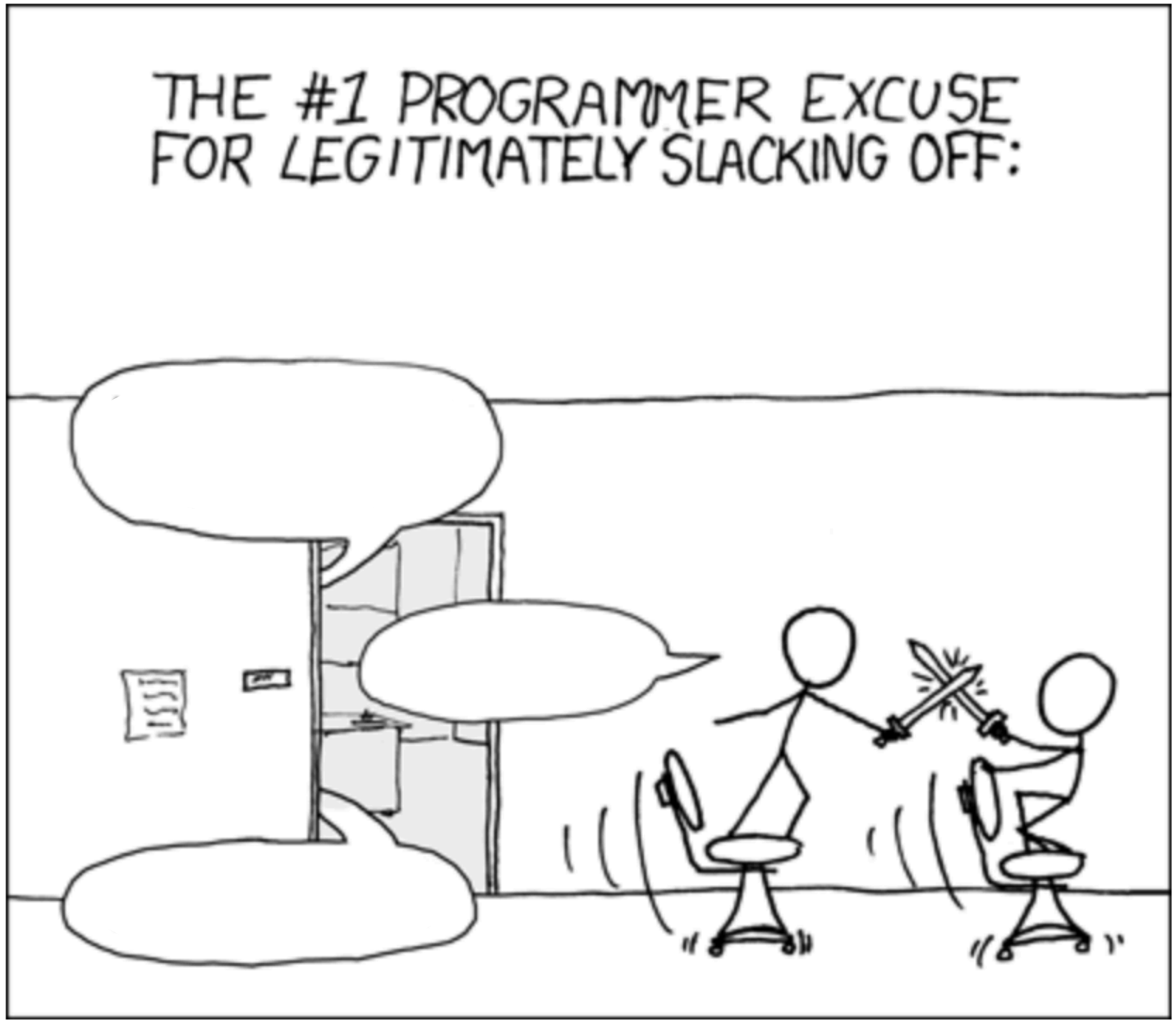
Gradle: Incremental compilation 101

Сергей Опивалов

Gradle Inc.

mobius


THE #1 PROGRAMMER EXCUSE
FOR LEGITIMATELY SLACKING OFF:

План

1. **Application Binary Interface обзор**
2. **Инкрементальная компиляция**
3. **Compilation avoidance**
4. **Как развивать свою кодовую базу?**

Погружение

Incremental compilation & compile avoidance in Gradle 3.4

Gradle Java-plugin compile avoidance doc

Gradle Java-plugin incremental compilation doc

Dark secrets of Kotlin compilation

A New Approach to Incremental Compilation in Kotlin

Погружение

extracting the ABI	how to compare ABIs.	Multi project ABI-compatible change
ABI comparison	detect a change in the ABI	Library ABI-breaking change
identical ABIs	ABI-compatible way	track ABI changes

ABI

ABI

- **Application Binary Interface**
- **Интерфейс между двумя программными модулями**
- **Соглашение о вызове функций (Calling convention/Procedure call standard)**

ABI в мире JVM

Два класса имеют одинаковый ABI если они взаимозаменяемы в compile classpath

https://blog.jetbrains.com/kotlin/2022/07/a-new-approach-to-incremental-compilation-in-kotlin/

ABI в мире JVM

```java
public class Foo {

    public void publicApi() {
        System.out.println("Foo");
    }

    private void privateMethod() {}
}
```

==

```java
public class Foo {

    public void publicApi() {
       //method body changed
    }

    private void privateMethodNew() {}
}
```

https://blog.jetbrains.com/kotlin/2022/07/a-new-approach-to-incremental-compilation-in-kotlin/

Kotlin

Source

```
package com.ic101

class MyClass {

    fun foo(x : Int, y : Int) : Int =
        x + y

    private fun bar() {
        println("Bar")
    }
}
```

ABI

```
public final class com/ic101/MyClass {
    public fun <init> ()V
    public final fun foo (II)I
}
```

Kotlin

Source

```
package com.ic101

class MyClass {

    fun foo(x : Int, y : Int) : Int =
        x + y

    private fun bar() {
        println("Bar")
    }
}
```

ABI

```
public final class com/ic101/MyClass {
    public fun <init> ()V
    public final fun foo (II)I
}
```

Kotlin

Source

```
package com.ic101

class MyClass {

    fun foo(x : Int, y : Int) : Int =
        x + y

    private fun bar() {
        println("Bar")
    }
}
```

ABI

```
public final class com/ic101/MyClass {
    public fun <init> ()V
    public final fun foo (II)I
}
```

Kotlin

Source

```
package com.ic101

class MyClass(){

    fun foo(x : Int, y : Int) : Int =
        x + y

    private fun bar() {
        println("Bar")
    }
}
```

ABI

```
public final class com/ic101/MyClass {
    public fun <init> ()V
    public final fun foo (II)I
}
```

Kotlin binary compatibility validator

https://github.com/Kotlin/binary-compatibility-validator

Incremental compilation

Incremental compilation(IC)

Java IC: Initial

```
public class CoffeeMachine {

    public void makeCoffee() {
        System.out.println("Making coffee");
    }
}


public class CoffeeShop {

    private CoffeeMachine coffeeMachine =
         new CoffeeMachine();

    public void orderCoffee() {
        System.out.println("Take your coffee");
        coffeeMachine.makeCoffee();
    }
}
```

Java IC: Initial

```
public class CoffeeMachine {

    public void makeCoffee() {
        System.out.println("Making coffee");
    }
}
```

```
./gradlew compileJava -d
```

```
public class CoffeeShop {

    private CoffeeMachine coffeeMachine =
         new CoffeeMachine();

    public void orderCoffee() {
        System.out.println("Take your coffee");
        coffeeMachine.makeCoffee();
    }
}
```

Java IC: Round 1

```
public class CoffeeMachine {

    public void makeCoffee() {
        System.out.println("Making coffee");
    }
}
```

```
Recompiled classes [com.ic101.CoffeeShop]
```

```
public class CoffeeShop {

    private CoffeeMachine coffeeMachine =
         new CoffeeMachine();

    public void orderCoffee() {
        System.out.println("Take your coffee sir");
        coffeeMachine.makeCoffee();
    }
}
```

Java IC: Round 2

```
public class CoffeeMachine {

    public String makeCoffee() {
        System.out.println("Making coffee");
        return "Cappuccino";
    }
}
```

```
Recompiled classes
[com.ic101.CoffeeMachine,com.ic101.CoffeeShop]
```

```
public class CoffeeShop {

    private CoffeeMachine coffeeMachine =
         new CoffeeMachine();

    public void orderCoffee() {
        System.out.println("Take your coffee sir");
        coffeeMachine.makeCoffee();
    }
}
```

Java IC: Round 3

```
public class CoffeeMachine {

    public void makeCoffee() {
        System.out.println("Making cappuccino");
    }
}
```

```
Recompiled classes
[com.ic101.CoffeeMachine,com.ic101.CoffeeShop]
```

```
public class CoffeeShop {

    private CoffeeMachine coffeeMachine =
         new CoffeeMachine();

    public void orderCoffee() {
        System.out.println("Take your coffee sir");
        coffeeMachine.makeCoffee();
    }
}
```

Java plugin doc

- Gradle will recompile all classes affected by a change
- A class is affected if it has been changed or if it depends on another affected class…
- A class's dependencies are determined from type references in its bytecode …

https://docs.gradle.org/current/userguide/java_plugin.html#sec:incremental_compile

Java IC

```
class CoffeeShop {
  private CoffeeMachine coffeeMachine
}

class CoffeeMachine {
  private WaterPump waterPump
}

class WaterPump {
}
```

Java IC

```
class CoffeeShop {
  private CoffeeMachine coffeeMachine;
}

class CoffeeMachine {
  private WaterPump waterPump;
}

class WaterPump {
  private void getWater() {}
}
```

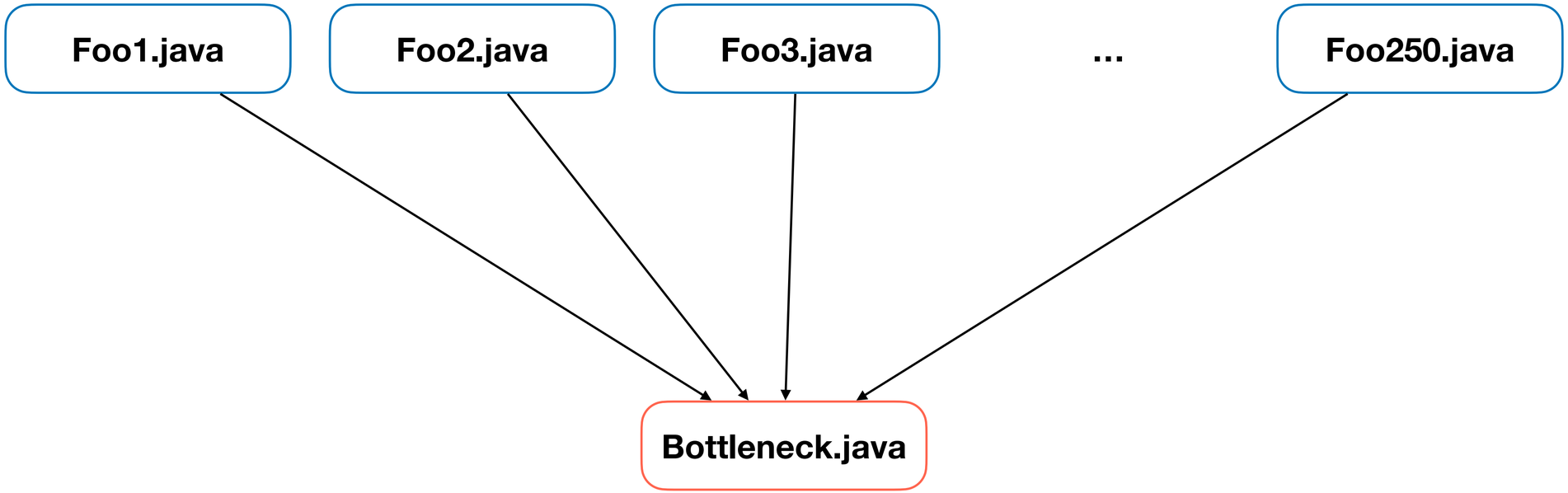
Gradle version < 6.0

```
Recompiled classes
[com.ic101.CoffeeShop,com.ic101.CoffeeMachine,
com.ic101.WaterPump]
```

