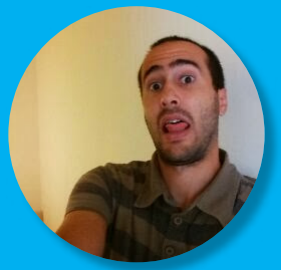
Deep Dive Kotlin : du Hello World au ByteCode

#DevoxxFR #Kotlin @emmanuelVinas @ilaborie

DEVOXX France



Emmanuel Vinas

Expert Android & Java

@emmanuelvinas

emmanuel@monkeypatch.io

Igor Laborie

Expert Java & Web

@ilaborie

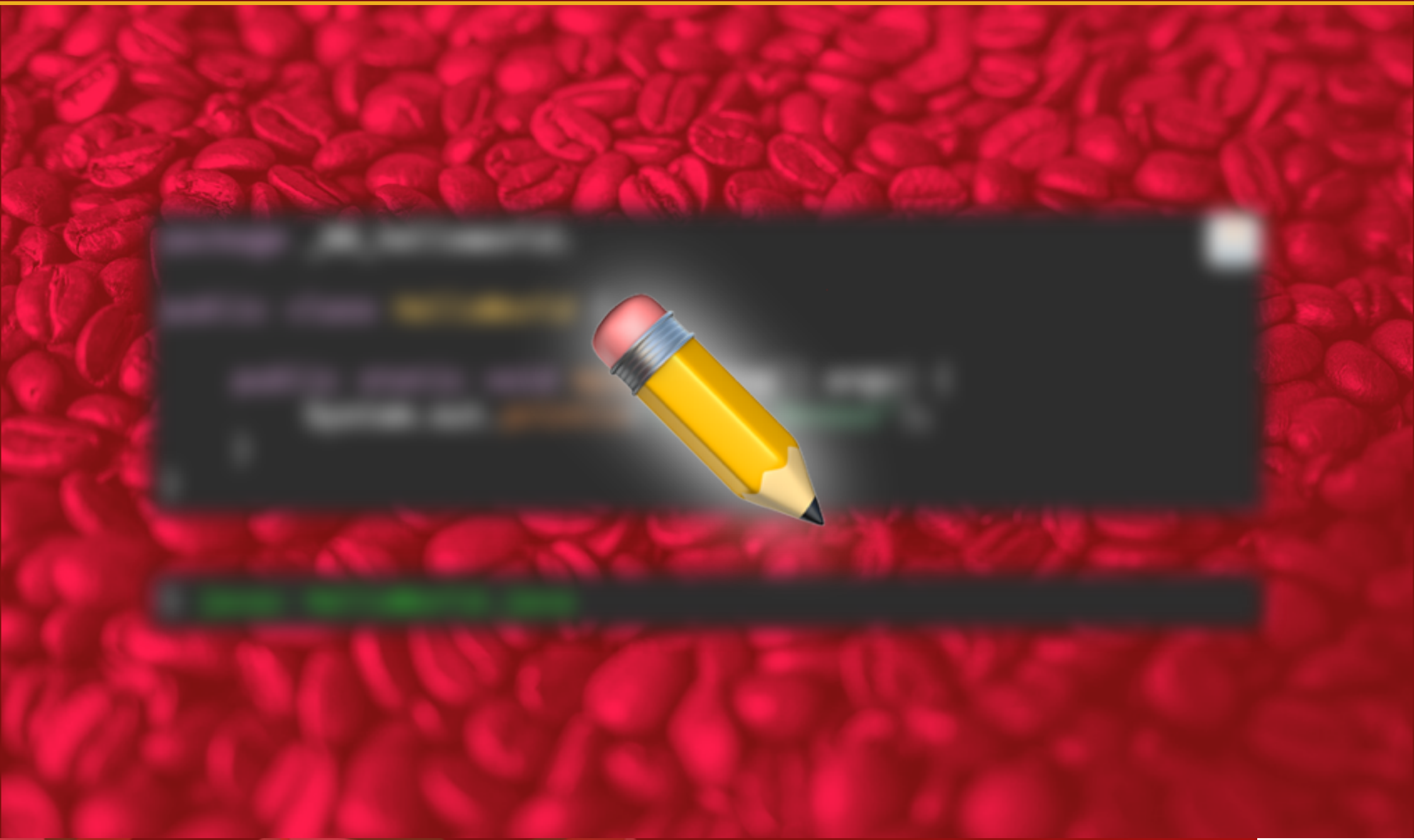
igor@monkeypatch.io

Roadmap

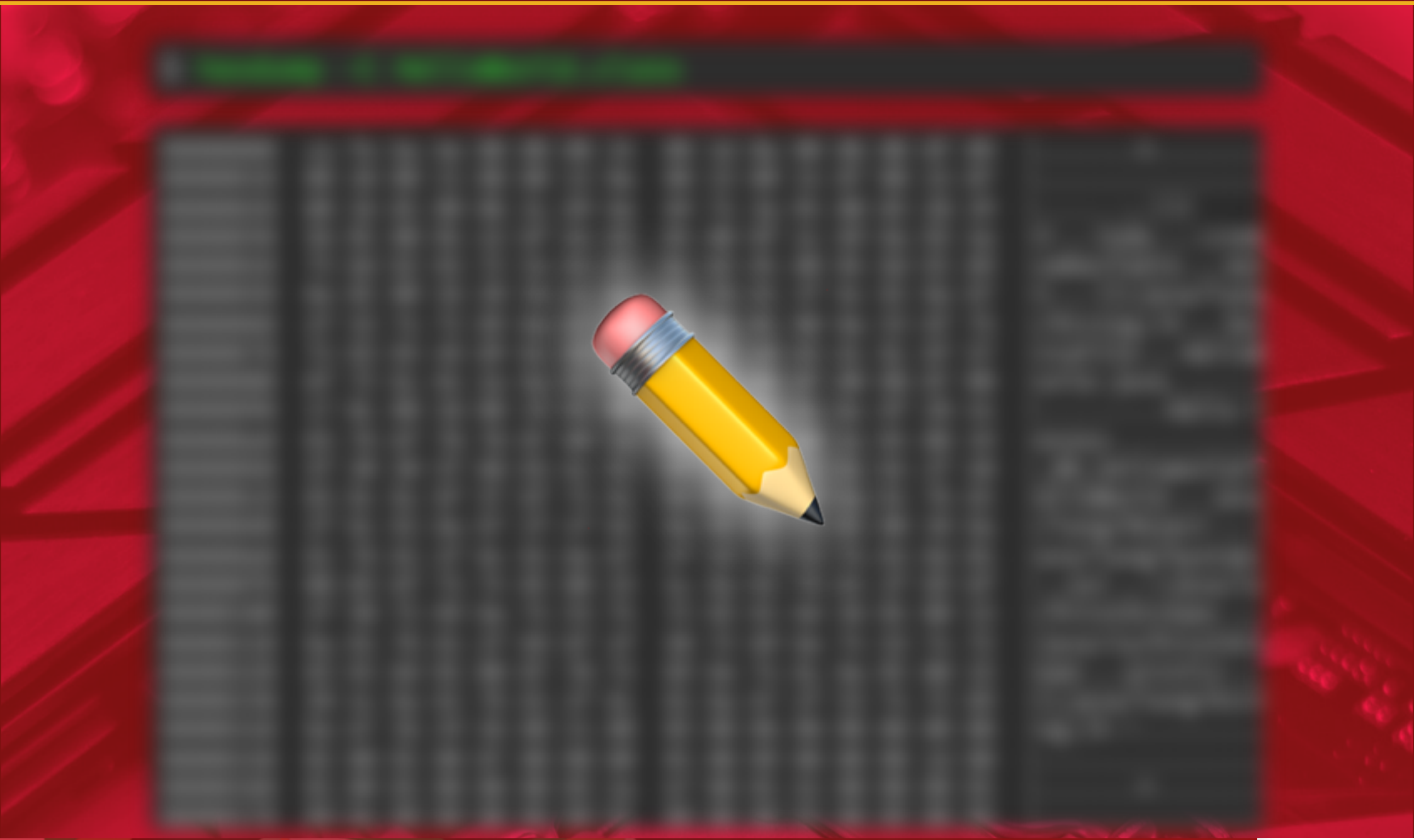
I. ByteCode Java ?
II. Introduction Kotlin
III. Les bases
IV. null-safety
V. Les types
VI. Les fonctions
VII. Les lambdas
VIII. Les classes
IX. Extensions de fonctions
X. Pause
XI. ByteCode Android
XII. Autres structures
XIII. Les collections
XIV. Les delegates
XV. Un peu plus sur les fonctions
XVI. Conclusion

ByteCode Java ?

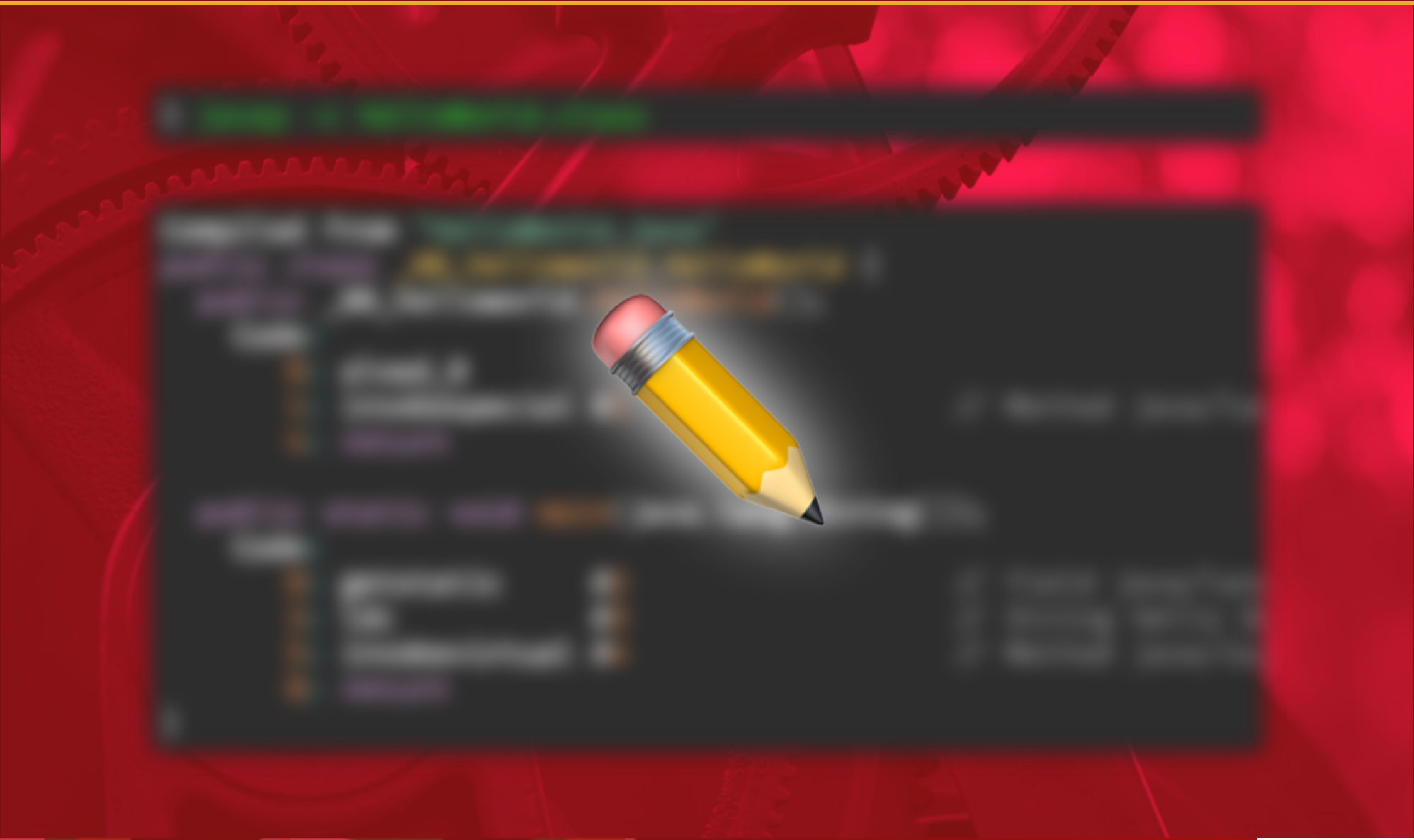
HelloWorld.java



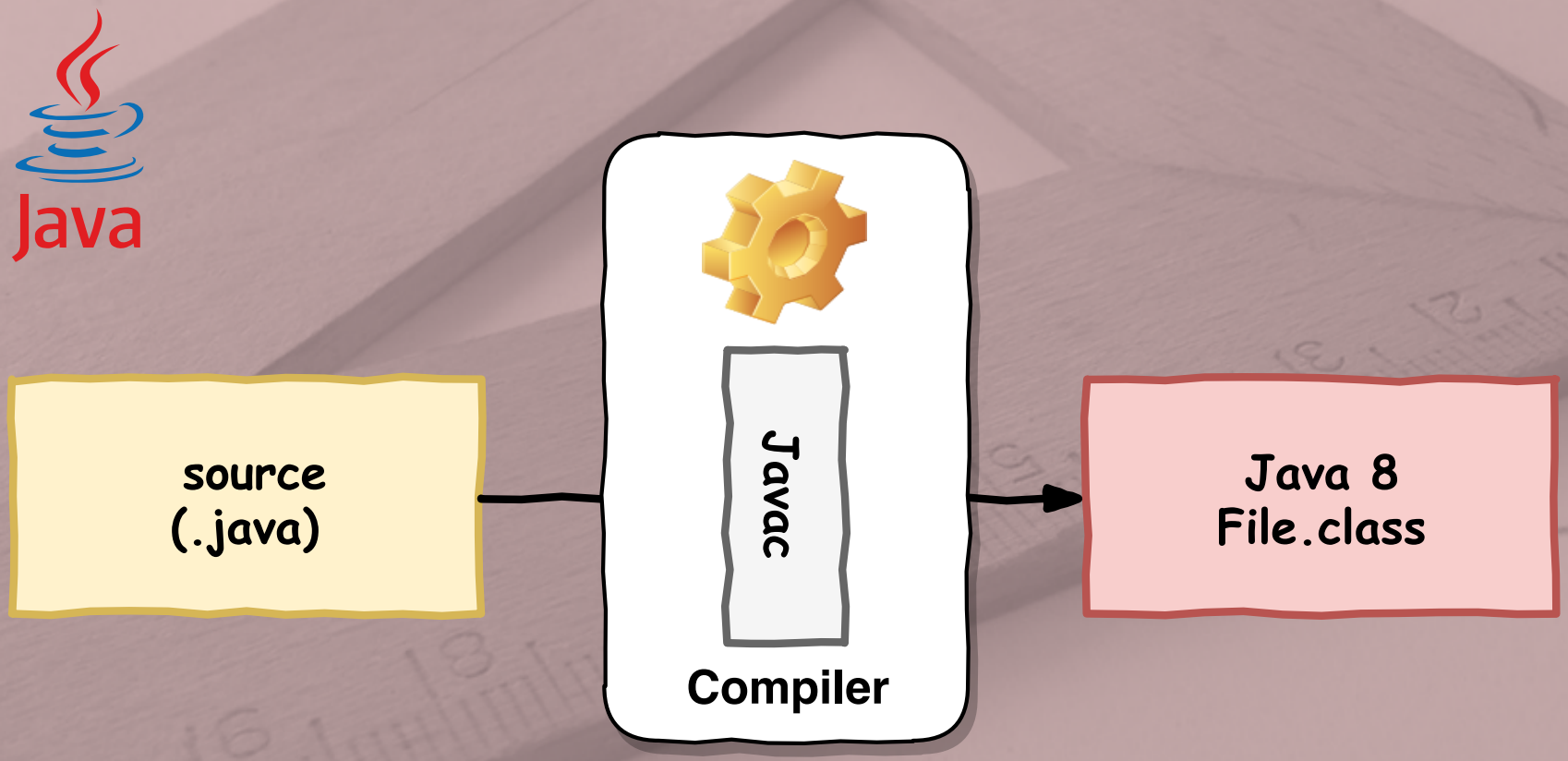
Java ByteCode binary



Explorons le ByteCode



javac



À propos du ByteCode

- environ 200 opérations possibles (maxi. 256 opscodes)
- préfix pour le type d'opérations (`i` pour entier, `d` pour double, ...)
- manipulation de la pile, des variables locales (`iconst_0`, `istore`, `iload`, ...)
- controle du flux des instructions (`if_icmpgt`, `goto`, ...)
- manipulation d'objets (`invokevirtual`, `invokedynamic`, ...)
- arithmétiques et conversion de type (`iadd`, `iinc`, `i2d`, ...)
- autres (`athrow`, ...)

Jouons un peu

▶ Constant Pool

▼ Frames

Liens

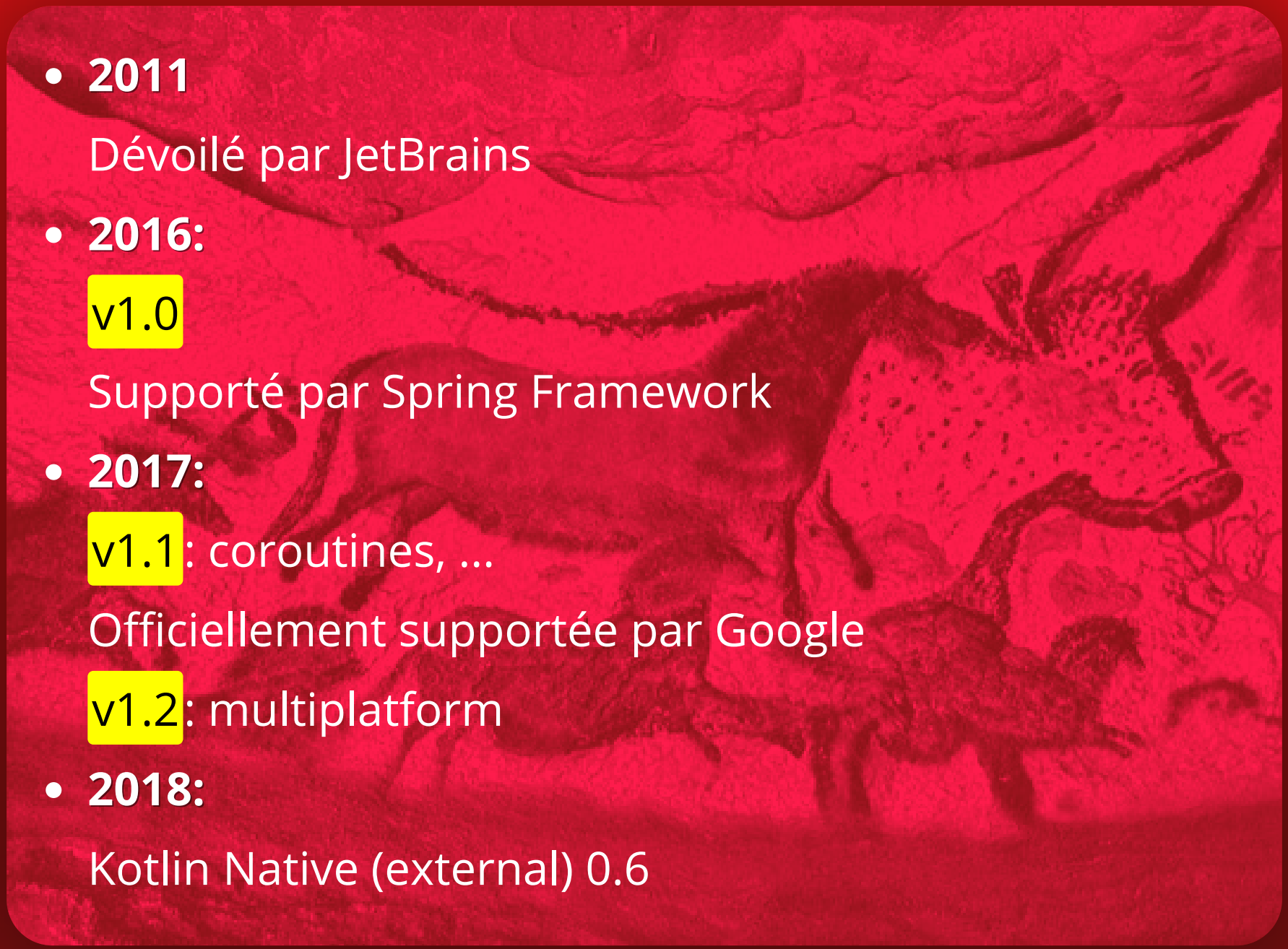
- ➳ Mastering Java Bytecode at the Core of the JVM
- ➳ Introduction to Java Bytecode
- ➳ The Java® Virtual Machine Specification
- ➳ The Java Virtual Machine Instruction Set

- ➳ Byte Buddy
- ➳ asm

> Soyez curieux: regardez comment ça marche avec `javap -c`

Introduction Kotlin

Historique



- **2011**

  Dévoilé par JetBrains
- **2016:**

  v1.0

  Supporté par Spring Framework
- **2017:**

  v1.1: coroutines, ...

  Officiellement supportée par Google

  v1.2: multiplatform
- **2018:**

  Kotlin Native (external) 0.6

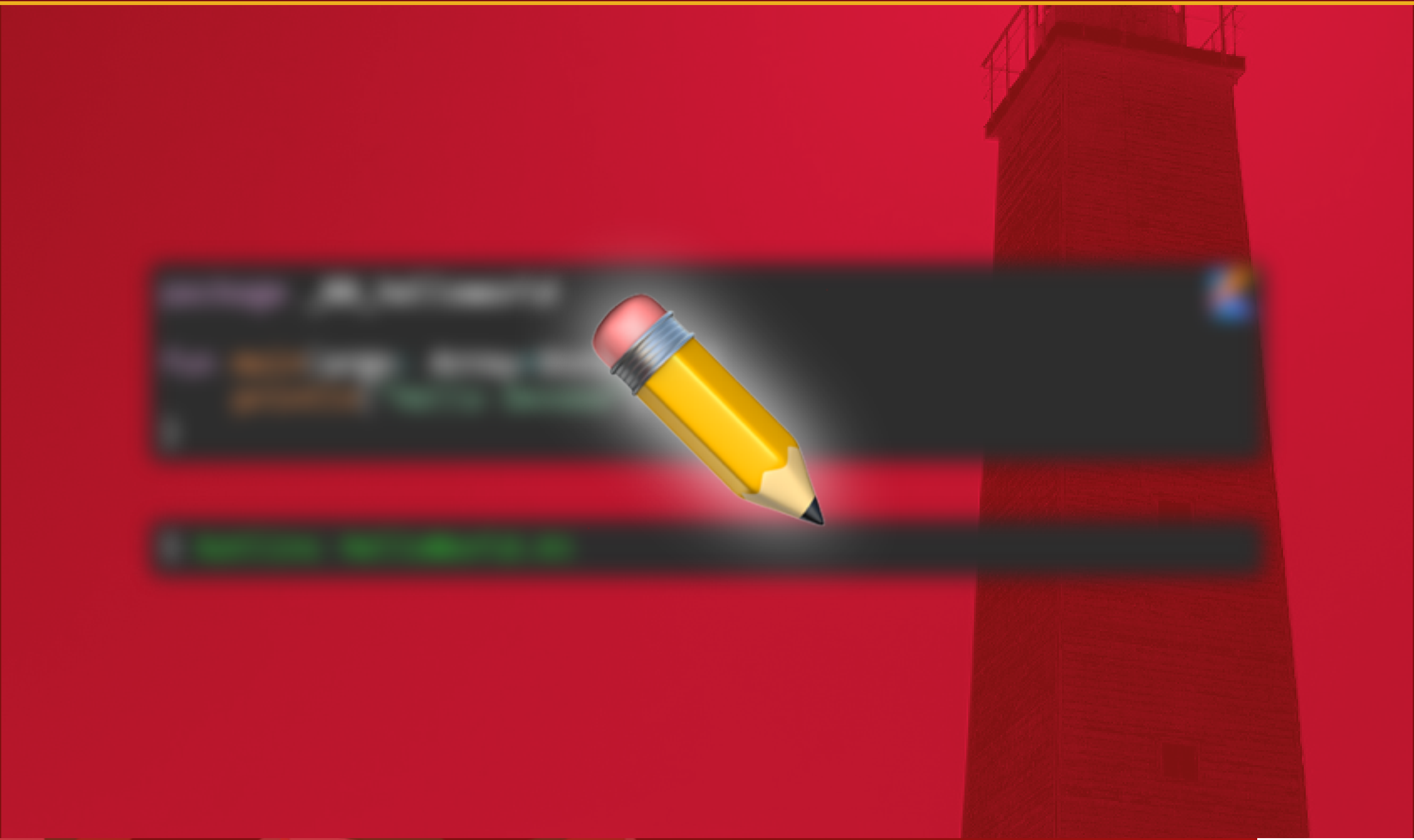
Cibles

JVM et Android

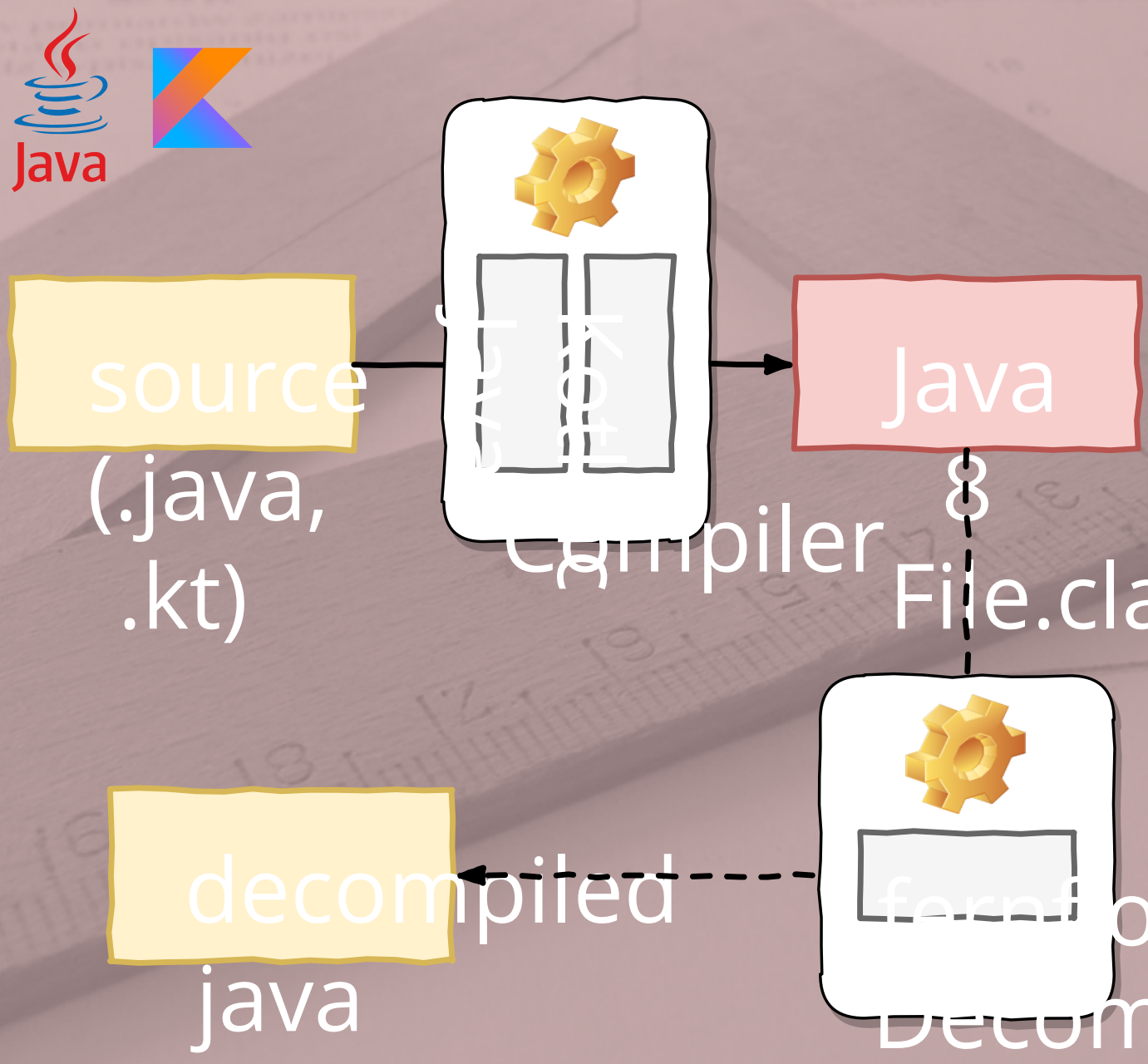
JavaScript

Native avec LLVM

HelloWorld.kt



kotlinc



hexdump


```
00000000  ca fe ba be 00 00 00 32  00 33 01 00 1b 5f 30 30  |.......2.3..._00|
00000010  5f 68 65 6c 6c 6f 77 6f  72 6c 64 2f 48 65 6c 6c  |_helloworld/Hell|
00000020  6f 57 6f 72 6c 64 4b 74  07 00 01 01 00 10 6a 61  |oWorldKt......ja|
00000030  76 61 2f 6c 61 6e 67 2f  4f 62 6a 65 63 74 07 00  |va/lang/Object..|
00000040  03 01 00 04 6d 61 69 6e  01 00 16 28 5b 4c 6a 61  |....main...([Lja|
00000050  76 61 2f 6c 61 6e 67 2f  53 74 72 69 6e 67 3b 29  |va/lang/String;)|
00000060  56 01 00 23 4c 6f 72 67  2f 6a 65 74 62 72 61 69  |V..#Lorg/jetbrai|
00000070  6e 73 2f 61 6e 6e 6f 74  61 74 69 6f 6e 73 2f 4e  |ns/annotations/N|
00000080  6f 74 4e 75 6c 6c 3b 01  00 04 61 72 67 73 08 00  |otNull;...args..|
00000090  08 01 00 1e 6b 6f 74 6c  69 6e 2f 6a 76 6d 2f 69  |....kotlin/jvm/i|
000000a0  6e 74 65 72 6e 61 6c 2f  49 6e 74 72 69 6e 73 69  |nternal/Intrinsi|
000000b0  63 73 07 00 0a 01 00 17  63 68 65 63 6b 50 61 72  |cs......checkPar|
000000c0  61 6d 65 74 65 72 49 73  4e 6f 74 4e 75 6c 6c 01  |ameterIsNotNull.|
000000d0  00 27 28 4c 6a 61 76 61  2f 6c 61 6e 67 2f 4f 62  |.'(Ljava/lang/Ob|
000000e0  6a 65 63 74 3b 4c 6a 61  76 61 2f 6c 61 6e 67 2f  |ject;Ljava/lang/|
000000f0  53 74 72 69 6e 67 3b 29  56 0c 00 0c 00 0d 0a 00  |String;)V.......|
00000100  0b 00 0e 01 00 0c 48 65  6c 6c 6f 20 44 65 76 6f  |......Hello Devo|
00000110  78 78 08 00 10 01 00 10  6a 61 76 61 2f 6c 61 6e  |xx......java/lan|
00000120  67 2f 53 79 73 74 65 6d  07 00 12 01 00 03 6f 75  |g/System......ou|
00000130  74 01 00 15 4c 6a 61 76  61 2f 69 6f 2f 50 72 69  |t...Ljava/io/Pri|
00000140  6e 74 53 74 72 65 61 6d  3b 0c 00 14 00 15 09 00  |ntStream;.......|
00000150  13 00 16 01 00 13 6a 61  76 61 2f 69 6f 2f 50 72  |......java/io/Pr|
00000160  69 6e 74 53 74 72 65 61  6d 07 00 18 01 00 07 70  |intStream......p|
00000170  72 69 6e 74 6c 6e 01 00  15 28 4c 6a 61 76 61 2f  |rintln...(Ljava/|
00000180  6c 61 6e 67 2f 4f 62 6a  65 63 74 3b 29 56 0c 00  |lang/Object;)V..|
00000190  1a 00 1b 0a 00 19 00 1c  01 00 13 5b 4c 6a 61 76  |...........[Ljav|
000001a0  61 2f 6c 61 6e 67 2f 53  74 72 69 6e 67 3b 01 00  |a/lang/String;..|
000001b0  11 4c 6b 6f 74 6c 69 6e  2f 4d 65 74 61 64 61 74  |.Lkotlin/Metadat|
```