Gradle version ⩾ 6.0

```
Recompiled classes
[com.ic101.CoffeeMachine,com.ic101.WaterPump]
```

Java IC: Синтетический тест



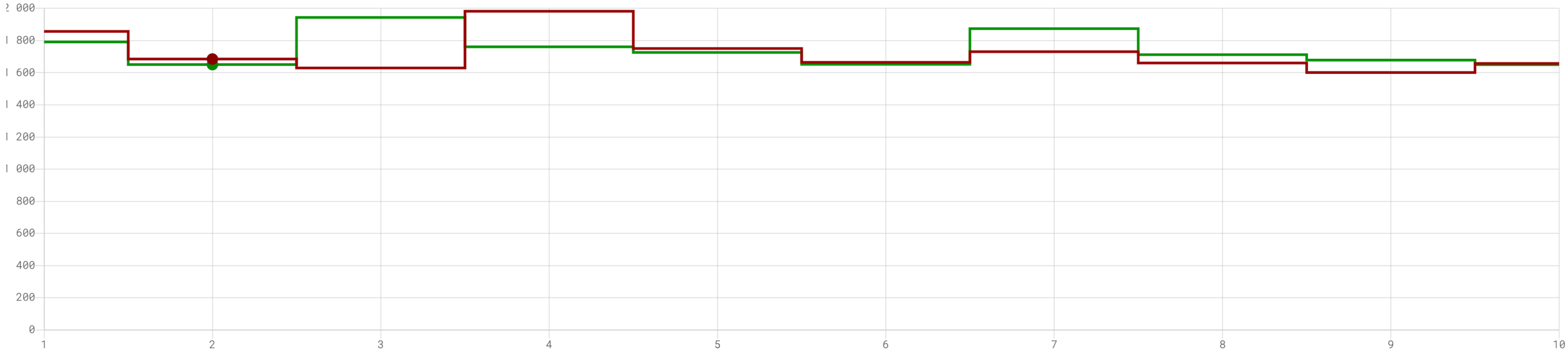
Android Studio poet

https://github.com/android/android-studio-poet

Java IC: Синтетический тест

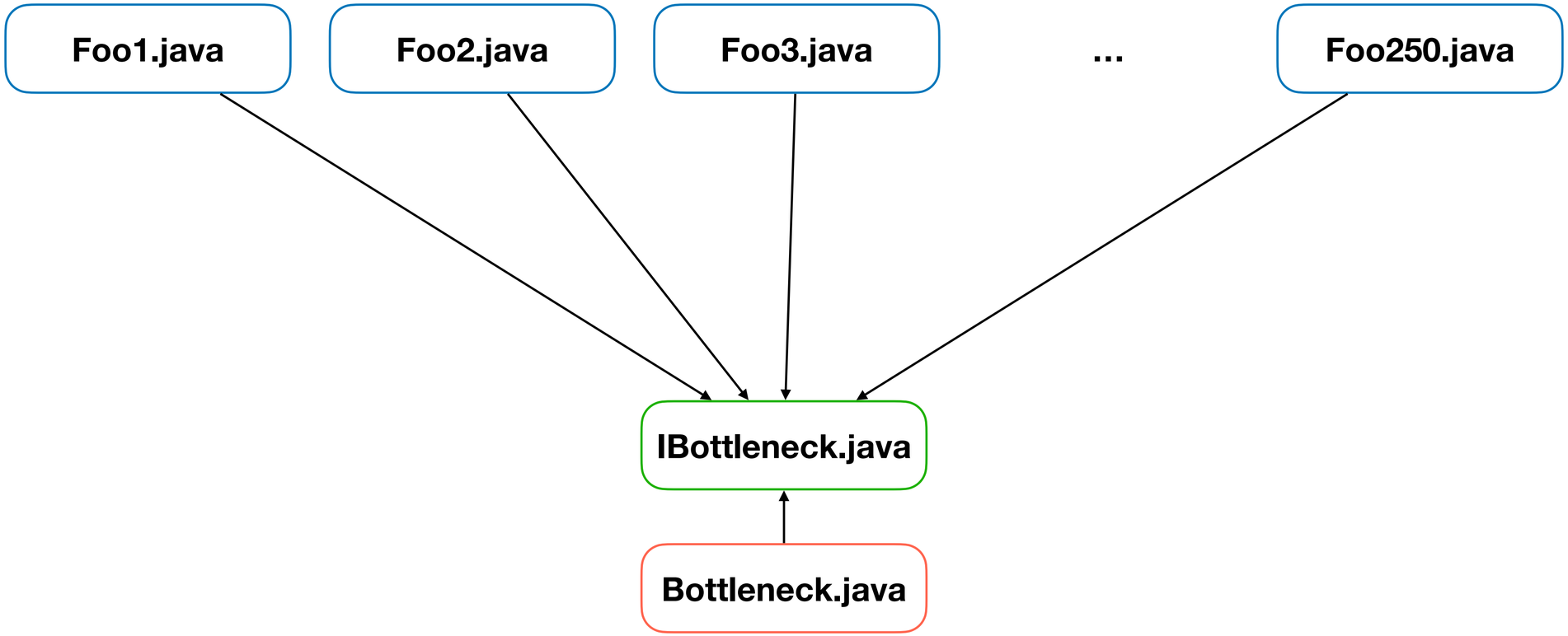
scenario	bottleneck_non_abi_change	bottleneck_abi_change
value	total execution time	
measured build #1	1857	1791
measured build #2	1685	1650
measured build #3	1629	1943
measured build #4	1982	1761
measured build #5	1750	1725
measured build #6	1664	1652
measured build #7	1730	1873
measured build #8	1660	1712
measured build #9	1601	1678
measured build #10	1657	1650

Java IC: Синтетический тест



Сценарий	Медиана
bottleneck_non_abi_change	1 675,08 ms
bottleneck_abi_change	1 718,96 ms

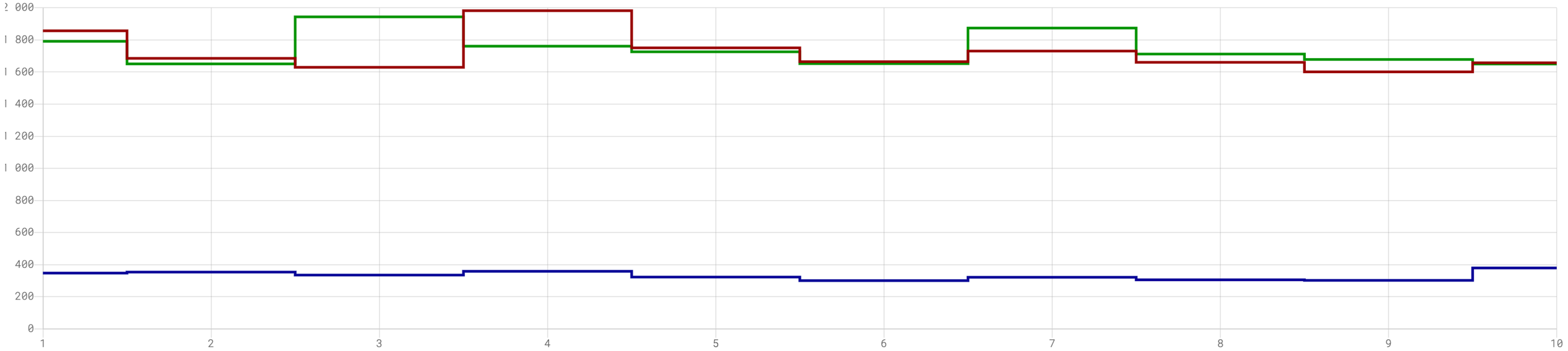
Java IC: Синтетический тест



Java IC: Синтетический тест

scenario	bottleneck_non_abi_change	bottleneck_optimized_change
value	total execution time	
measured build #1	1857	347
measured build #2	1685	353
measured build #3	1629	334
measured build #4	1982	358
measured build #5	1750	322
measured build #6	1664	300
measured build #7	1730	320
measured build #8	1660	305
measured build #9	1601	301
measured build #10	1657	378

Java IC: Синтетический тест



Сценарий	Медиана
bottleneck_non_abi_change	1 675,08 ms
bottleneck_abi_change	1 718,96 ms
bottleneck_optimized_change	328,67 ms

Gradle profiler

https://github.com/gradle/gradle-profiler

Java IC: Константы

```
public class WaterHeater {

    public static final int TEMP = 80;
}
```

```
public class CoffeeMachine {

    void makeCoffee() {
        System.out.println("Water temp is " + WaterHeater.TEMP);
    }
}
```

Java IC: Константы

```
public class WaterHeater {

    public static final int TEMP = 80;
}
```

```
public class CoffeeMachine {

    void makeCoffee() {
        System.out.println("Water temp is " + WaterHeater.TEMP);
    }
}
```

Java IC: Константы

javap -c CoffeeMachine.class

```
void makeCoffee();
   Code:
      0: getstatic     #2  // Field java/lang/System.out:Ljava/io/PrintStream;
      3: ldc           #4  // String Water temp is 80
      5: invokevirtual #5  // Method java/io/PrintStream.println:(Ljava/lang/String;)V
      8: return
```

https://docs.oracle.com/javase/specs/jls/se8/html/jls-13.html

Java IC: Константы

javap -c CoffeeMachine.class

```
void makeCoffee();
   Code:
      0: getstatic     #2  // Field java/lang/System.out:Ljava/io/PrintStream;
      3: ldc           #4  // String Water temp is 80
      5: invokevirtual #5  // Method java/io/PrintStream.println:(Ljava/lang/String;)V
      8: return
```

https://docs.oracle.com/javase/specs/jls/se8/html/jls-13.html

Java IC: Константы

```
public class WaterHeater {

    public static final int TEMP = 80;
}
```

Java IC: Константы

```
public class WaterHeater {

    public static final int TEMP = 100;
}
```

Java IC: Константы

```
public class WaterHeater {

    public static final int TEMP = 80;

    private int duration = 60;
}
```

```
Gradle version < 7.1

Full recompilation is required because
'WaterHeater.java' was changed
```

Java IC: Константы

```
public class WaterHeater {

    public static final int TEMP = 80;
}
```

```
public class WaterHeater {

    public static int getWaterTemp() {
        return 80;
    }
}
```

Java IC: Выводы

- **Class-based эвристика. Компилируются измененные файлы и файлы которые зависят от измененных**

- **Изменение файлов, содержащих публичные константы может привести к полной компиляции проекта**

Kotlin IC: Initial

```
class CoffeeMachine {

    fun makeCoffee() {
        println("Making coffee")
    }
}

class CoffeeShop {

    private val coffeeMachine =
        CoffeeMachine()

    fun orderCoffee() {
        println("Take your coffee")
        coffeeMachine.makeCoffee()
    }
}
```