Java ByteCode


```
Compiled from "HelloWorld.kt"
public final class _00_helloworld.HelloWorldKt {
  public static final void main(java.lang.String[]);
    Code:
       0: aload_0
       1: ldc           #9                  // String args
       3: invokestatic  #15                 // Method kotlin/
       6: ldc           #17                 // String Hello D
       8: astore_1
       9: getstatic     #23                 // Field java/lan
      12: aload_1
      13: invokevirtual #29                 // Method java/io/
      16: return
}
```

HelloWorld-java

```java
package _00_helloworld;

import kotlin.Metadata;
import kotlin.jvm.internal.Intrinsics;
import org.jetbrains.annotations.NotNull;

@Metadata(
   mv = {1, 1, 9},
   bv = {1, 0, 2},
   k = 2,
   d1 = {"\u0000\u0012\n\u0000\n\u0002\u0010\u0002\n\u0000\n\u
   d2 = {"main", "", "args", "", "", "([Ljava/lang/String;)V"
)
public final class HelloWorldKt {
   public static final void main(@NotNull String[] args) {
      Intrinsics.checkParameterIsNotNull(args, "args");
      String var1 = "Hello Devoxx";
      System.out.println(var1);
   }
}
```

Bilan HelloWorld.kt

- Kotlin ajoute des contrôles
- du coup on a besoin de JARs en plus

jar	taille
kotlin-stdlib-1.2.31.jar	919K
kotlin-stdlib-jdk7-1.2.31.jar	3.1K
kotlin-stdlib-jdk8-1.2.31.jar	13K
kotlin-reflect-1.2.31.jar	2.5M
guava-18.0.jar	2.2M
lombok-1.16.18.jar	1.4M
spring-core-5.0.5.RELEASE.jar	1.2M
jackson-databind-2.9.5.jar	1.3M

- Performances ?

Performance HelloWorld.kt

> Ne croyez pas les benchmarks, faites les vous-même !

- https://github.com/JetBrains/kotlin-benchmarks
- https://github.com/MonkeyPatchIo/kotlin-perf

Benchmark	Mode	Cnt	Score	Error	Units
testJava	thrpt	200	66490.271	± 879.996	ops/s
testKotlin	thrpt	200	72393.914	± 935.962	ops/s

Les bases

val-var.kt


```kotlin
var x: Int = 10
val y: Int = 3
x += 4
// y += 4 ⟸ 💣 Compilation Error

println(x * y) // 42
```

string-template.kt


```
fun greeting(who: Someone) {
    println("Hello $who!")
    println("Hello ${who.firstName} ${who.lastName}!")
}
```

string-template.java

```java
package _01_basic;

import kotlin.Metadata;
import kotlin.jvm.internal.Intrinsics;
import org.jetbrains.annotations.NotNull;

@Metadata(
   mv = {1, 1, 9},
   bv = {1, 0, 2},
   k = 2,
   d1 = {"\u0000\f\n\u0000\n\u0002\u0010\u0002\n\u0000\n\u000
   d2 = {"greeting", "", "who", "L_01_basic/Someone;"}
)
public final class String_templatesKt {
   public static final void greeting(@NotNull Someone who) {
      Intrinsics.checkParameterIsNotNull(who, "who");
      String var1 = "Hello " + who + '!';
      System.out.println(var1);
      var1 = "Hello " + who.getFirstName() + ' ' + who.getLas
      System.out.println(var1);
   }
}
```

ByteCode de string-template


```
Compiled from "string-templates.kt"
public final class _01_basic.String_templatesKt {
  public static final void greeting(_01_basic.Someone);
    Code:
       0: aload_0
       1: ldc           #9                  // String who
       3: invokestatic  #15                 // Method kotlin/
       6: new           #17                 // class java/lang
       9: dup
      10: invokespecial #21                 // Method java/la
      13: ldc           #23                 // String Hello
      15: invokevirtual #27                 // Method java/la
      18: aload_0
      19: invokevirtual #30                 // Method java/la
      22: bipush        33
      24: invokevirtual #33                 // Method java/la
      27: invokevirtual #37                 // Method java/la
      30: astore_1
      31: getstatic     #43                 // Field java/lang
      34: aload_1
      35: invokevirtual #49                 // Method java/io/
```

numeric.kt


```
val anInt = 42  // type inference: Int
val aLong = 42L // type inference: Long
var aDouble: Double? = null
```

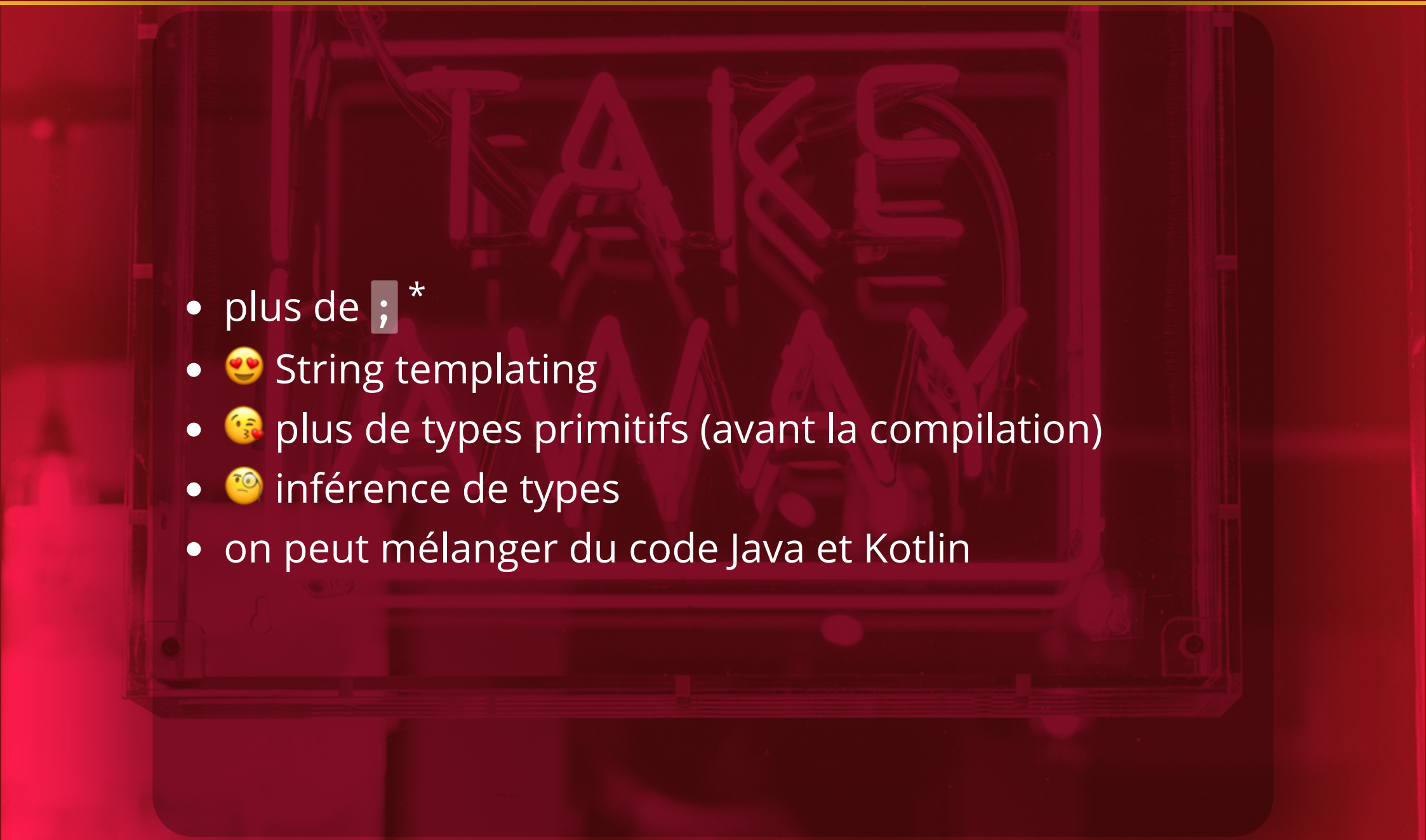
numeric.java

```java
package _01_basic;

import kotlin.Metadata;

@Metadata(
   mv = {1, 1, 9},
   bv = {1, 0, 2},
   k = 2,
   d1 = {"\u0000\u0006\n\u0000\n\u0002\u0010\u0002\u001a\u0000
   d2 = {"tryNumeric", ""}
)
public final class NumericKt {
   public static final void tryNumeric() {
      int anInt = true;
      long aLong = 42L;
      Double aDouble = (Double)null;
   }
}
```

ByteCode de numeric


```
Compiled from "numeric.kt"
public final class _01_basic.NumericKt {
  public static final void tryNumeric();
    Code:
       0: bipush        42
       2: istore_0
       3: ldc2_w        #7                  // long 42l
       6: lstore_1
       7: aconst_null
       8: checkcast     #10                 // class java/lan
      11: astore_3
      12: return
}
```

C'est simple



- plus de ; *
- 😍 String templating
- 😘 plus de types primitifs (avant la compilation)
- 🧐 inférence de types
- on peut mélanger du code Java et Kotlin

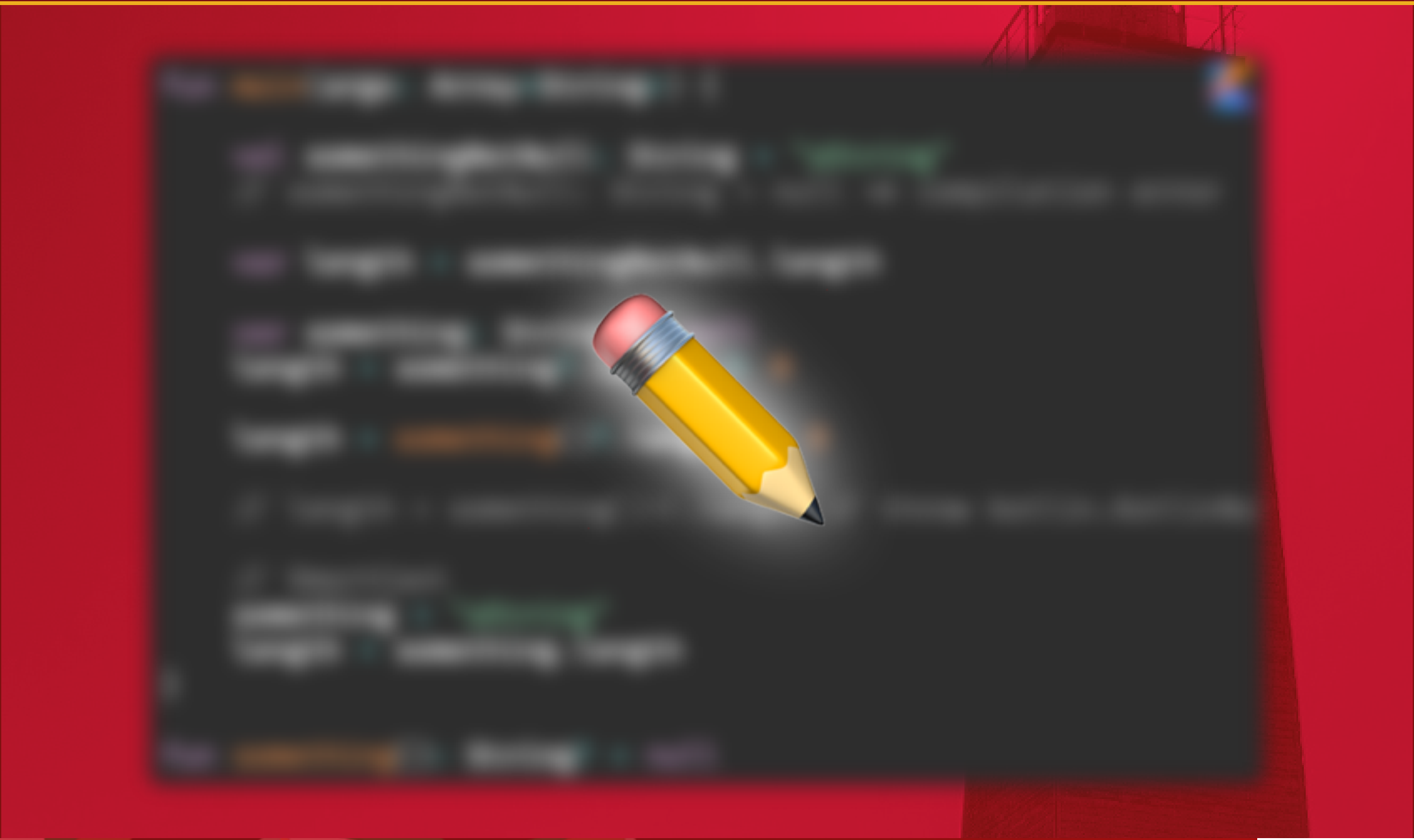
null-safety

billion-dollar mistake

> I call it my billion-dollar mistake. It was the invention of the `null` reference in 1965. At that time, I was designing the first comprehensive type system for references in an object oriented language (ALGOL W). My goal was to ensure that all use of references should be absolutely safe, with checking performed automatically by the compiler. But I couldn't resist the temptation to put in a `null` reference, simply because it was so easy to implement. This has led to innumerable errors, vulnerabilities, and system crashes, which have probably caused a billion dollars of pain and damage in the last forty years.
>
> --Tony Hoare (C.A.R. Hoare)