Kotlin IC: Initial

```
class CoffeeMachine {

    fun makeCoffee() {
        println("Making coffee")
    }
}
```

```
./gradlew compileKotlin -d
```

```
class CoffeeShop {

    private val coffeeMachine =
        CoffeeMachine()

    fun orderCoffee() {
        println("Take your coffee")
        coffeeMachine.makeCoffee()
    }
}
```

Kotlin IC: Round 1

```
class CoffeeMachine {

    fun makeCoffee() {
        println("Making coffee")
    }
}

class CoffeeShop {

    private val coffeeMachine =
        CoffeeMachine()

    fun orderCoffee() {
        println("Take your coffee sir")
        coffeeMachine.makeCoffee()
    }
}
```

```
[KOTLIN] [IC] …/CoffeeShop.kt is marked dirty:
was modified since last time
[KOTLIN] compile iteration: …/CoffeeShop.kt
```

Kotlin IC: Round 2

```
class CoffeeMachine {

    fun makeCoffee() {
        println("Making cappuccino")
    }
}

class CoffeeShop {

    private val coffeeMachine =
        CoffeeMachine()

    fun orderCoffee() {
        println("Take your coffee sir")
        coffeeMachine.makeCoffee()
    }
}
```

```
[KOTLIN] [IC] …/CoffeeMachine.kt is marked dirty:
was modified since last time
[KOTLIN] compile iteration: …/CoffeeMachine.kt
```

Kotlin IC: Round 3

```
class CoffeeMachine {

    fun makeCoffee() : String {
        println("Making cappuccino")
        return "Cappuccino"
    }
}
class CoffeeShop {

    private val coffeeMachine =
        CoffeeMachine()

    fun orderCoffee() {
        println("Take your coffee sir")
        coffeeMachine.makeCoffee()
    }
}
```

```
[KOTLIN] [IC] .../CoffeeMachine.kt is marked dirty:
was modified since last time
[KOTLIN] compile iteration: .../CoffeeMachine.kt

[KOTLIN] [IC] .../CoffeeShop.kt is marked dirty:
dirty member CoffeeMachine#makeCoffee
[KOTLIN] compile iteration: .../CoffeeShop.kt
```

Kotlin IC: Round 3

```
class CoffeeMachine {

    fun makeCoffee() : String {
        println("Making cappuccino")
        return "Cappuccino"
    }
}
class CoffeeShop {

    private val coffeeMachine =
        CoffeeMachine()

    fun orderCoffee() {
        println("Take your coffee sir")
        coffeeMachine.makeCoffee()
    }
}
```

```
[KOTLIN] [IC] .../CoffeeMachine.kt is marked dirty:
was modified since last time
[KOTLIN] compile iteration: .../CoffeeMachine.kt

[KOTLIN] [IC] .../CoffeeShop.kt is marked dirty:
dirty member CoffeeMachine#makeCoffee
[KOTLIN] compile iteration: .../CoffeeShop.kt
```

Kotlin IC: Round 3

```
class CoffeeMachine {

    fun makeCoffee() : String {
        println("Making cappuccino")
        return "Cappuccino"
    }
}
class CoffeeShop {

    private val coffeeMachine =
        CoffeeMachine()

    fun orderCoffee() {
        println("Take your coffee sir")
        coffeeMachine.makeCoffee()
    }
}
```

```
[KOTLIN] [IC] .../CoffeeMachine.kt is marked dirty:
was modified since last time
[KOTLIN] compile iteration: .../CoffeeMachine.kt

[KOTLIN] [IC] .../CoffeeShop.kt is marked dirty:
dirty member CoffeeMachine#makeCoffee
[KOTLIN] compile iteration: .../CoffeeShop.kt
```

Kotlin IC: Round 4

```
class WaterPump {

    fun getHotWater() {
        println("Hot water")
    }

    fun getColdWater() {
        println("Cold water")
    }
}

class CoffeeShop {

    private val waterPump = WaterPump()

    fun getGlassOfWater() {
        waterPump.getColdWater()
    }
}

class CoffeeMachine {

    private val waterPump = WaterPump()

    fun makeCoffee() {
        waterPump.getHotWater()
        println("Making coffee")
    }
}
```

Kotlin IC: Round 4

```
class WaterPump {

    fun getHotWater() {
        println("Hot water")
    }

    fun getColdWater() {
        println("Cold water")
    }
}

class CoffeeShop {

    private val waterPump = WaterPump()

    fun getGlassOfWater() {
        waterPump.getColdWater()
    }
}

class CoffeeMachine {

    private val waterPump = WaterPump()

    fun makeCoffee() {
        waterPump.getHotWater()
        println("Making coffee")
    }
}
```

Kotlin IC: Round 4

```
class WaterPump {

    fun getHotWater() {
        println("Hot water")
    }

    fun getColdWater() {
        println("Cold water")
    }
}

class CoffeeShop {

    private val waterPump = WaterPump()

    fun getGlassOfWater() {
        waterPump.getColdWater()
    }
}

class CoffeeMachine {

    private val waterPump = WaterPump()

    fun makeCoffee() {
        waterPump.getHotWater()
        println("Making coffee")
    }
}
```

Kotlin IC: Round 4

```
class WaterPump {

    fun getHotWater() {
        println("Hot water")
    }

    fun getColdWater() {
        println("Cold water")
    }
}

class CoffeeShop {

    private val waterPump = WaterPump()

    fun getGlassOfWater() {
        waterPump.getColdWater()
    }
}

class CoffeeMachine {

    private val waterPump = WaterPump()

    fun makeCoffee() {
        waterPump.getHotWater()
        println("Making coffee")
    }
}
```

Kotlin IC: Round 4

```
class WaterPump {

    fun getHotWater() : String {
        return "Hot water"
    }


    fun getColdWater() {
        println("Cold water")
    }
}

class CoffeeShop {

    private val waterPump = WaterPump()

    fun getGlassOfWater() {
        waterPump.getColdWater()
    }
}

class CoffeeMachine {

    private val waterPump = WaterPump()

    fun makeCoffee() {
        waterPump.getHotWater()
        println("Making coffee")
    }
}
```

```
[KOTLIN] [IC] .../WaterPump.kt is marked dirty:
was modified since last time
[KOTLIN] compile iteration: .../WaterPump.kt

[KOTLIN] [IC] .../CoffeeMachine.kt is marked dirty:
dirty member WaterPump#getHotWater
[KOTLIN] compile iteration: .../CoffeeMachine.kt
```

Kotlin IC: Round 4

```
class WaterPump {

    fun getHotWater() : String {
        return "Hot water"
    }


    fun getColdWater() {
        println("Cold water")
    }
}

class CoffeeShop {

    private val waterPump = WaterPump()

    fun getGlassOfWater() {
        waterPump.getColdWater()
    }
}

class CoffeeMachine {

    private val waterPump = WaterPump()

    fun makeCoffee() {
        waterPump.getHotWater()
        println("Making coffee")
    }
}
```

```
[KOTLIN] [IC] .../WaterPump.kt is marked dirty:
was modified since last time
[KOTLIN] compile iteration: .../WaterPump.kt

[KOTLIN] [IC] .../CoffeeMachine.kt is marked dirty:
dirty member WaterPump#getHotWater
[KOTLIN] compile iteration: .../CoffeeMachine.kt
```

Kotlin IC: Round 4

```
class WaterPump {

    fun getHotWater() : String {
        return "Hot water"
    }


    fun getColdWater() {
        println("Cold water")
    }
}

class CoffeeShop {

    private val waterPump = WaterPump()

    fun getGlassOfWater() {
        waterPump.getColdWater()
    }
}

class CoffeeMachine {

    private val waterPump = WaterPump()

    fun makeCoffee() {
        waterPump.getHotWater()
        println("Making coffee")
    }
}
```

```
[KOTLIN] [IC] .../WaterPump.kt is marked dirty:
was modified since last time
[KOTLIN] compile iteration: .../WaterPump.kt

[KOTLIN] [IC] .../CoffeeMachine.kt is marked dirty:
dirty member WaterPump#getHotWater
[KOTLIN] compile iteration: .../CoffeeMachine.kt
```

CoffeeShop.kt не был скомпилирован!

Kotlin IC: Inline функции

```
class WaterPump {

    inline fun getColdWater() {
        println("Cold water")
    }
}

class CoffeeShop {

    private val waterPump = WaterPump()

    fun getGlassOfWater() {
        waterPump.getColdWater()
    }
}
```

Kotlin IC: Inline функции

```
class WaterPump {

    inline fun getColdWater() {
        println("Cold water")
        println("Done")
    }
}
class CoffeeShop {

    private val waterPump = WaterPump()

    fun getGlassOfWater() {
        waterPump.getColdWater()
    }
}
```

```
[KOTLIN] [IC] .../WaterPump.kt is marked dirty:
was modified since last time
[KOTLIN] compile iteration: .../WaterPump.kt

[KOTLIN] [IC] .../CoffeeShop.kt is marked dirty:
dirty member WaterPump#getColdWater()
[KOTLIN] compile iteration: .../CoffeeShop.kt
```

Kotlin IC: Inline функции

```
class WaterPump {

    inline fun getColdWater() {
        println("Cold water")
        println("Done")
    }
}
class CoffeeShop {

    private val waterPump = WaterPump()

    fun getGlassOfWater() {
        waterPump.getColdWater()
    }
}
```

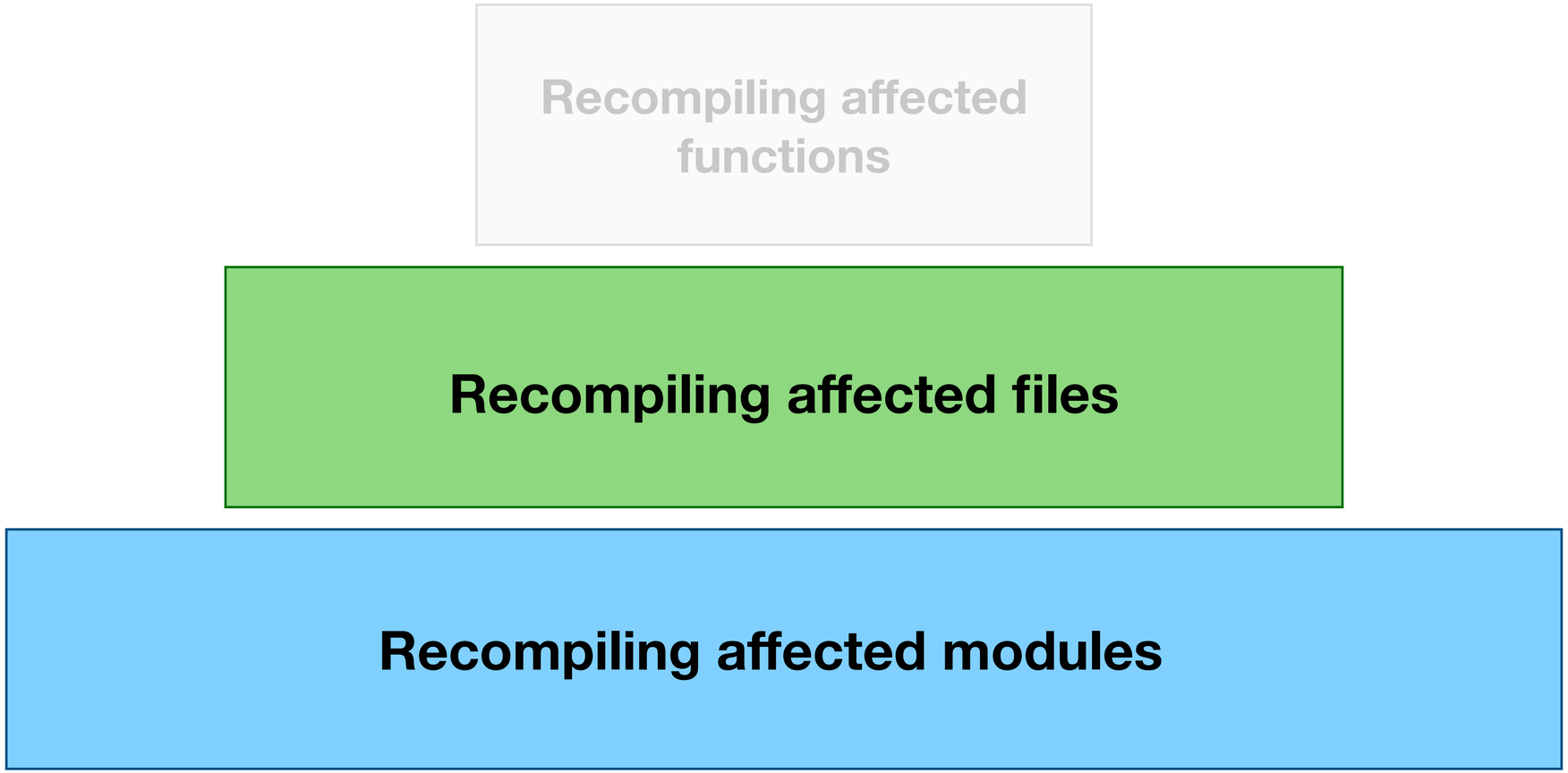
```
[KOTLIN] [IC] .../WaterPump.kt is marked dirty:
was modified since last time
[KOTLIN] compile iteration: .../WaterPump.kt

[KOTLIN] [IC] .../CoffeeShop.kt is marked dirty:
dirty member WaterPump#getColdWater
[KOTLIN] compile iteration: .../CoffeeShop.kt
```