➳ Null References: The Billion Dollar Mistake

null-safety.kt



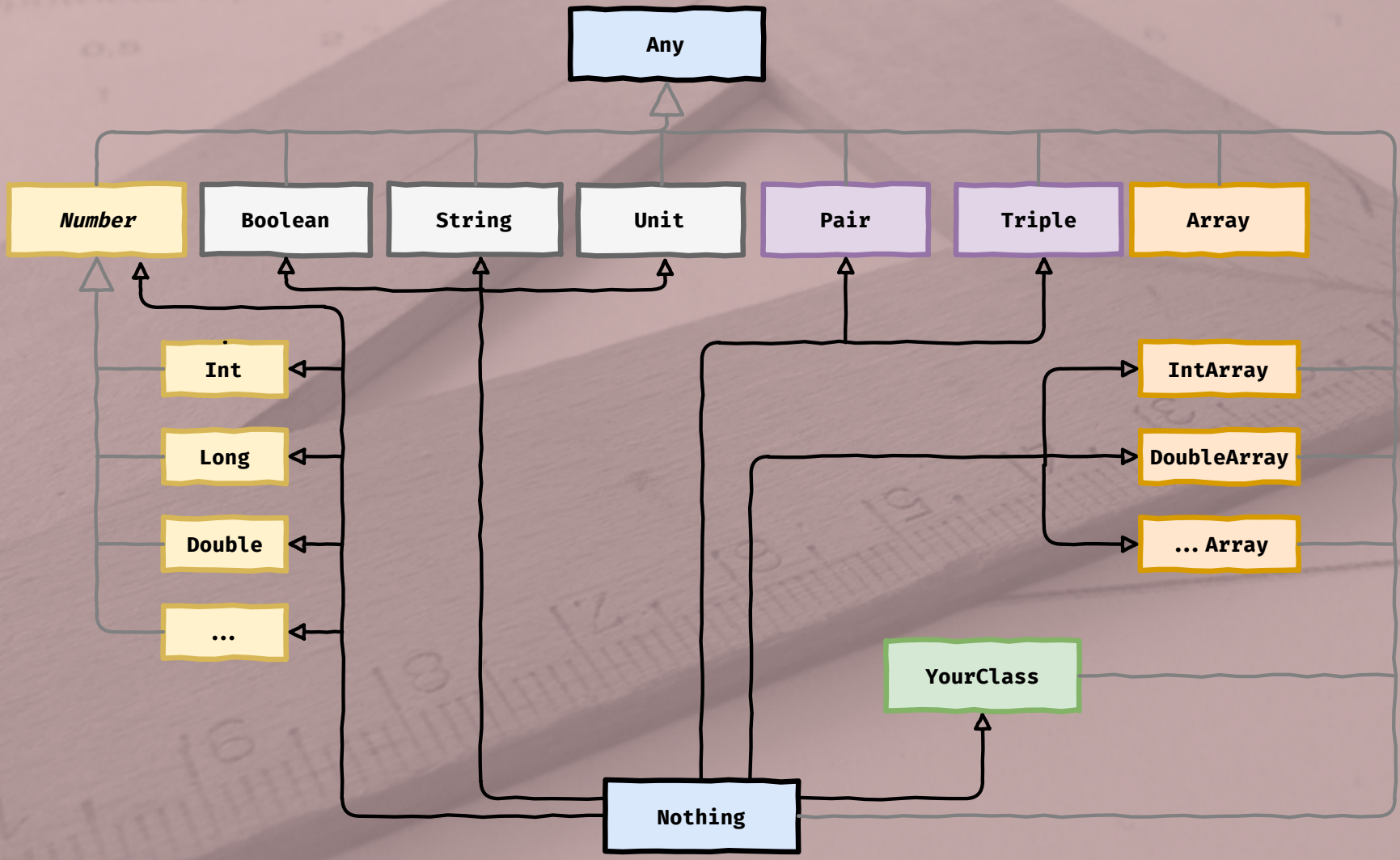
null-safety.java

```
package _02_null_safety;

import kotlin.Metadata;
import kotlin.jvm.internal.Intrinsics;
import org.jetbrains.annotations.NotNull;
import org.jetbrains.annotations.Nullable;

@Metadata(
   mv = {1, 1, 9},
   bv = {1, 0, 2},
   k = 2,
   d1 = {"\u0000\u0014\n\u0000\n\u0002\u0010\u0002\n\u0000\n\u
   d2 = {"main", "", "args", "", "", "([Ljava/lang/String;)V"
)
public final class NullSafetyKt {
   public static final void main(@NotNull String[] args) {
      Intrinsics.checkParameterIsNotNull(args, "args");
      String somethingNotNull = "aString";
      int length = somethingNotNull.length();
      String something = (String)null;
      int length = false;
```

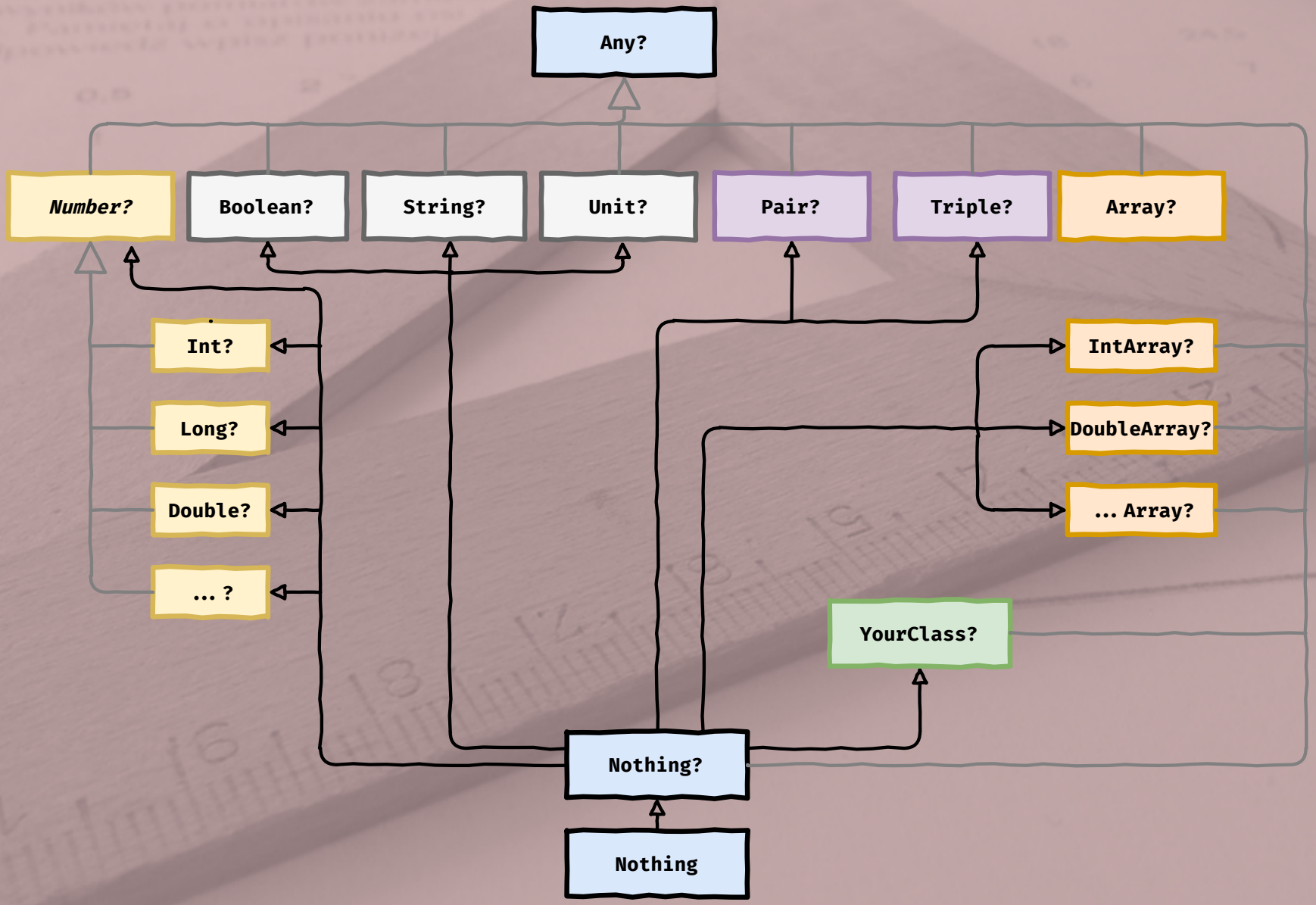
- plus de `NullPointerException`
- ⚠️ quand on appel du Java

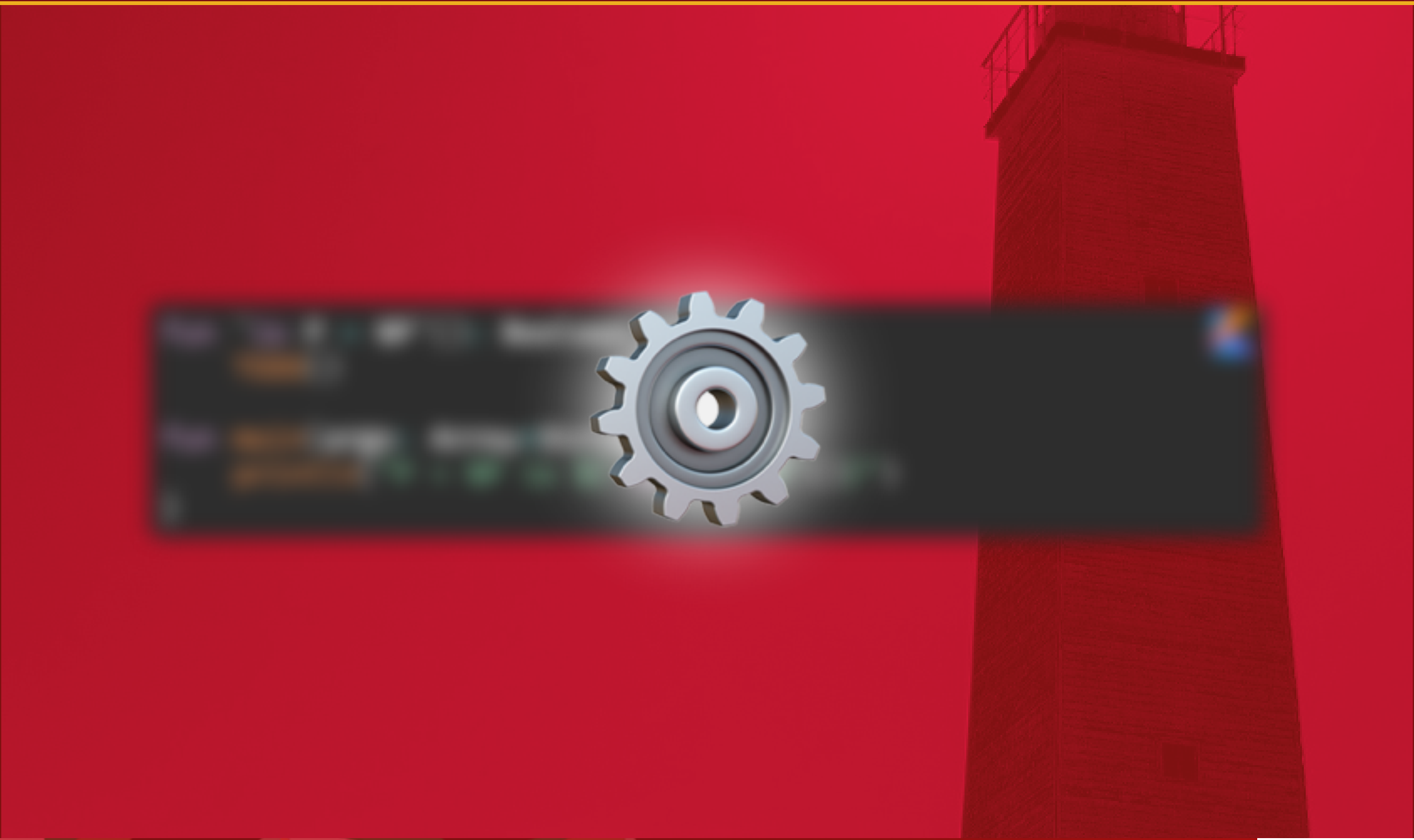
Les types

Types basiques



Types basiques nullable





Hierarchie des types

- 🤝 le `TODO()` est l'ami du TDD

Les fonctions

named-params.kt

```kotlin
fun buildString(prefix: String,
                who: String,
                enhanced: Boolean): String {
    var msg = "$prefix $who"
    if (enhanced) {
        msg += '!'
    }
    return msg
}

fun greetings(): String =
    buildString(enhanced = true, who = "Devoxx", prefix = "He
```

named-params.java


```java
package _03_fun;

import kotlin.Metadata;
import kotlin.jvm.internal.Intrinsics;
import org.jetbrains.annotations.NotNull;

@Metadata(
   mv = {1, 1, 9},
   bv = {1, 0, 2},
   k = 2,
   d1 = {"\u0000\u0010\n\u0000\n\u0002\u0010\u000e\n\u0002\b\u
   d2 = {"buildString", "", "prefix", "who", "enhanced", "",
)
public final class NamedKt {
   @NotNull
   public static final String buildString(@NotNull String pre
      Intrinsics.checkParameterIsNotNull(prefix, "prefix");
      Intrinsics.checkParameterIsNotNull(who, "who");
      String msg = "" + prefix + ' ' + who;
      if (enhanced) {
         msg = msg + '!';
```

default-value.kt

```kotlin
fun buildString2(prefix: String = "Hello",
                 who: String,
                 enhanced: Boolean = true): String {
    var msg = "$prefix $who"
    if (enhanced) {
        msg += '!'
    }
    return msg
}

fun greetings2(): String =
    buildString2(who = "Devoxx")
```

default-value.java

```java
package _03_fun;

import kotlin.Metadata;
import kotlin.jvm.internal.Intrinsics;
import org.jetbrains.annotations.NotNull;

@Metadata(
   mv = {1, 1, 9},
   bv = {1, 0, 2},
   k = 2,
   d1 = {"\u0000\u0010\n\u0000\n\u0002\u0010\u000e\n\u0002\b\u
   d2 = {"buildString2", "", "prefix", "who", "enhanced", "",
)
public final class Default_valueKt {
   @NotNull
   public static final String buildString2(@NotNull String pre
      Intrinsics.checkParameterIsNotNull(prefix, "prefix");
      Intrinsics.checkParameterIsNotNull(who, "who");
      String msg = "" + prefix + ' ' + who;
      if (enhanced) {
         msg = msg + '!';
```

ByteCode de default-value


```
Compiled from "default-value.kt"
public final class _03_fun.Default_valueKt {
  public static final java.lang.String buildString2(java.lang
    Code:
       0: aload_0
       1: ldc           #9                  // String prefix
       3: invokestatic  #15                 // Method kotlin/
       6: aload_1
       7: ldc           #17                 // String who
       9: invokestatic  #15                 // Method kotlin/
      12: new           #19                 // class java/lang
      15: dup
      16: invokespecial #23                 // Method java/la
      19: ldc           #25                 // String
      21: invokevirtual #29                 // Method java/la
      24: aload_0
      25: invokevirtual #29                 // Method java/la
      28: bipush        32
      30: invokevirtual #32                 // Method java/la
      33: aload_1
      34: invokevirtual #29                 // Method java/la
```

C'est fun !

✨ Conseils

- Toujours typer le retour de vos fonctions (sauf si c'est évident et une surcharge comme le `toString`)
- Kotlin est plus expressif que Java => évitez de faire des fonctions trop longues
- Sautez une ligne après le `=`
- Utilisez le passage des arguments par nom quand ça lève des ambigüités

📝 Notes

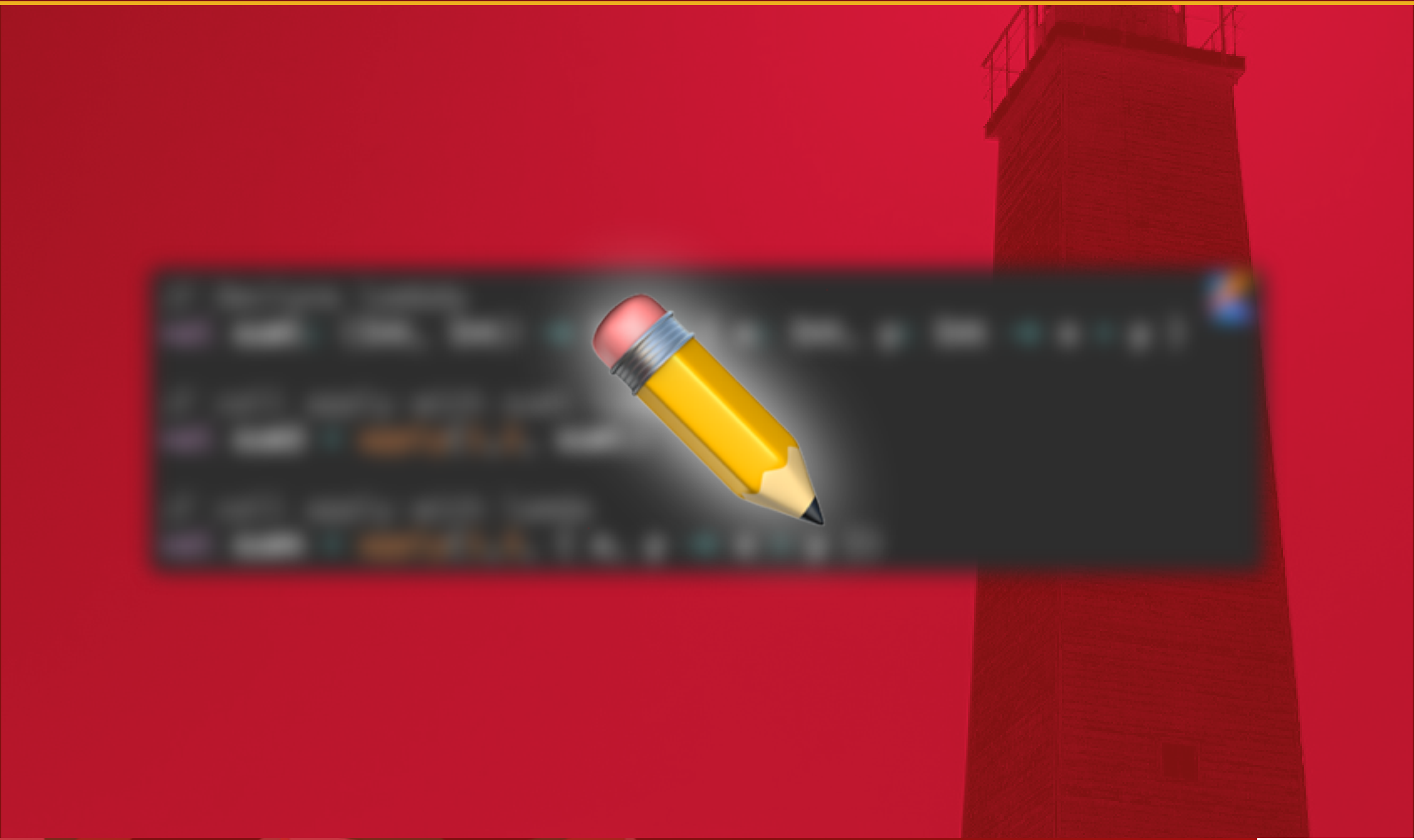
- Le passage des arguments par nom, ne marche pas sur les appels de code Java

Les lambdas

```kotlin
// Declare apply function with function as parameter
fun apply(x: Int, y: Int, operation: (Int, Int) → Int): Int =
    operation(x, y)

// Declare function
fun sumf(x: Int, y: Int) =
    x + y

// call apply with function reference
val sum5 = apply(2,3, ::sumf)

// store function reference
val sumLam = ::sumf

// call apply with the function reference
val sum6 = apply(1,5, sumLam)
```

function.java


```java
package _04_lamda;

import kotlin.Metadata;
import kotlin.jvm.functions.Function2;
import kotlin.jvm.internal.Intrinsics;
import kotlin.reflect.KFunction;
import org.jetbrains.annotations.NotNull;

@Metadata(
   mv = {1, 1, 9},
   bv = {1, 0, 2},
   k = 2,
   d1 = {"\u0000\u001c\n\u0000\n\u0002\u0010\b\n\u0002\b\u000
   d2 = {"sum5", "", "getSum5", "()I", "sum6", "getSum6", "su
)
public final class FunctionKt {
   private static final int sum5;
   @NotNull
   private static final KFunction sumLam;
   private static final int sum6;
```



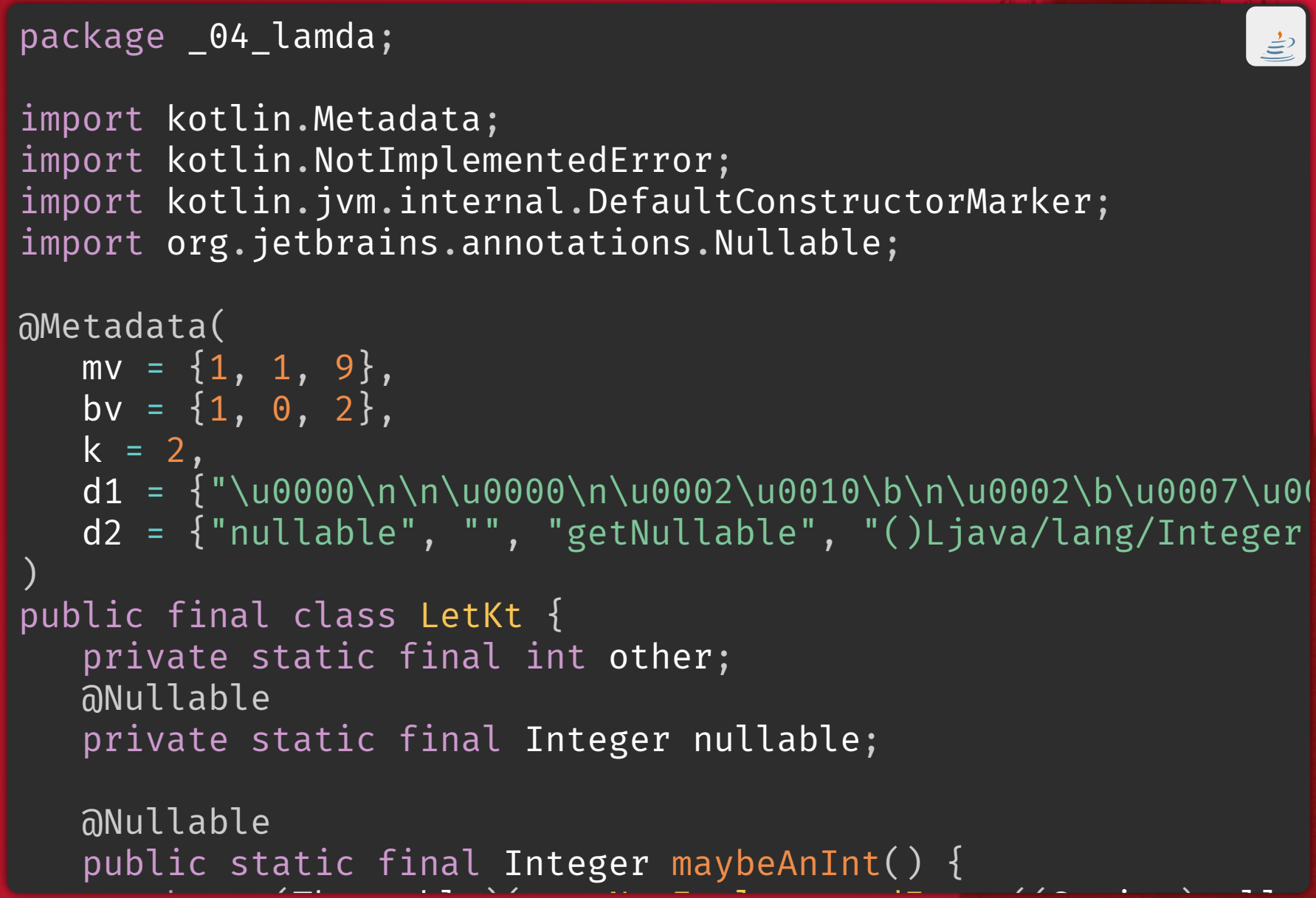
lambda.java


```java
package _04_lamda;

import kotlin.Metadata;
import kotlin.jvm.functions.Function2;
import org.jetbrains.annotations.NotNull;

@Metadata(
   mv = {1, 1, 9},
   bv = {1, 0, 2},
   k = 2,
   d1 = {"\u0000\u0012\n\u0000\n\u0002\u0010\b\n\u0002\b\u000
   d2 = {"sum3", "", "getSum3", "()I", "sum4", "getSum4", "su
)
public final class LambdaKt {
   @NotNull
   private static final Function2 suml;
   private static final int sum3;
   private static final int sum4;

   @NotNull
   public static final Function2 getSuml() {
```

let.kt

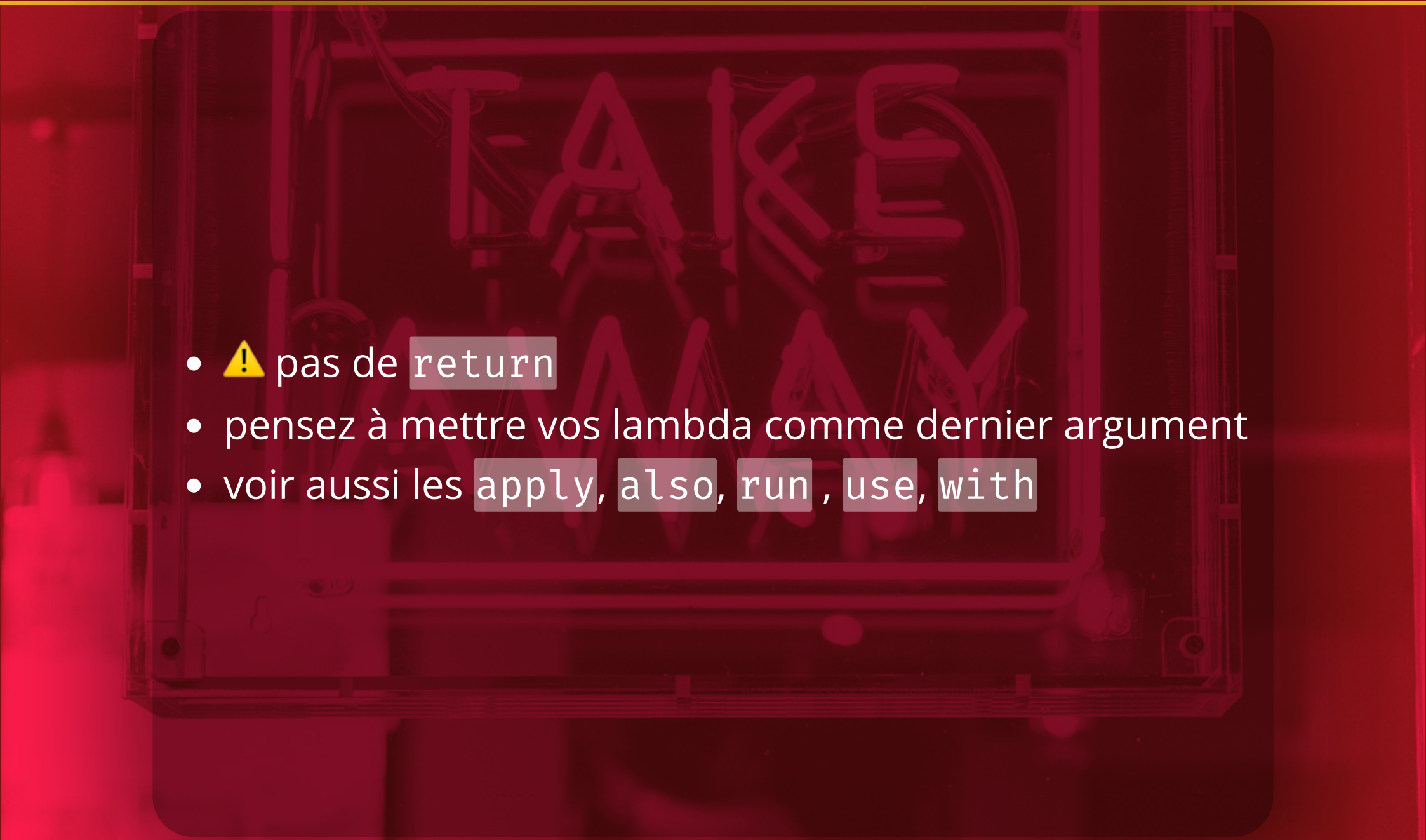

```kotlin
val other = sumf(1,2)
    .let { it + 1 }

val nullable = maybeAnInt()
    ?.let { it + 1 }
```

let.java



```java
package _04_lamda;

import kotlin.Metadata;
import kotlin.NotImplementedError;
import kotlin.jvm.internal.DefaultConstructorMarker;
import org.jetbrains.annotations.Nullable;

@Metadata(
    mv = {1, 1, 9},
    bv = {1, 0, 2},
    k = 2,
    d1 = {"\u0000\n\n\u0000\n\u0002\u0010\b\n\u0002\b\u0007\u00(
    d2 = {"nullable", "", "getNullable", "()Ljava/lang/Integer
)
public final class LetKt {
    private static final int other;
    @Nullable
    private static final Integer nullable;

    @Nullable
    public static final Integer maybeAnInt() {
```

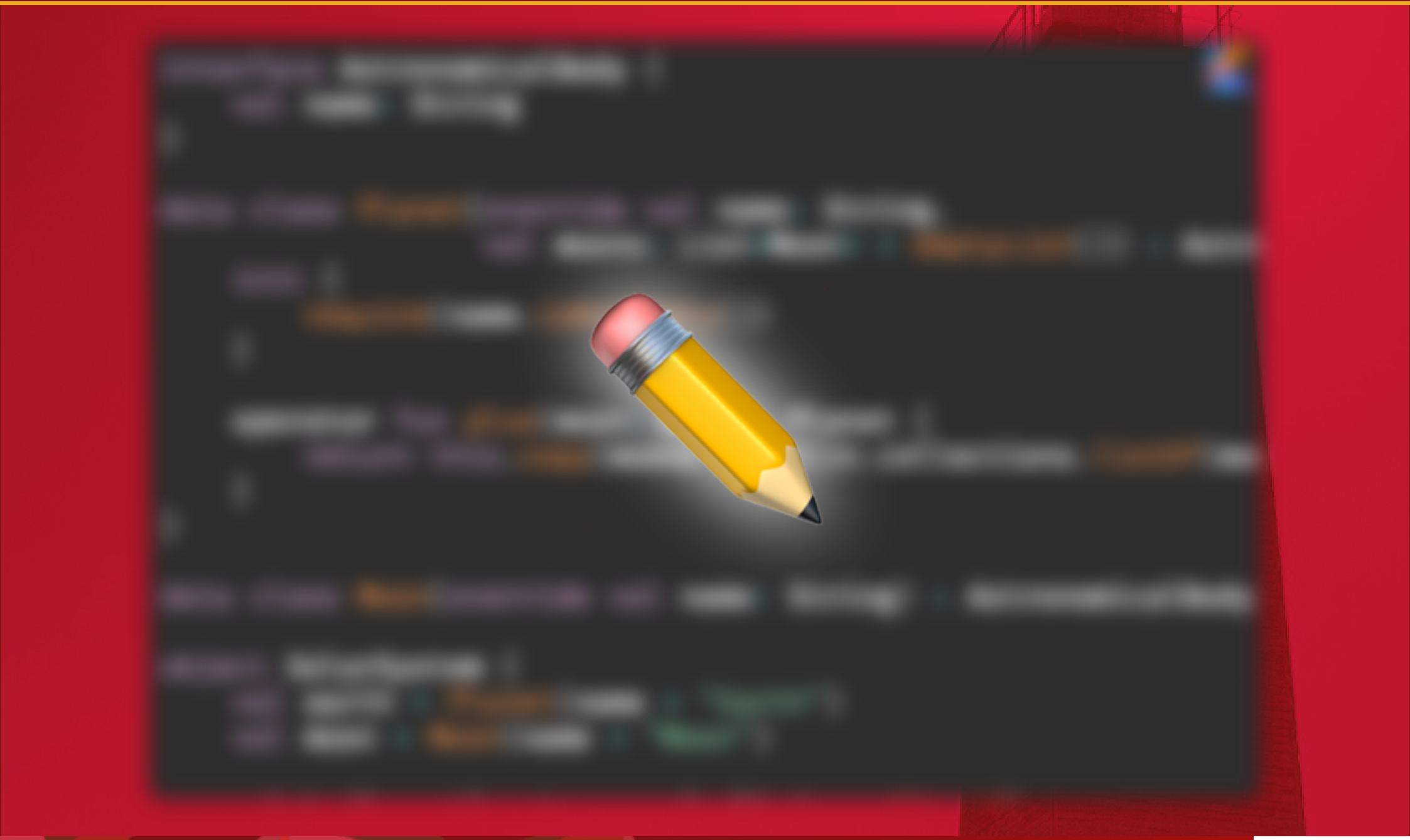
Les lambdas



- ⚠️ pas de `return`
- pensez à mettre vos lambda comme dernier argument
- voir aussi les `apply`, `also`, `run` , `use`, `with`