https://kotlinlang.org/docs/inline-functions.html#restrictions-for-public-api-inline-functions

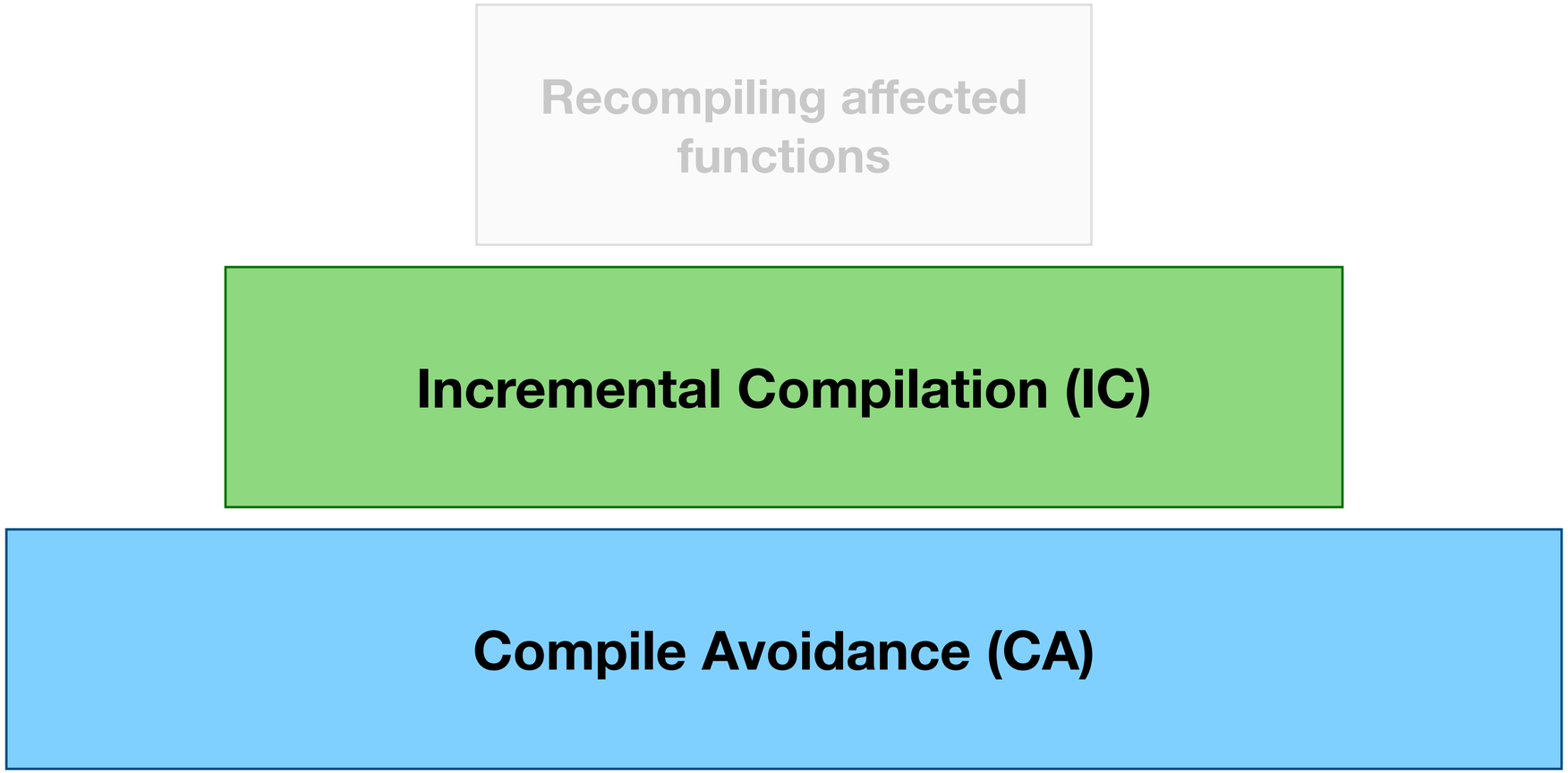
Kotlin IC: Выводы

- Symbol-based эвристика для определения набора файлов для компиляции
- Изменения исходного кода могут быть ABI-совместимые и ABI-несовместимые
- Изменения публичных сигнатур класса не всегда приводят к компиляции всех зависимых классов
- Изменение тела inline функции приводит к компиляции зависимых от функции классов

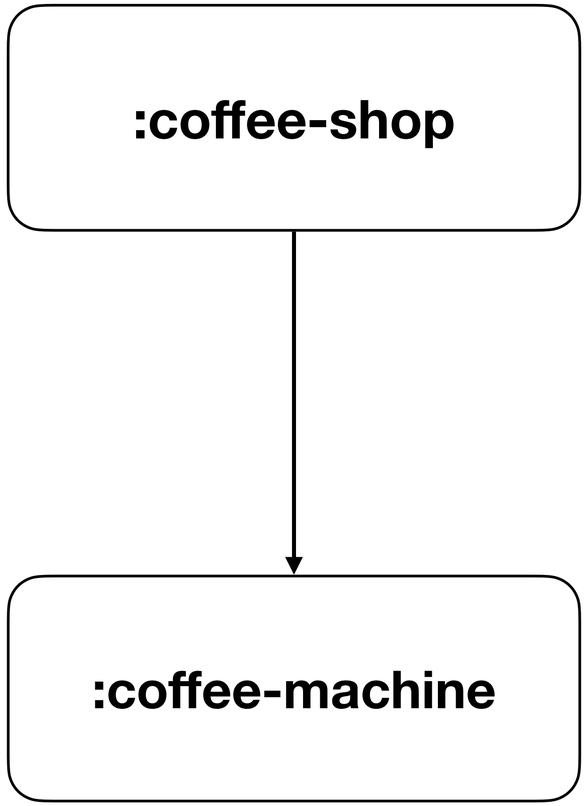
Степени инкрементальности



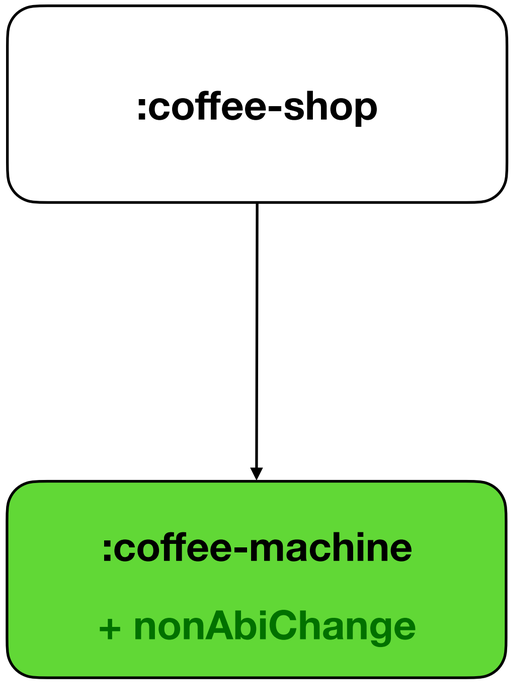
Степени инкрементальности



Java CA: Initial

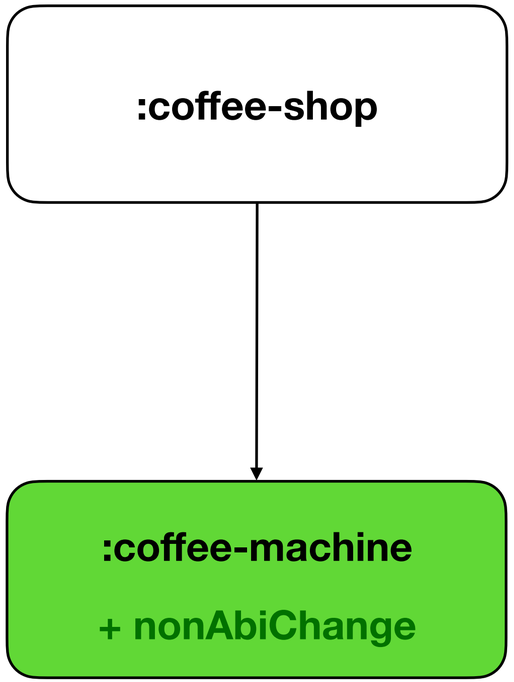


Java CA: Round 1



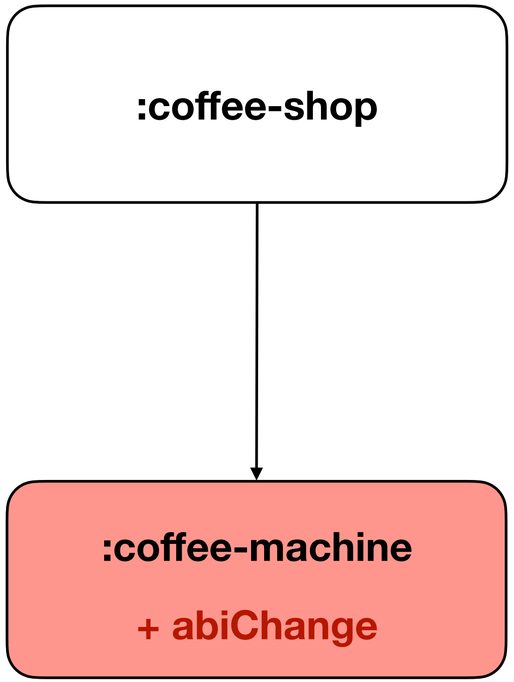

```
> Task :coffee-machine:compileJava
Task ':coffee-machine:compileJava' is not up-to-date because:
Input property 'stableSources' file .../CoffeeMachine.java has changed.

> Task :coffee-shop:compileJava UP-TO-DATE
Skipping task ':coffee-shop:compileJava' as it is up-to-date.
```