Les classes

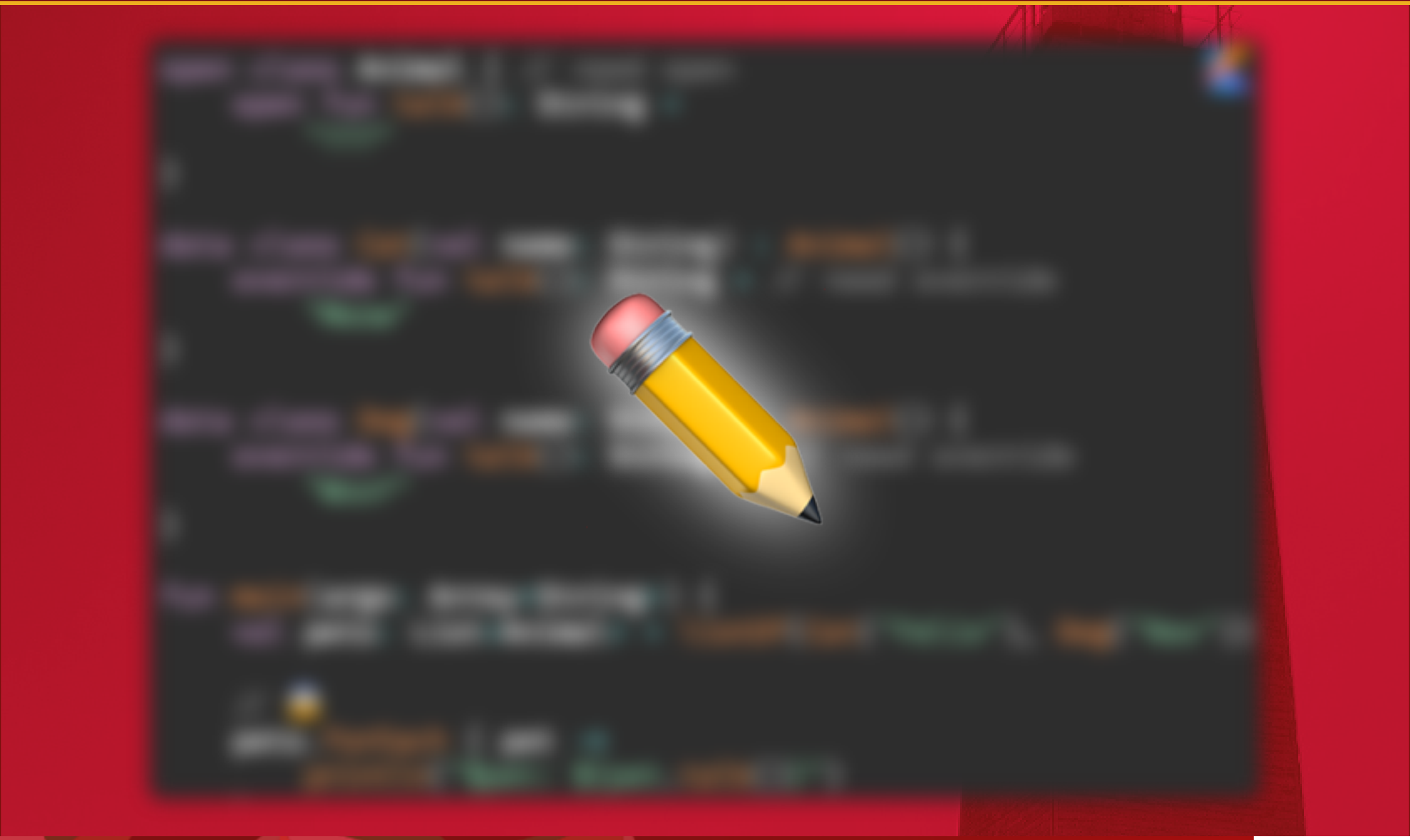
astronomy.kt



astronomy.java

```java
package _05_class_1;

import java.util.List;
import kotlin.Metadata;
import kotlin.collections.CollectionsKt;
import kotlin.jvm.internal.DefaultConstructorMarker;
import kotlin.jvm.internal.Intrinsics;
import org.jetbrains.annotations.NotNull;

@Metadata(
   mv = {1, 1, 9},
   bv = {1, 0, 2},
   k = 1,
   d1 = {"\u00000\n\u0002\u0018\u0002\n\u0002\u0018\u0002\n\u
   d2 = {"L_05_class_1/Planet;", "L_05_class_1/AstronomicalBo
)
public final class Planet implements AstronomicalBody {
   @NotNull
   private final String name;
   @NotNull
   private final List moons;
```

Héritage en Kotlin



Génerics

- ⚠️ Les contrôles de types génerics ne sont fait qu'au moment de la compilation
- Covariant (consome): `out`, en java `? extends T`
- Contravariant (produit): `in`, en java `? super T`

Borne supérieur

```
fun <T : Comparable<T>> sort(list: List<T>): List<T>
```

Les détails:
https://kotlinlang.org/docs/reference/generics.html

Projection *


```
interface Function<in T, out U>
```

```
Function<*, String> // correspond à Function<in Nothing, String>
```

```
Function<Int, *> // correspond à Function<Int, out Any?>
```

```
Function<*, *> // correspond à Function<in Nothing, out Any?>
```

Sealed


```
sealed class JsonValue

data class JsonObject(val attributes: Map<String, JsonValue>)
data class JsonArray(val values: List<JsonValue>) : JsonValue(
data class JsonString(val value: String) : JsonValue()
data class JsonNumber(val value: Number) : JsonValue()
data class JsonBoolean(val value: Boolean) : JsonValue()
object JsonNull : JsonValue()
```

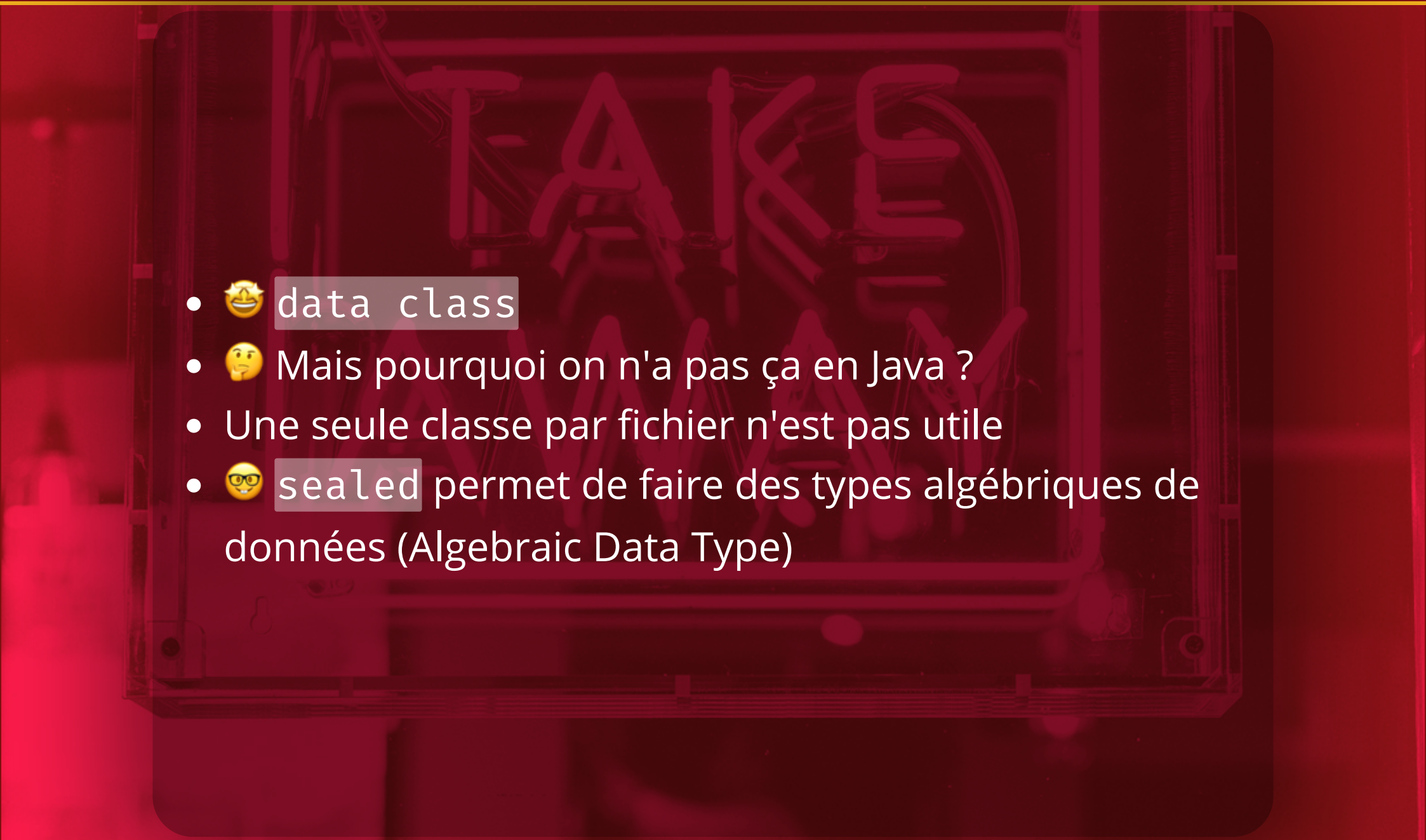
Alias en Kotlin

```kotlin
interface Entity

typealias Id = String
typealias Version = Int
typealias EntityKey = Pair<Id, Version>

// fun getAllEntities(): Map<Pair<String, Int>, List<Entity>>
fun getAllEntities(): Map<EntityKey, List<Entity>> = emptyMap(
```

ByteCode d'alias


```
Compiled from "typealias.kt"
public final class _06_class_2.TypealiasKt {
  public static final java.util.Map<kotlin.Pair<java.lang.Str
    Code:
       0: invokestatic  #12                 // Method kotlin/
       3: areturn
}
```

Classe, le bilan



- 🤩 `data class`
- 🤔 Mais pourquoi on n'a pas ça en Java ?
- Une seule classe par fichier n'est pas utile
- 🤓 `sealed` permet de faire des types algébriques de données (Algebraic Data Type)

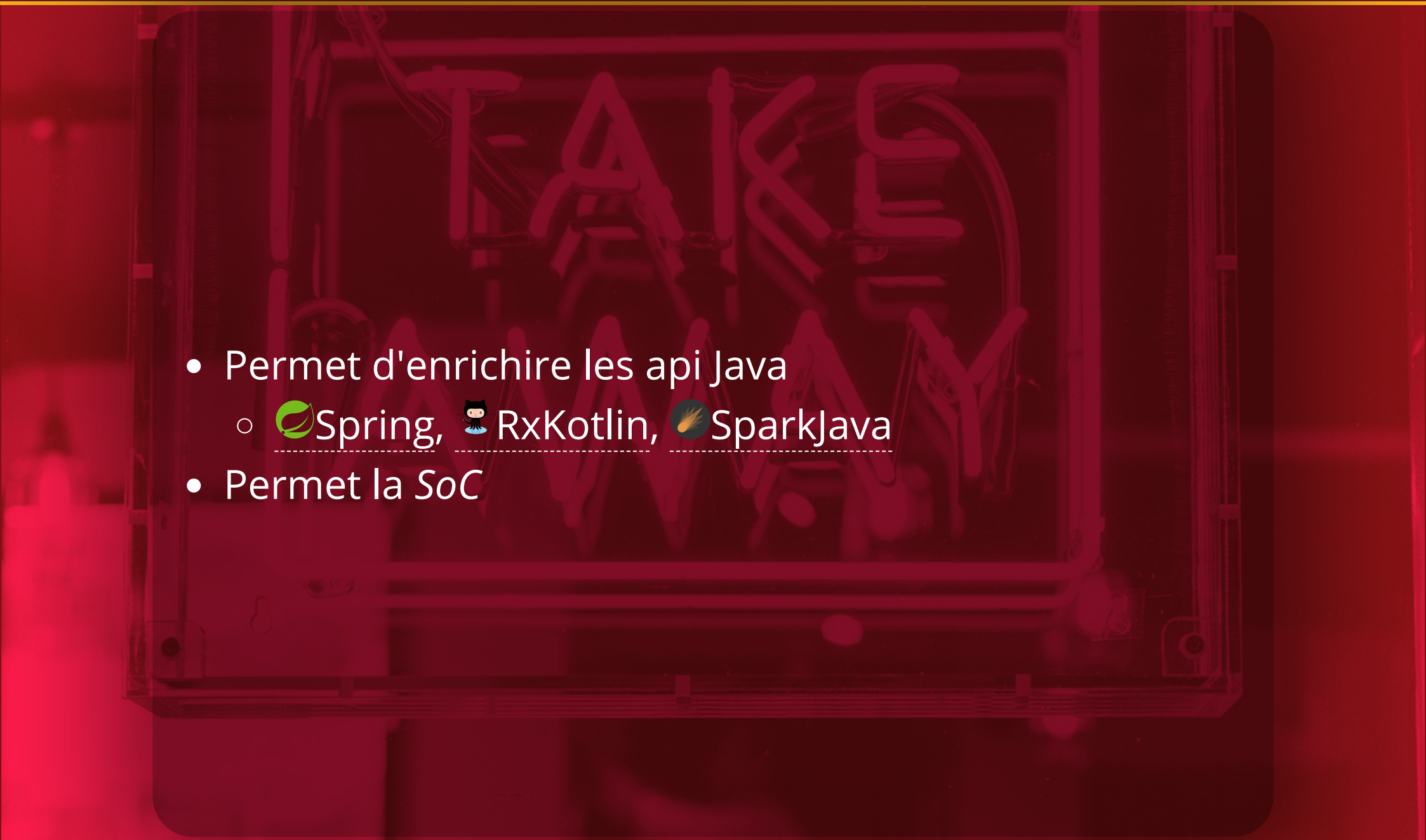
Extensions de fonctions

extension.kt

extension.java

```java
package _08_extension;

import astronomy.AstronomicalBody;
import astronomy.SolarSystem;
import java.util.Iterator;
import kotlin.Metadata;
import kotlin.jvm.internal.Intrinsics;
import org.jetbrains.annotations.NotNull;

@Metadata(
   mv = {1, 1, 9},
   bv = {1, 0, 2},
   k = 2,
   d1 = {"\u0000 \n\u0000\n\u0002\u0010\b\n\u0002\u0018\u0002
   d2 = {"size", "", "Lastronomy/AstronomicalBody;", "getSize
)
public final class ExtensionKt {
   public static final int getSize(@NotNull AstronomicalBody
      Intrinsics.checkParameterIsNotNull($receiver, "$receive
      return $receiver.getName().length();
   }
```

Extension

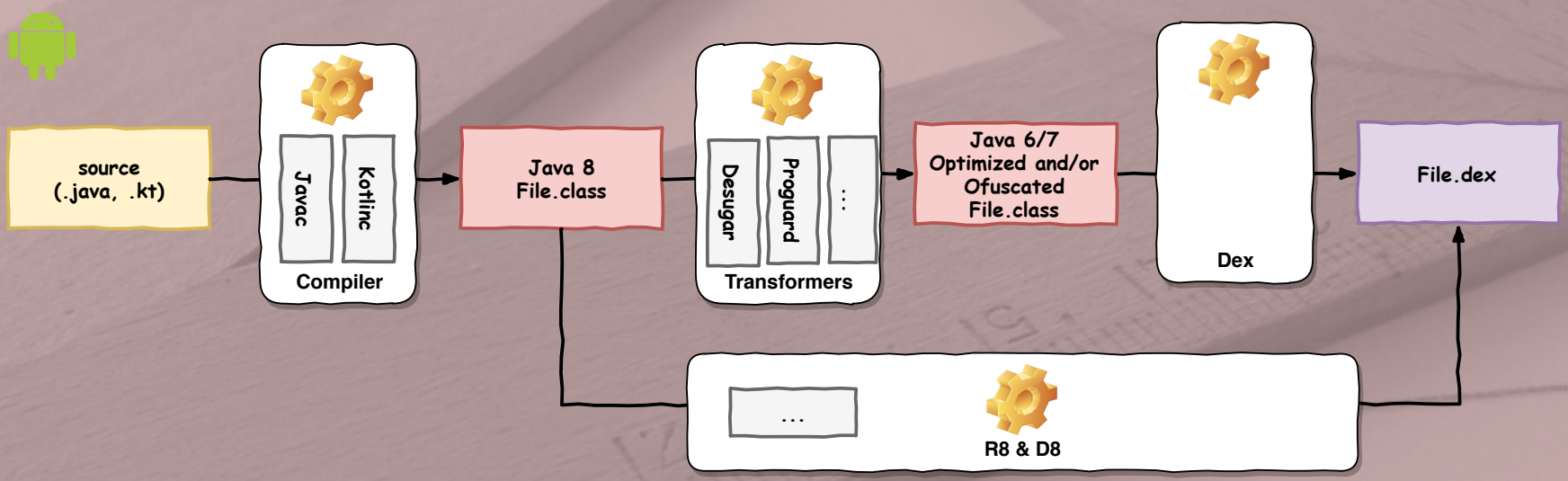


- Permet d'enrichire les api Java
  - Spring, RxKotlin, SparkJava
- Permet la *SoC*

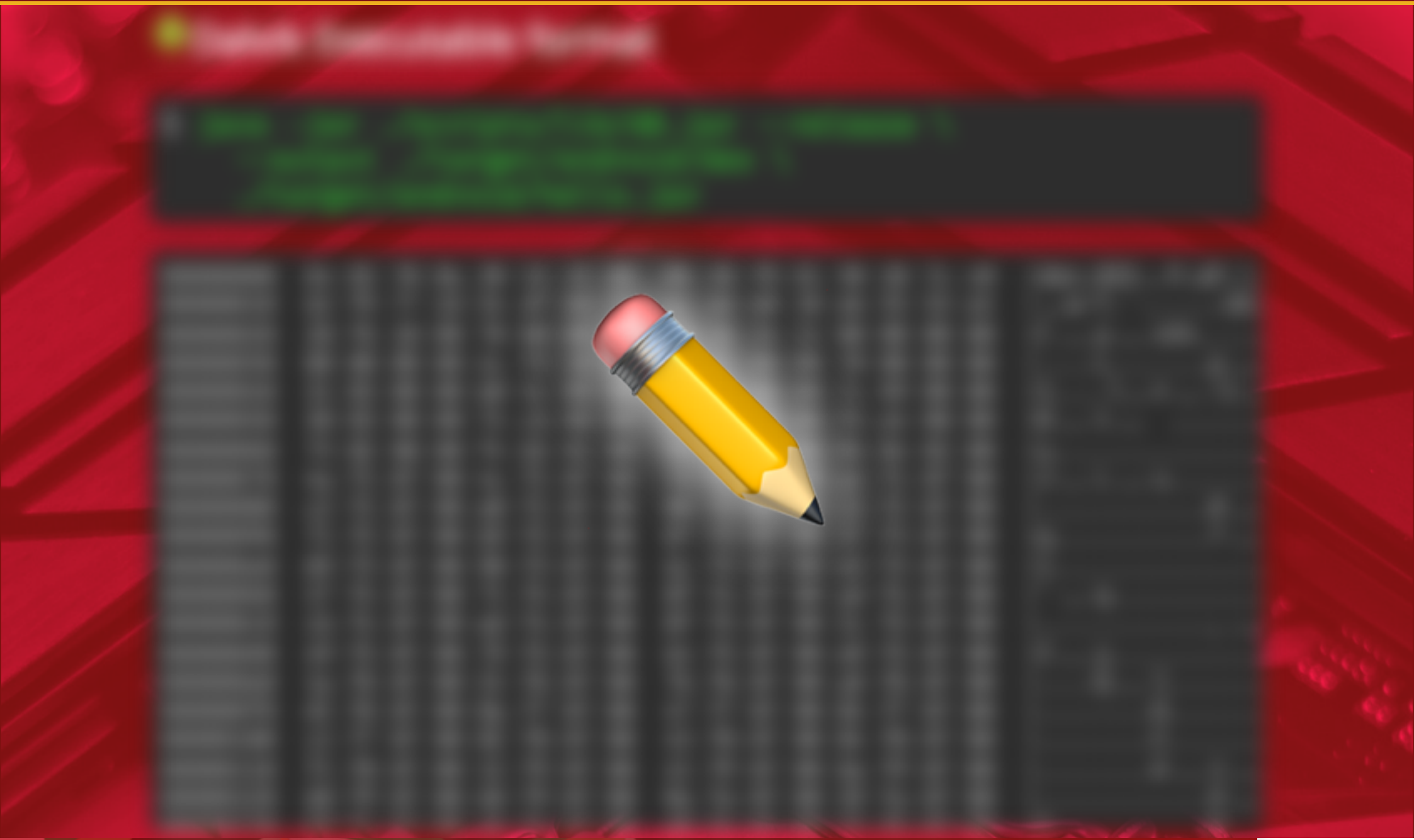
Pause

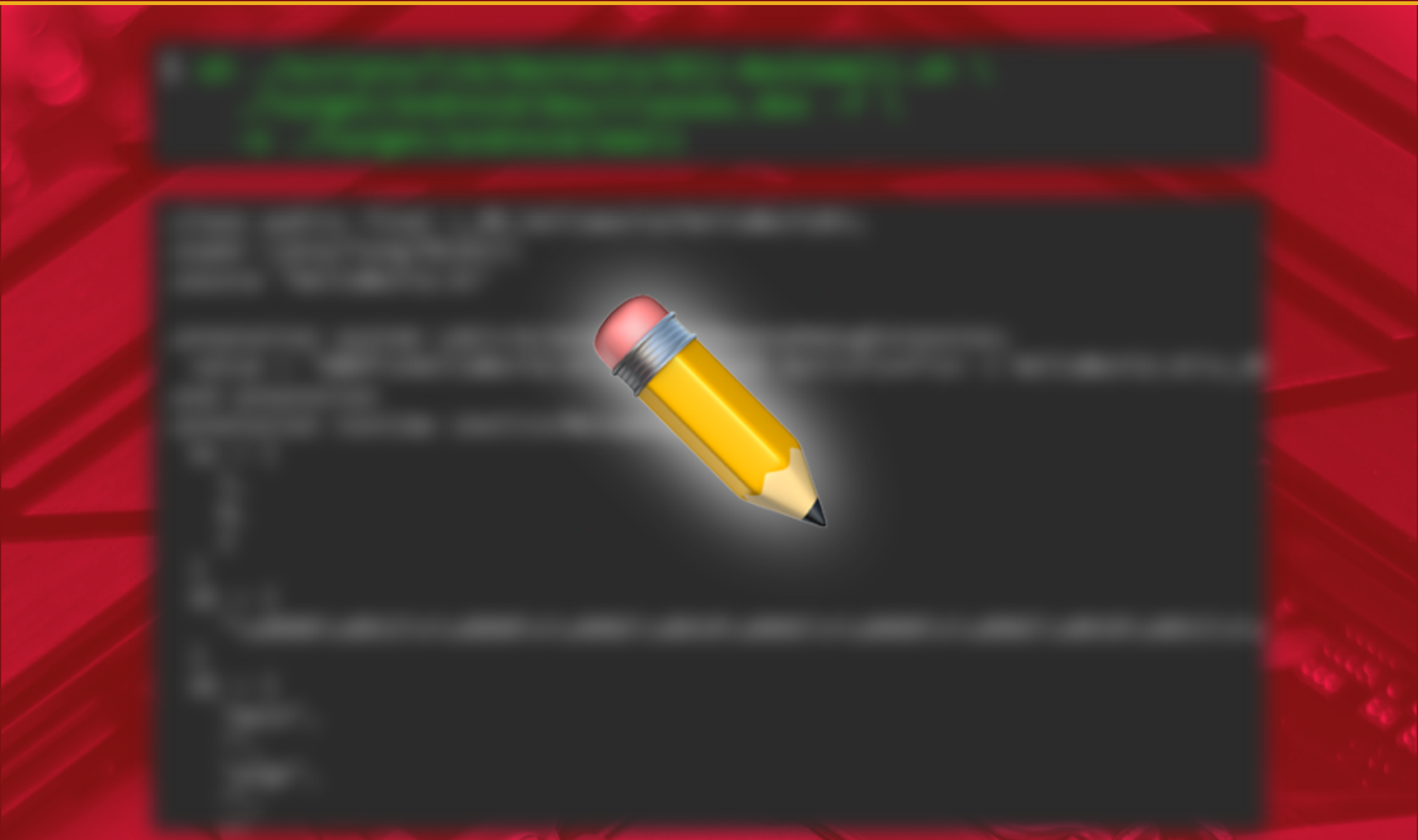
ByteCode Android

Compilation pour Android



Dalvik EXecutable format





Autres structures

if-then-else.kt


```kotlin
fun handleAstronomicalBody(body: AstronomicalBody) {
    val message =
        if (body is Planet &&
            body.name == "Earth"
        ) "Welcome Earth"
        else "Welcome martian"

    println(message)
}
```

```
fun main(args: Array<String>) {
    for (body in SolarSystem.bodies) { // 🤢
        print(body)
    }
}
```

```java
package _09_structures;

import astronomy.AstronomicalBody;
import astronomy.SolarSystem;
import java.util.Iterator;
import kotlin.Metadata;
import kotlin.jvm.internal.Intrinsics;
import org.jetbrains.annotations.NotNull;

@Metadata(
   mv = {1, 1, 9},
   bv = {1, 0, 2},
   k = 2,
   d1 = {"\u0000\u0012\n\u0000\n\u0002\u0010\u0002\n\u0000\n\u
   d2 = {"main", "", "args", "", "", "([Ljava/lang/String;)V"
)
public final class ForKt {
   public static final void main(@NotNull String[] args) {
      Intrinsics.checkParameterIsNotNull(args, "args");
      Iterator var2 = SolarSystem.INSTANCE.getBodies().iterat
```

ByteCode du for


```
Compiled from "for.kt"
public final class _09_structures.ForKt {
  public static final void main(java.lang.String[]);
    Code:
       0: aload_0
       1: ldc           #9                  // String args
       3: invokestatic  #15                 // Method kotlin/
       6: getstatic     #21                 // Field astronomy
       9: invokevirtual #25                 // Method astronor
      12: invokeinterface #31,  1           // InterfaceMetho
      17: astore_2
      18: aload_2
      19: invokeinterface #37,  1           // InterfaceMetho
      24: ifeq          47
      27: aload_2
      28: invokeinterface #41,  1           // InterfaceMetho
      33: checkcast     #43                 // class astronomy
      36: astore_1
      37: getstatic     #49                 // Field java/lang
      40: aload_1
      41: invokevirtual #55                 // Method java/io/
```

while-do.kt


```
while (x > 0) {
    x--
}

do {
    val y = retrieveData()
} while (y ≠ null) // y is visible here!
```

when.kt


```
for (body in SolarSystem.bodies) { // 🤢

    val message = when (body) {
        is Planet → "Planet ${body.name}"
        is Star   → "Star ${body.name}"
        else      → null
    }

    if (message ≠ null) {
        println(message)
    }
}
```

for-factorial.kt


```kotlin
// Note: assert(n ⩾ 0)
fun forFactorial(n: Int): Int { // 🤢
    var acc = 1
    for (i in 1..n) {
        acc *= i
    }
    return acc
}
```

ByteCode factoriel avec for


```
Compiled from "for-factorial.kt"
public final class _09_structures.recusion.For_factorialKt {
  public static final int forFactorial(int);
    Code:
       0: iconst_1
       1: istore_1
       2: iconst_1
       3: istore_2
       4: iload_0
       5: istore_3
       6: iload_2
       7: iload_3
       8: if_icmpgt     26
      11: iload_1
      12: iload_2
      13: imul
      14: istore_1
      15: iload_2
      16: iload_3
      17: if_icmpeq     26
      20: iinc          2, 1
```

rec-factorial.kt


```kotlin
// Note: assert(n ⩾ 0)
fun recFactorial(n: Int): Int =
    if (n < 1) 1 else n * recFactorial(n - 1)
```

ByteCode factoriel avec recursivité


```
Compiled from "rec-factorial.kt"
public final class _09_structures.recusion.Rec_factorialKt {
  public static final int recFactorial(int);
    Code:
       0: iload_0
       1: iconst_1
       2: if_icmpge     9
       5: iconst_1
       6: goto          17
       9: iload_0
      10: iload_0
      11: iconst_1
      12: isub
      13: invokestatic  #8                  // Method recFacto
      16: imul
      17: ireturn
}
```

tailrec-factorial.kt


```kotlin
// Note: assert(n ⩾ 0)
fun tailRecFactorial(n: Int): Int {

    tailrec fun aux(n: Int, acc: Int): Int =
        if (n < 1) 1 else aux(n - 1, acc * n)

    return aux(n, 1)
}
```

ByteCode factoriel avec recursivité terminal 1/2


```
Compiled from "tailrec-factorial.kt"
public final class _09_structures.recusion.Tailrec_factorialK
  public static final int tailRecFactorial(int);
    Code:
       0: getstatic     #12                 // Field _09_struc
       3: astore_1
       4: aload_1
       5: iload_0
       6: iconst_1
       7: invokevirtual #16                 // Method _09_str
      10: ireturn
}
```

ByteCode factoriel avec recursivité terminal 2/2


```
Compiled from "tailrec-factorial.kt"
final class _09_structures.recusion.Tailrec_factorialKt$tailRe
  public static final _09_structures.recusion.Tailrec_factoria

  public java.lang.Object invoke(java.lang.Object, java.lang.(
    Code:
       0: aload_0
       1: aload_1
       2: checkcast     #11                 // class java/lang
       5: invokevirtual #15                 // Method java/la
       8: aload_2
       9: checkcast     #11                 // class java/lang
      12: invokevirtual #15                 // Method java/la
      15: invokevirtual #18                 // Method invoke:(
      18: invokestatic  #24                 // Method java/la
      21: areturn

  public final int invoke(int, int);
    Code:
       0: iload_1
       1: iconst_1
```

Performances sur 10 !