Java CA: Round 1



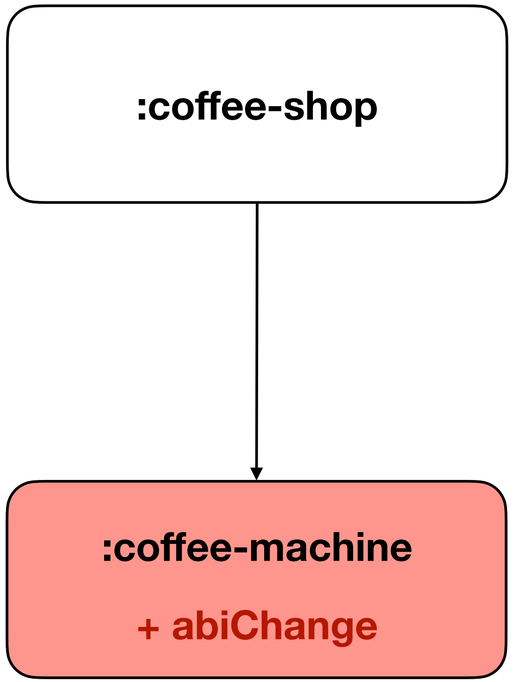

```
> Task :coffee-machine:compileJava
Task ':coffee-machine:compileJava' is not up-to-date because:
Input property 'stableSources' file .../CoffeeMachine.java has changed.

> Task :coffee-shop:compileJava UP-TO-DATE
Skipping task ':coffee-shop:compileJava' as it is up-to-date.
```

Java CA: Round 2



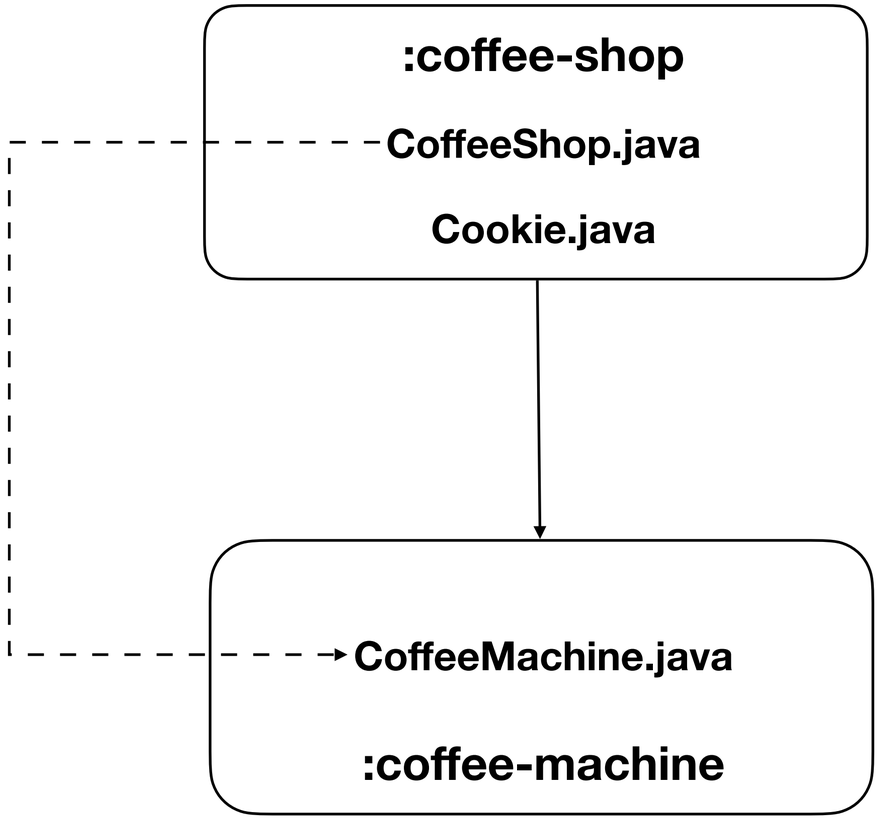

```
> Task :coffee-machine:compileJava
Task ':coffee-machine:compileJava' is not up-to-date because:
Input property 'stableSources' file .../CoffeeMachine.java has changed.

> Task :coffee-shop:compileJava
Task ':coffee-shop:compileJava' is not up-to-date because:
Input property 'classpath' file .../CoffeeMachine.class has changed.
```

Java CA: Round 2

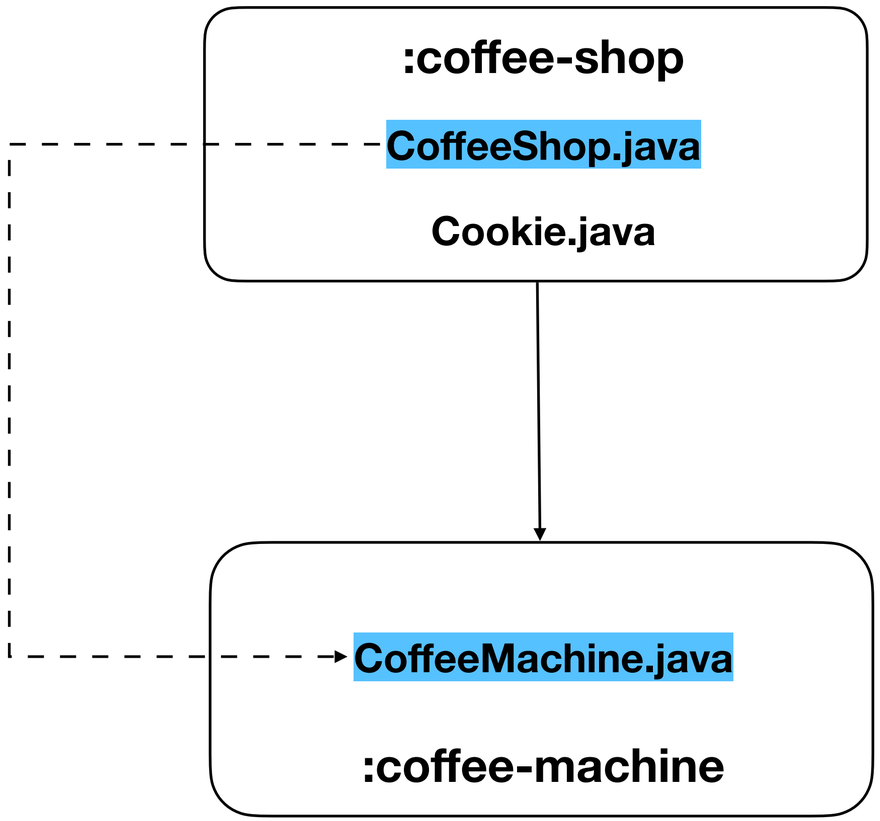



```
> Task :coffee-machine:compileJava
Task ':coffee-machine:compileJava' is not up-to-date because:
Input property 'stableSources' file .../CoffeeMachine.java has changed.

> Task :coffee-shop:compileJava
Task ':coffee-shop:compileJava' is not up-to-date because:
Input property 'classpath' file .../CoffeeMachine.class has changed.
```

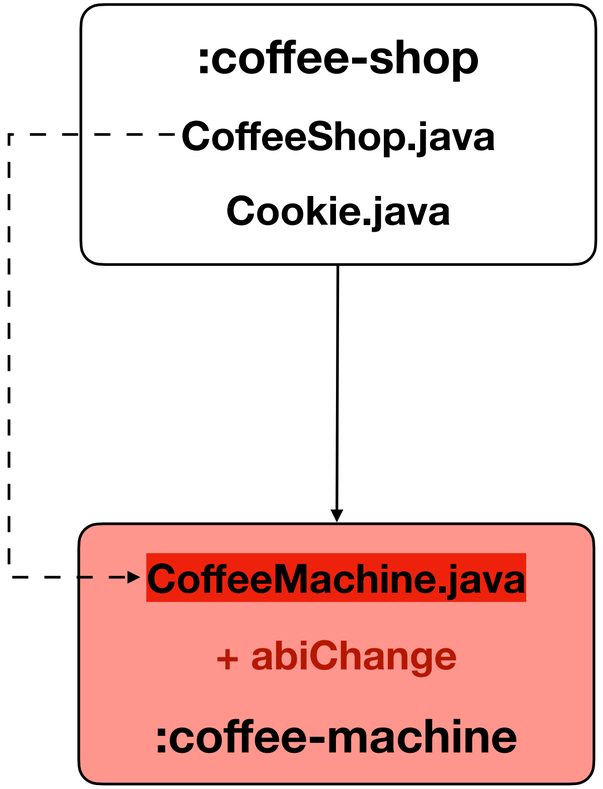
Java CA + IC: Initial



Java CA + IC: Initial

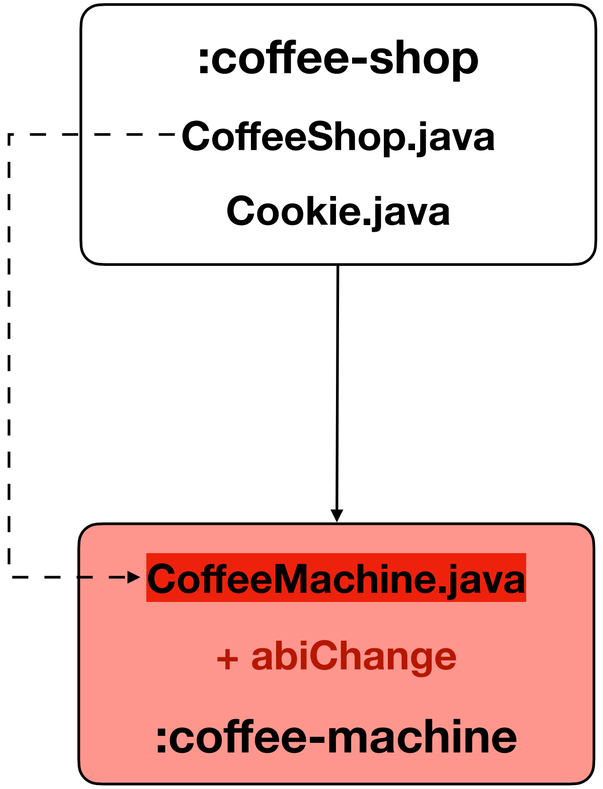


Java CA + IC: Initial



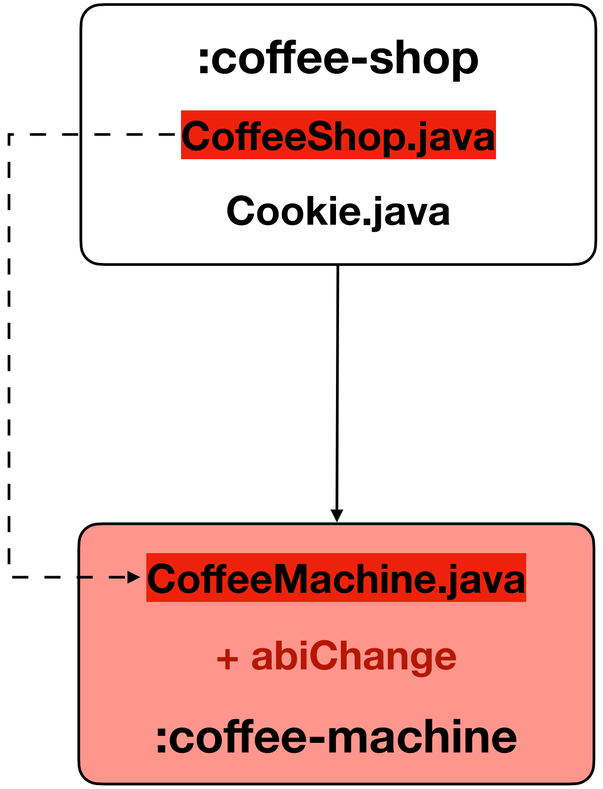

```
> Task :coffee-machine:compileJava
Task ':coffee-machine:compileJava' is not up-to-date because:
  Input property 'stableSources' file .../CoffeeMachine.java has
  changed.

> Task :coffee-shop:compileJava
Task ':coffee-shop:compileJava' is not up-to-date because:
  Input property 'classpath' file .../CoffeeMachine.class has
  changed.
Incremental compilation of 1 classes completed in 0.012 secs.
Recompiled classes [com.ic101.CoffeeShop]
```

Java CA + IC: Initial



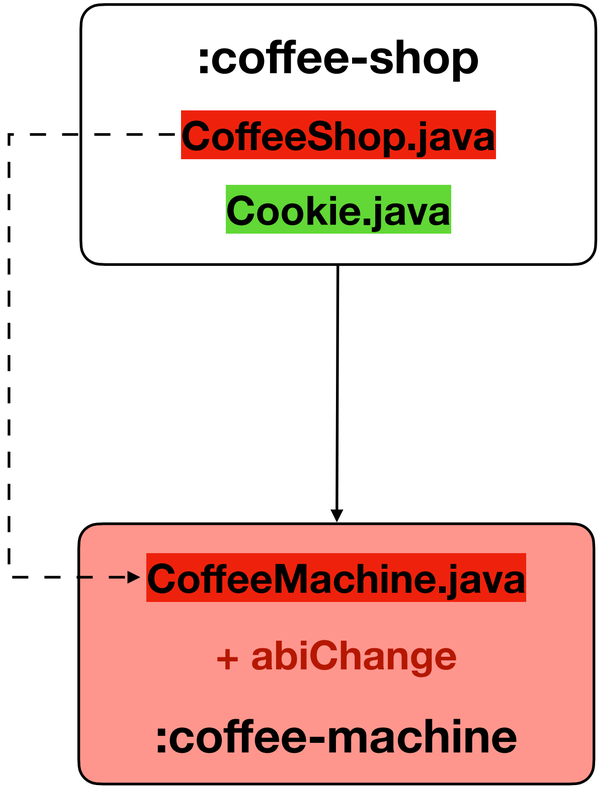

```
> Task :coffee-machine:compileJava
Task ':coffee-machine:compileJava' is not up-to-date because:
  Input property 'stableSources' file .../CoffeeMachine.java has
  changed.

> Task :coffee-shop:compileJava
Task ':coffee-shop:compileJava' is not up-to-date because:
  Input property 'classpath' file .../CoffeeMachine.class has
  changed.
Incremental compilation of 1 classes completed in 0.012 secs.
Recompiled classes [com.ic101.CoffeeShop]
```

Java CA + IC: Initial




```
> Task :coffee-machine:compileJava
Task ':coffee-machine:compileJava' is not up-to-date because:
  Input property 'stableSources' file .../CoffeeMachine.java has
  changed.

> Task :coffee-shop:compileJava
Task ':coffee-shop:compileJava' is not up-to-date because:
  Input property 'classpath' file .../CoffeeMachine.class has
  changed.
Incremental compilation of 1 classes completed in 0.012 secs.
Recompiled classes [com.ic101.CoffeeShop]
```

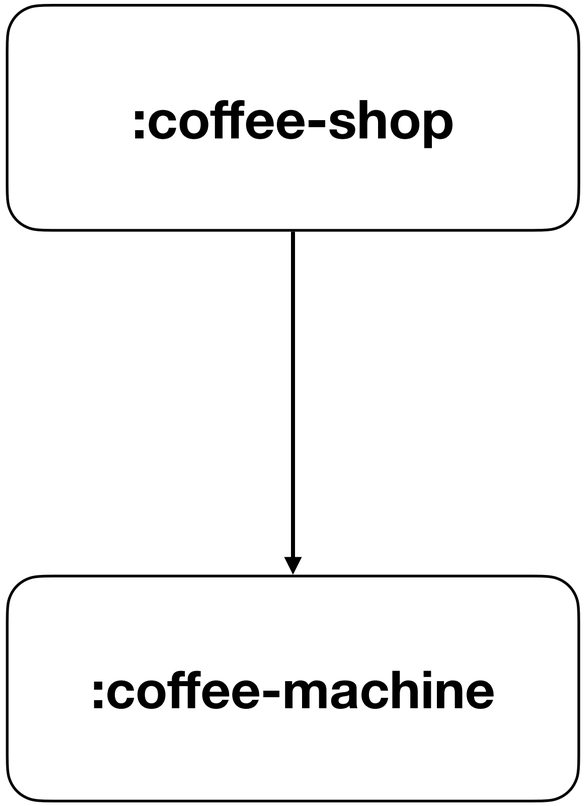
Java CA + IC: Initial



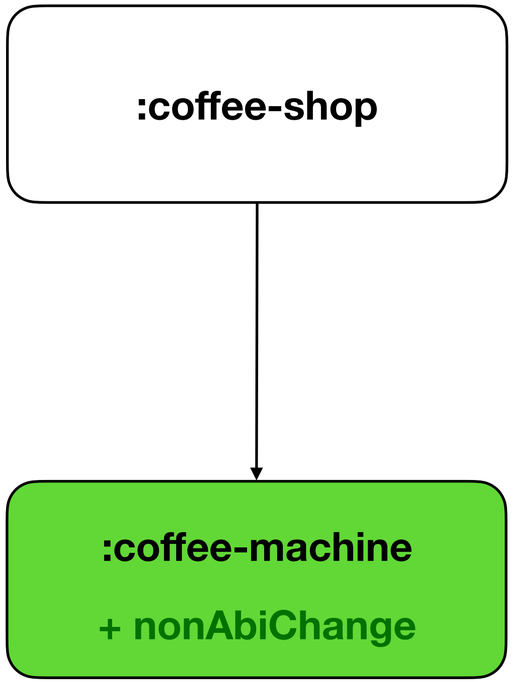

```
> Task :coffee-machine:compileJava
Task ':coffee-machine:compileJava' is not up-to-date because:
  Input property 'stableSources' file .../CoffeeMachine.java has
  changed.

> Task :coffee-shop:compileJava
Task ':coffee-shop:compileJava' is not up-to-date because:
  Input property 'classpath' file .../CoffeeMachine.class has
  changed.
Incremental compilation of 1 classes completed in 0.012 secs.
Recompiled classes [com.ic101.CoffeeShop]
```

Cookie.java не был скомпилирован!

Kotlin CA: Initial

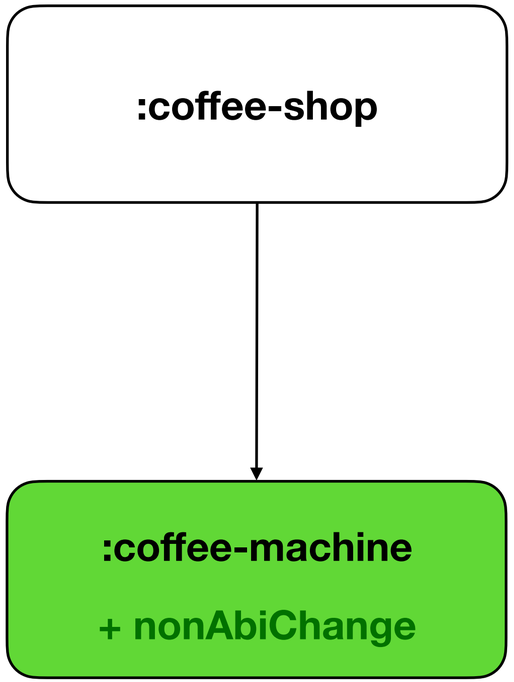


Kotlin CA: Round 1



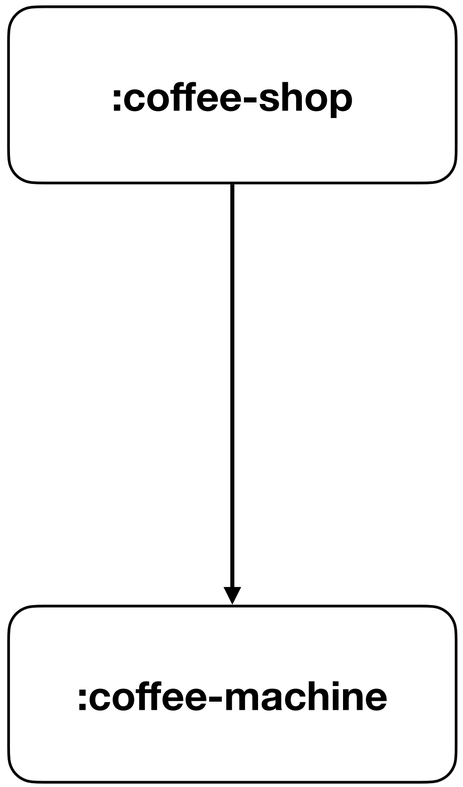

```
> Task :coffee-machine:compileKotlin
Task ':coffee-machine:compileKotlin' is not up-to-date because:
  Input property 'sources' file .../CoffeeMachine.kt has changed.

> Task :coffee-shop:compileKotlin
Task ':coffee-shop:compileKotlin' is not up-to-date because:
  Input property 'classpathSnapshotProperties.classpath'
  file .../CoffeeMachine.class has changed.
```

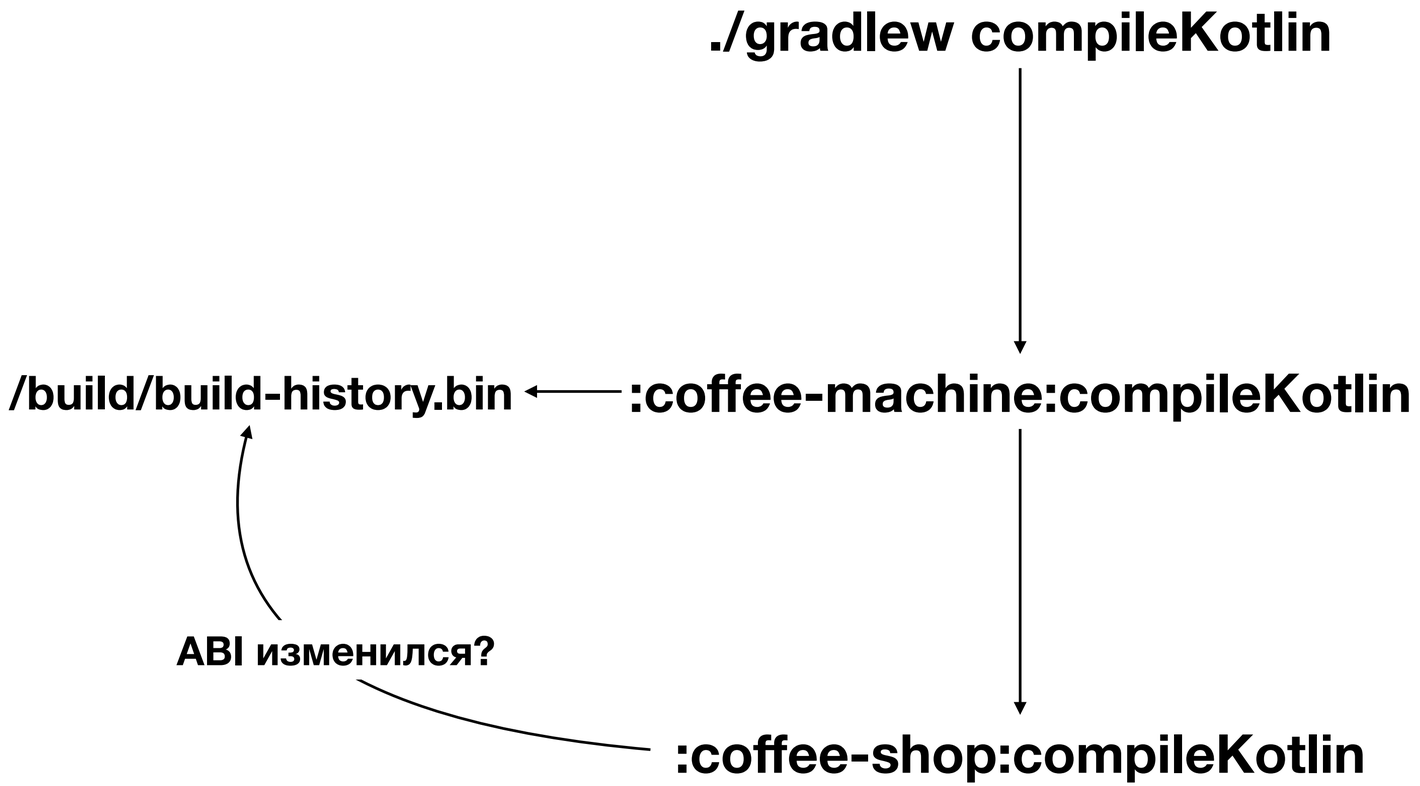
Kotlin CA: Round 1




```
> Task :coffee-machine:compileKotlin
Task ':coffee-machine:compileKotlin' is not up-to-date because:
  Input property 'sources' file .../CoffeeMachine.kt has changed.

> Task :coffee-shop:compileKotlin
Task ':coffee-shop:compileKotlin' is not up-to-date because:
  Input property 'classpathSnapshotProperties.classpath'
  file .../CoffeeMachine.class has changed.
```

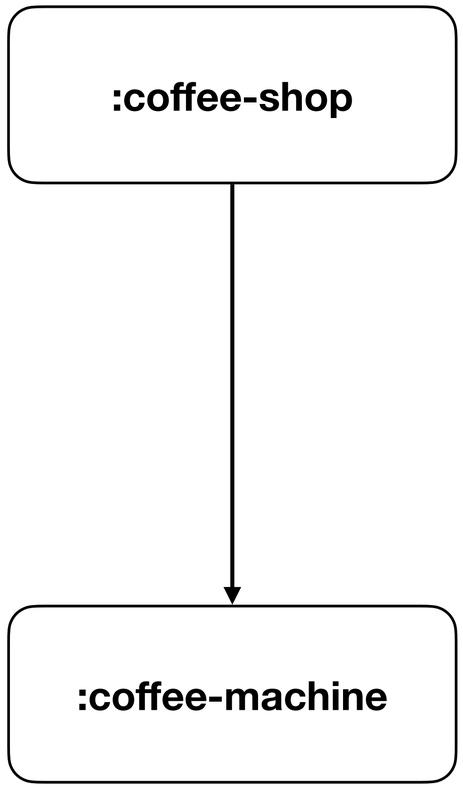
Kotlin CA: старый подход

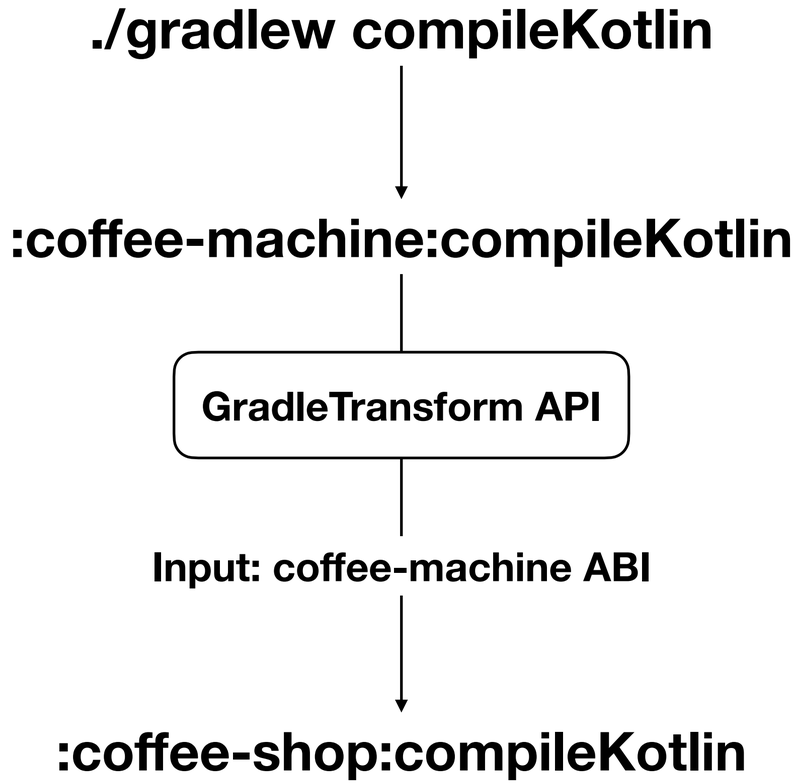




https://blog.jetbrains.com/kotlin/2022/07/a-new-approach-to-incremental-compilation-in-kotlin/

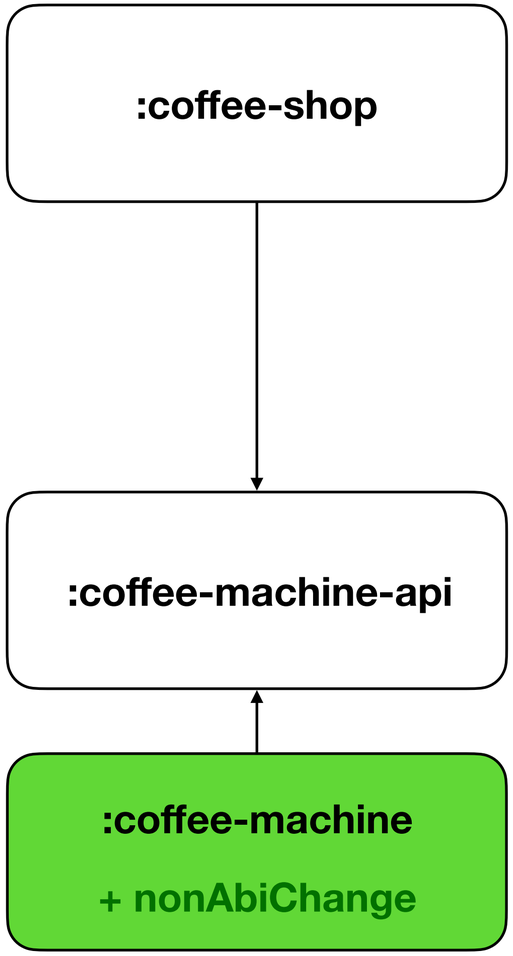
Kotlin CA: useClasspathSnapshot





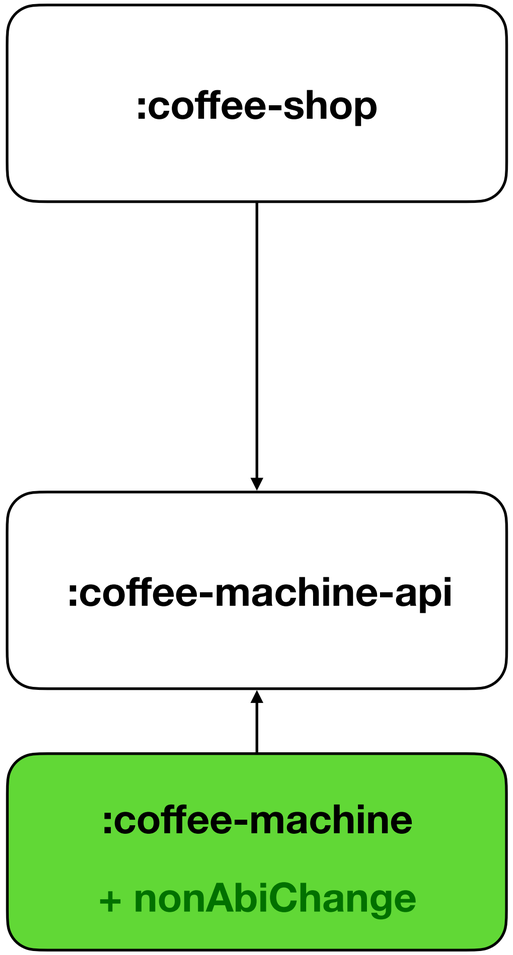
https://blog.jetbrains.com/kotlin/2022/07/a-new-approach-to-incremental-compilation-in-kotlin/

Kotlin CA: Round 1



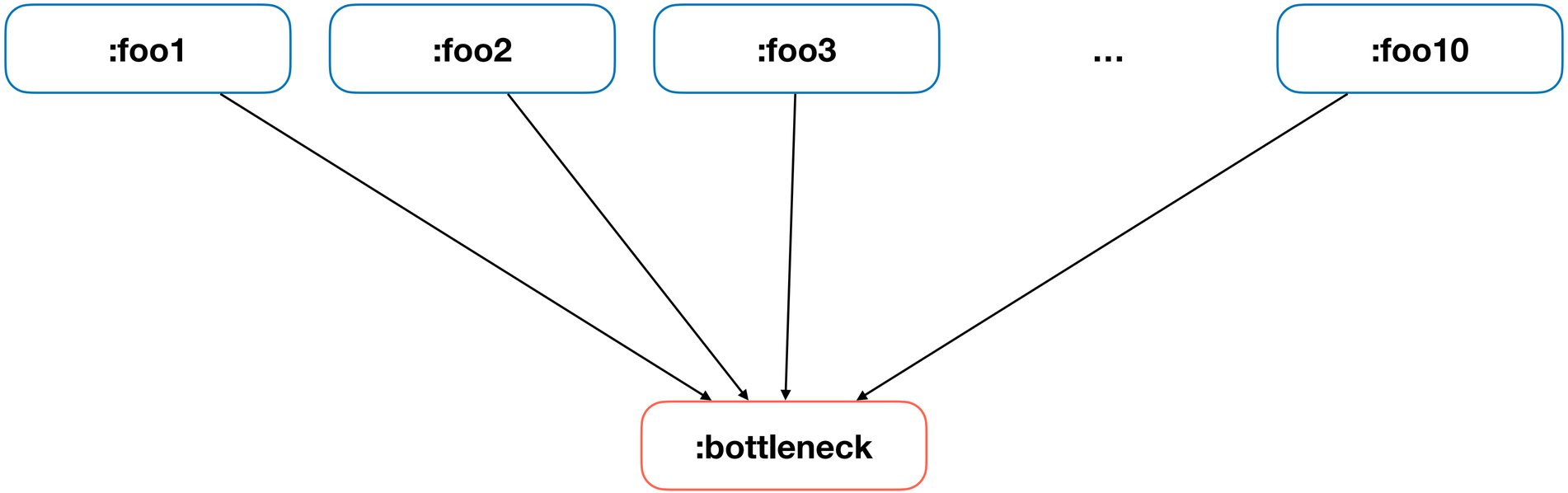

```
> Task :coffee-machine-api:compileKotlin UP-TO-DATE

> Task :coffee-machine:compileKotlin
Task ':coffee-machine:compileKotlin' is not up-to-date because:
  Input property 'sources' file .../CoffeeMachine.kt has changed.

> Task :coffee-shop:compileKotlin UP-TO-DATE
```

Kotlin CA: Round 1




```
> Task :coffee-machine-api:compileKotlin UP-TO-DATE

> Task :coffee-machine:compileKotlin
Task ':coffee-machine:compileKotlin' is not up-to-date because:
  Input property 'sources' file .../CoffeeMachine.kt has changed.

> Task :coffee-shop:compileKotlin UP-TO-DATE
```

Kotlin CA: Синтетический тест



Все модули - Kotlin JVM

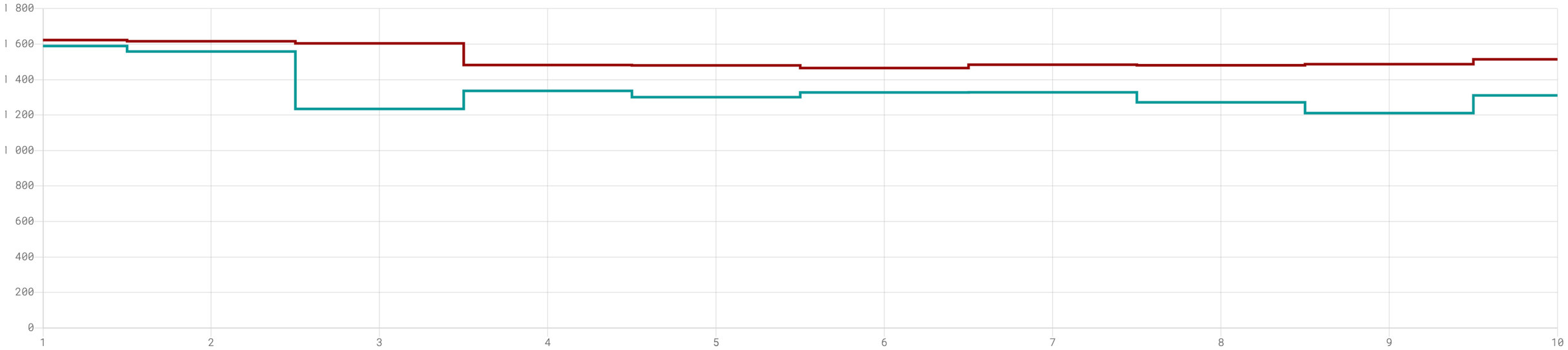
Kotlin CA: Синтетический тест

```
non_abi_change {
    tasks=["assemble"]
    apply-non-abi-change-to = ["bottleneck/src/main/kotlin/Foo0.kt"]
}

non_abi_change_classpath_snapshot {
    tasks=["assemble"]
    apply-non-abi-change-to = ["bottleneck/src/main/kotlin/Foo0.kt"]
    gradle-args = ["-Pkotlin.incremental.useClasspathSnapshot=true"]
}
```

Kotlin CA: Синтетический тест

```
non_abi_change {
    tasks=["assemble"]
    apply-non-abi-change-to = ["bottleneck/src/main/kotlin/Foo0.kt"]
}

non_abi_change_classpath_snapshot {
    tasks=["assemble"]
    apply-non-abi-change-to = ["bottleneck/src/main/kotlin/Foo0.kt"]
    gradle-args = ["-Pkotlin.incremental.useClasspathSnapshot=true"]
}
```

Kotlin CA: Синтетический тест



Сценарий	Медиана
non_abi_change	1 485,40 ms
non_abi_change_classpath_snapshot	1 319,24 ms

$\Delta \approx 12\%$

Итоги

- Java IC работает по class-based эвристикам и не различает ABI/nonABI изменения
- Java CA различает ABI/nonABI изменения, работает эффективно даже без api модулей
- Kotlin IC работает по symbol-based эвристикам, различает ABI/nonABI изменения
- В текущем виде Kotlin CA не различает ABI/nonABI изменения и несовместим с билд кешом
- Недостатки Kotlin CA лечатся с включенным экспериментальным consumer-side анализом useClasspathSnapshot=true (доступно с 1.6.20)
- CA и IC могут и работают вместе и для Java и для Kotlin

Что мы не обсудили?

- Смешанные кодовые базы
- Annotation processing
- Новый компилятор Kotlin K2

Дисклеймер

Все дальнейшие мысли и рекомендации - мои, и не являются официальной позицией команды Gradle Build Tool

Как же развивать свой проект?

1. Определите состояние ваших инструментов
2. Научитесь измерять результаты оптимизаций
3. Найдите эффективные для Вас паттерны
4. Инвестируйте в модуляризацию

Как же развивать свой проект?

1. Определите состояние ваших инструментов
2. Научитесь измерять результаты оптимизаций
3. Найдите эффективные для Вас паттерны => поддерживайте IC
4. Инвестируйте в модуляризацию

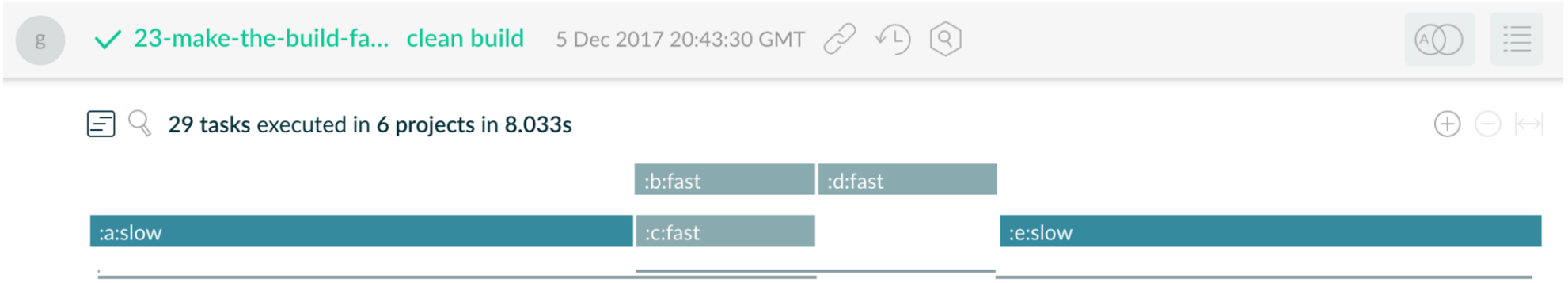
Как же развивать свой проект?

1. Определите состояние ваших инструментов
2. Научитесь измерять результаты оптимизаций
3. Найдите эффективные для Вас паттерны => поддерживайте IC
4. Инвестируйте в модуляризацию => держите фокус на CA

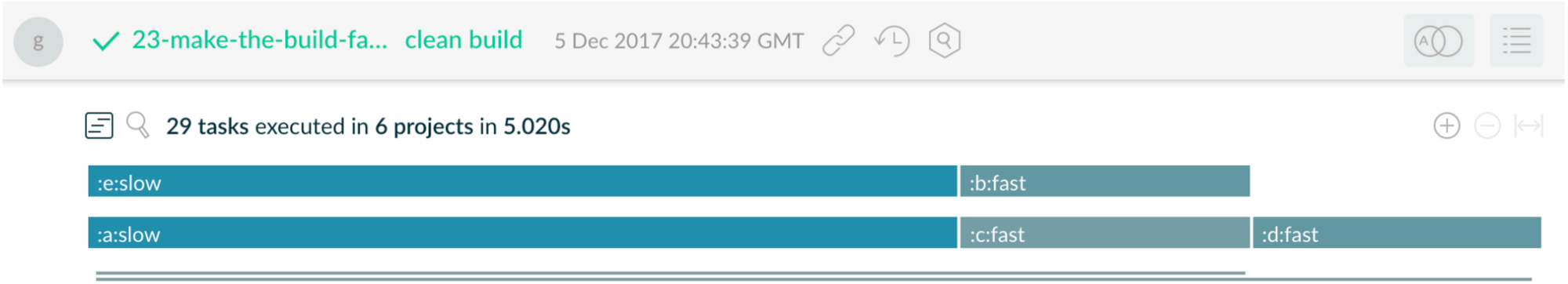
Модуляризация

- Вводит дополнительный уровень абстракции поверх языка программирования
- Снижает связность(coupling) кодовой базы
- Повышает зацепление(cohesion) кодовой базы
- Открывает возможности для новых оптимизаций

Параллельная сборка



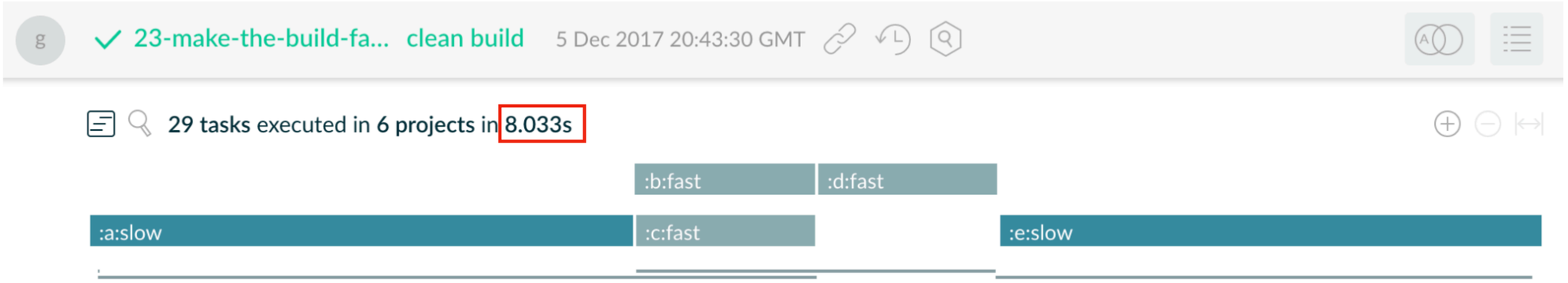
Bottleneck в параллельном выполнении



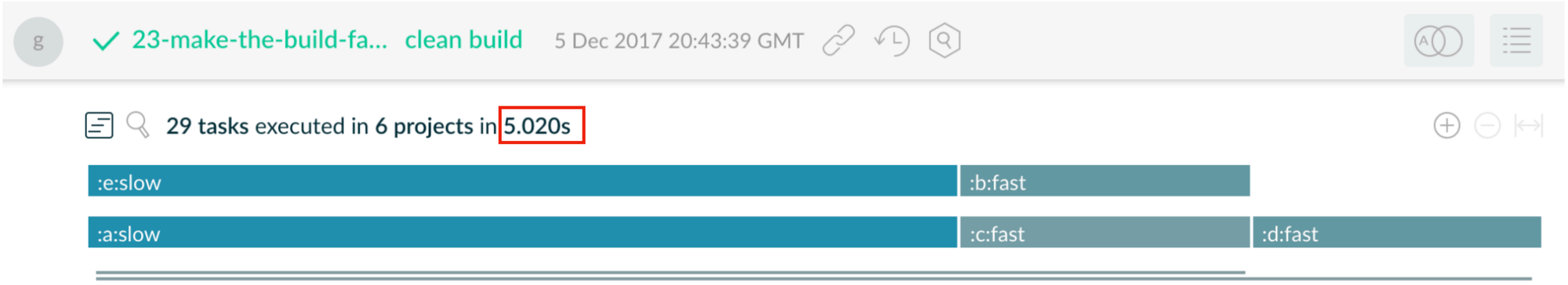
Оптимизированное параллельное выполнение

https://docs.gradle.org/current/userguide/performance.html#parallel_execution

Параллельная сборка



Bottleneck в параллельном выполнении



Оптимизированное параллельное выполнение

https://docs.gradle.org/current/userguide/performance.html#parallel_execution

Параллельная компиляция Java?

https://bugs.java.com/bugdatabase/view_bug.do?bug_id=4229449

Параллельная компиляция Kotlin?

https://blog.jetbrains.com/kotlin/2019/01/kotlin-1-3-20-released#Faster_Gradle_builds_by_parallelizing_tasks

https://kotlinlang.org/docs/whatsnew1620.html#support-for-parallel-compilation-of-a-single-module-in-the-jvm-backend