> Ne croyez pas les benchmarks, faites les vous-même !

 https://github.com/MonkeyPatchIo/kotlin-perf

Benchmark	Mode	Cnt	Score	Error	Units
factorialJava	thrpt	200	274141213.561	± 28963758.069	ops/s
factorialKotlinFor	thrpt	200	267717955.205	± 8457315.205	ops/s
factorialKotlinRec	thrpt	200	56270660.700	± 2453418.383	ops/s
factorialKotlinTailRec	thrpt	200	341898899.761	± 11456349.191	ops/s

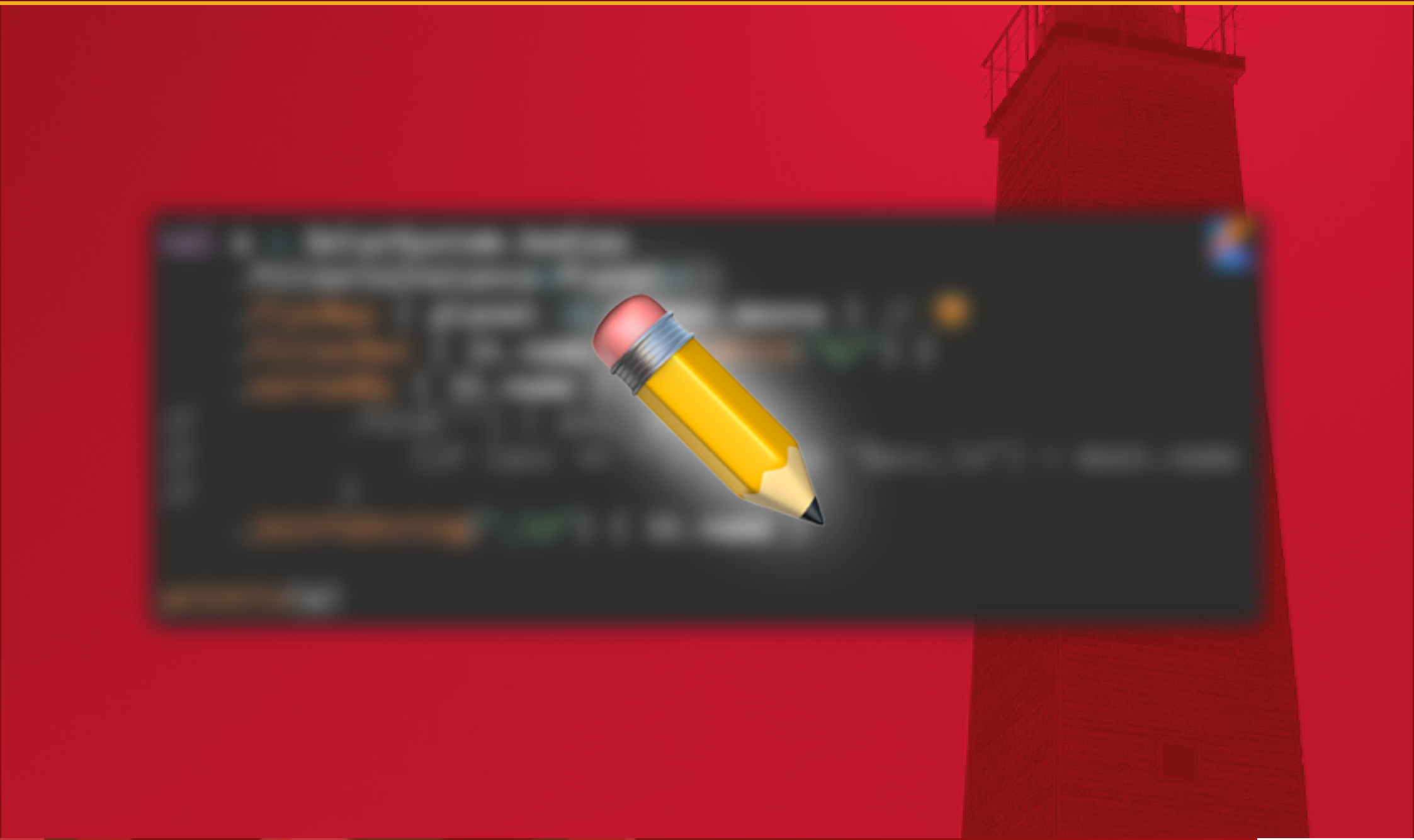
Bilan structures

- Il y a aussi des `break` et `continue`, label pour les boucles
- `when` peut être utiliser avec
  - des constantes,
  - plusieurs valeurs séparées par `,`
  - une expression
  - avec `is` et un type (avec un 'smart cast')

✨ **Tips**

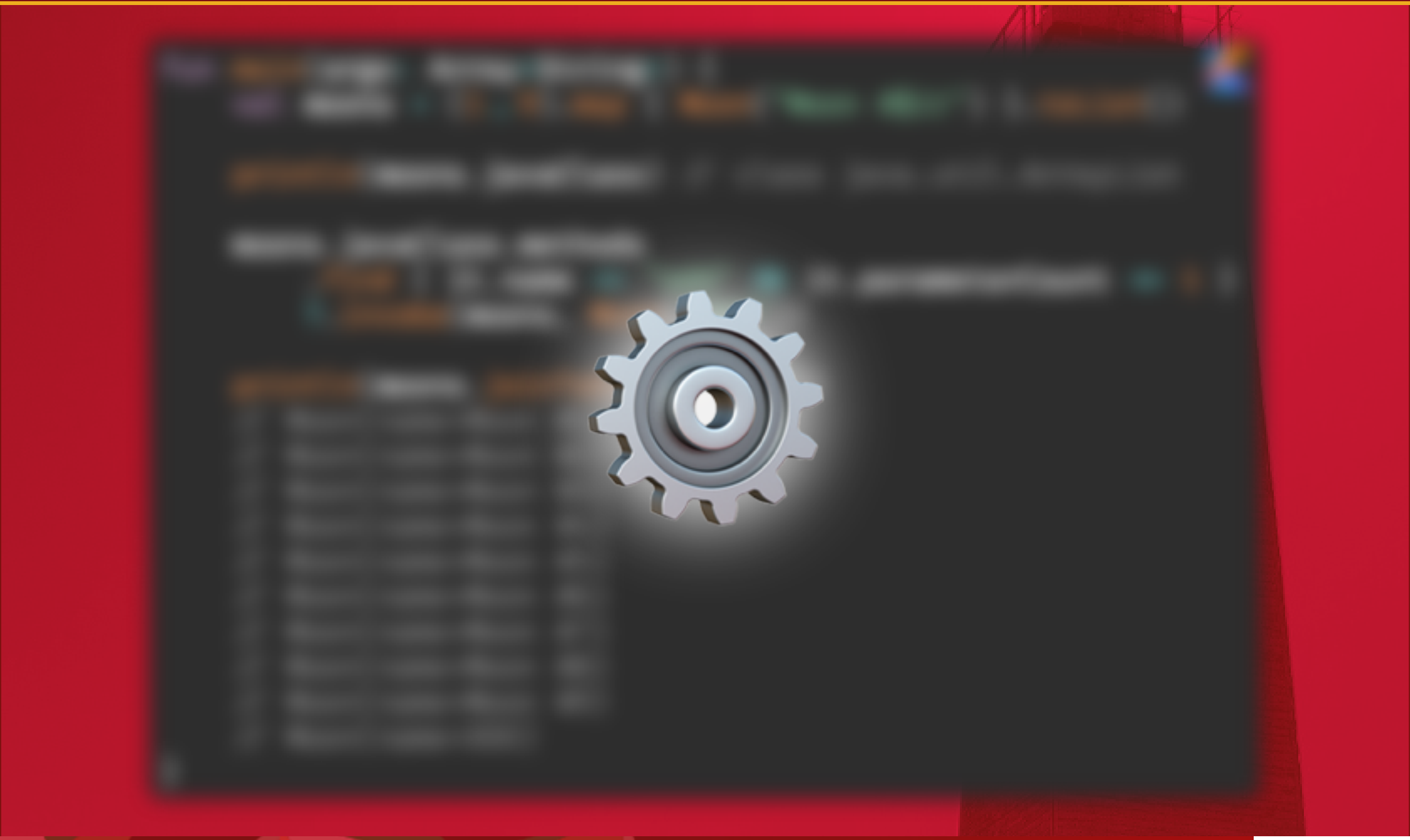
- privilégier les `when` si vous avez plus de 2 cas
- si vous faites des fonctions recursives, faites les `tailrec`

Les collections

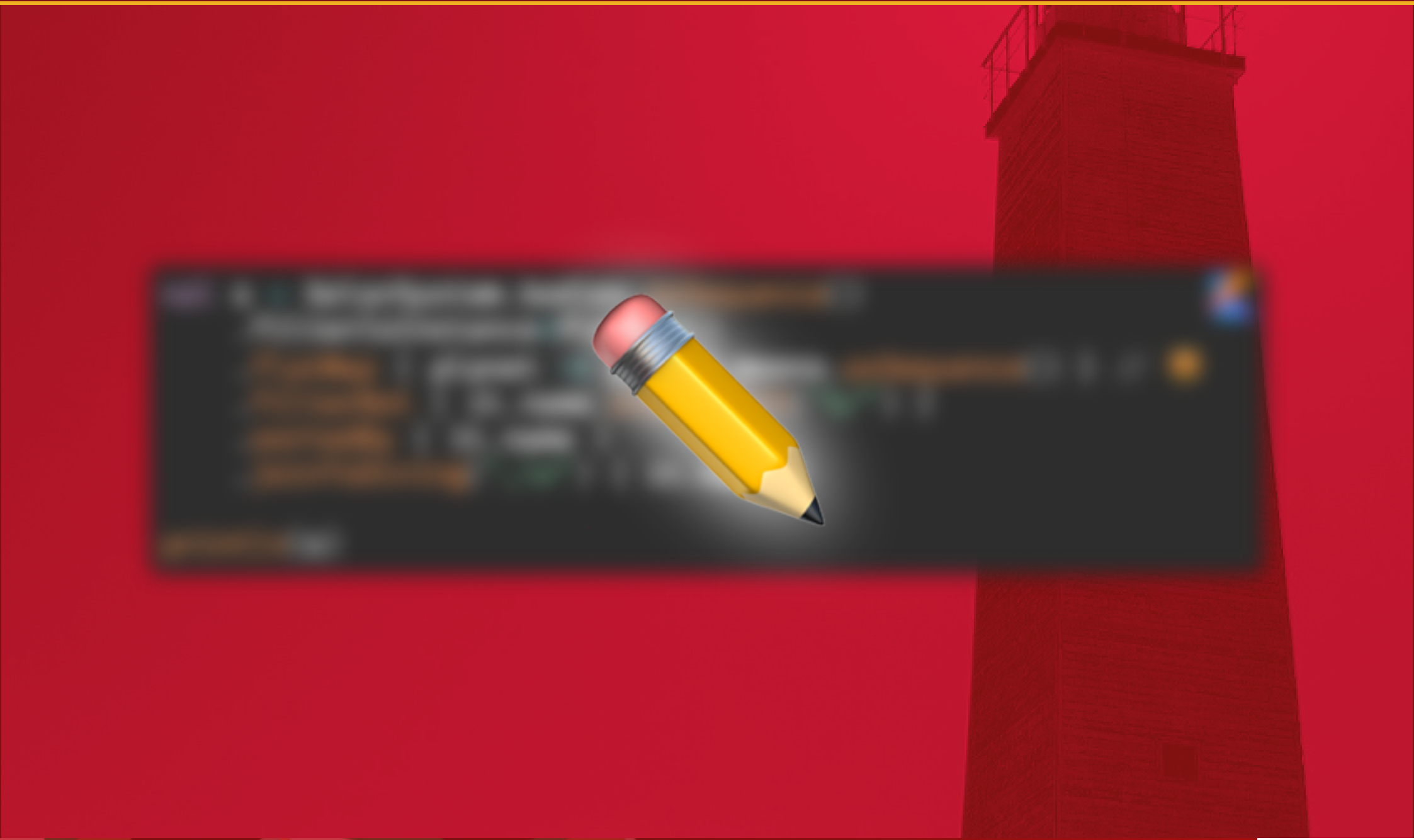


immutable-mutable.kt

break-immutable.kt



sequence.kt



Performance des sequences 1/2

> Ne croyez pas les benchmarks, faites les vous-même !

Benchmark	Mode	Cnt	Score	Error	Units
collectionApiClassic	thrpt	200	44535.029	± 3550.944	ops/s
collectionApiSequence	thrpt	200	23652.238	± 1967.535	ops/s

sequence2.kt


```kotlin
val s = SolarSystem.bodies.asSequence()
    .filterIsInstance<Planet>()
    .flatMap { planet → planet.moons.asSequence() } // 😻
    .filterNot { it.name.startsWith("S/") }
    .map { it.name }
    .first()

println(s)
```

Performance des sequences 2/2

> Ne croyez pas les benchmarks, faites les vous-même !

Benchmark	Mode	Cnt	Score	Error	Units
collectionApiClassicFirst	thrpt	200	241752.062	± 5022.663	ops/s
collectionApiSequenceFirst	thrpt	200	3615451.391	± 454502.198	ops/s

ranges.kt


```kotlin
for (i in 1..3) print(i) // prints 123

for (i in 3 downTo 1) print(i) // prints 321

for (i in 1..5 step 2) print(i) // prints 135
```

ranges.java

```
package _11_collections_2;

import kotlin.Metadata;
import kotlin.jvm.internal.Intrinsics;
import kotlin.ranges.IntProgression;
import kotlin.ranges.IntRange;
import org.jetbrains.annotations.NotNull;

@Metadata(
   mv = {1, 1, 9},
   bv = {1, 0, 2},
   k = 2,
   d1 = {"\u0000\u0012\n\u0000\n\u0002\u0010\u0002\n\u0000\n\u
   d2 = {"main", "", "args", "", "", "([Ljava/lang/String;)V"
)
public final class RangesKt {
   public static final void main(@NotNull String[] args) {
      Intrinsics.checkParameterIsNotNull(args, "args");
      int i = 1;

      byte var2;
```

tuples.kt


```kotlin
fun main(args: Array<String>) {
    val aPair = Pair("Earth", "Moon")
    val (planet, moon) = aPair

    val aTriple = Triple("Voyager 1", 1977, listOf("Jupiter",
    val (probeNane, launchYear, flyOver) = aTriple
}
```

tuples.java

```java
package _11_collections_2;

import java.util.List;
import kotlin.Metadata;
import kotlin.Pair;
import kotlin.Triple;
import kotlin.collections.CollectionsKt;
import kotlin.jvm.internal.Intrinsics;
import org.jetbrains.annotations.NotNull;

@Metadata(
   mv = {1, 1, 9},
   bv = {1, 0, 2},
   k = 2,
   d1 = {"\u0000\u0012\n\u0000\n\u0002\u0010\u0002\n\u0000\n\u
   d2 = {"main", "", "args", "", "", "([Ljava/lang/String;)V"
)
public final class TuplesKt {
   public static final void main(@NotNull String[] args) {
      Intrinsics.checkParameterIsNotNull(args, "args");
      Pair pair = new Pair(1, "x");
```

Bilan collection

- Super on a de l'immutabilité, des `map`, `flatMap`, `fold`, `aggregate`...
- Mais ça reste des collections Java
- Avant d'utiliser les sequences, faites des mesures

Les delegates

delegate.kt

lazy.kt

observable.kt

lateinit.kt

Delegate

Un peu plus sur les fonctions

inline.kt

```kotlin
import java.time.Instant

class Logger(private val name: String) {
    private enum class Level { TRACE, DEBUG, INFO, WARN, ERRO
    private val level = Level.INFO

    fun info(message: () → String) {
        log(Level.INFO, message)
    }

    private inline fun log(lvl: Level, message: () → String)
        if (level ⩾ lvl) {
            println("[${level.name}] $name - ${message()}")
        }
    }
}

fun main(args: Array<String>) {
    val logger = Logger("Main")

    logger.info { "Time: ${Instant.now()}" }
}
```

Logger.java

```java
package _13_advanced_function;

import kotlin.Metadata;
import kotlin.jvm.functions.Function0;
import kotlin.jvm.internal.Intrinsics;
import org.jetbrains.annotations.NotNull;

@Metadata(
   mv = {1, 1, 9},
   bv = {1, 0, 2},
   k = 1,
   d1 = {"\u0000&\n\u0002\u0018\u0002\n\u0002\u0010\u0000\n\u(
   d2 = {"L_13_advanced_function/Logger;", "", "name", "", "(
)
public final class Logger {
   private final Logger.Level level;
   private final String name;

   public final void info(@NotNull Function0 message) {
      Intrinsics.checkParameterIsNotNull(message, "message");
      Logger.Level lvl$iv = Logger.Level.INFO;
```

reified.kt

```kotlin
class Pojo {
    var name: String? = null
    override fun toString() = "Pojo $name"
}

object JavaBeanBuilder {
    fun <T> createBean(clazz: Class<T>): T =
        clazz.newInstance()
    inline fun <reified T> createBean(): T =
        createBean(T::class.java)
}

fun main(args: Array<String>) {
    val p1 = Pojo()
    p1.name = "Plop1"
    println(p1)

    val p2 = JavaBeanBuilder.createBean<Pojo>()
    p2.name = "Plop2"
    println(p2)
}
```

Plus sur les fonctions

Cas d'utilisation du `reified`

- pour créer des extensions kotlin des fonctions Java qui utilisent des `Class<T>`

Cas d'utilisation des `inline`, `noinline`

- quand on utilise `reified`
- quand on sait se qu'on fait, https://kotlinlang.org/docs/reference/inline-functions.html

Conclusion

- Faible surchage
- Support officiel par Google
- Using Project Kotlin for Android
- Kotlin Guide
- android-ktx
- Kotlin Android Extensions

Serveur

- Supporter officiellement depuis Spring 5, Spring Boot 2
- SparkJava, javalin
- Vert.x
- KTor
- ...

Web et Natif

Web

- Partager du code commun
-  Use Kotlin with npm, webpack and react

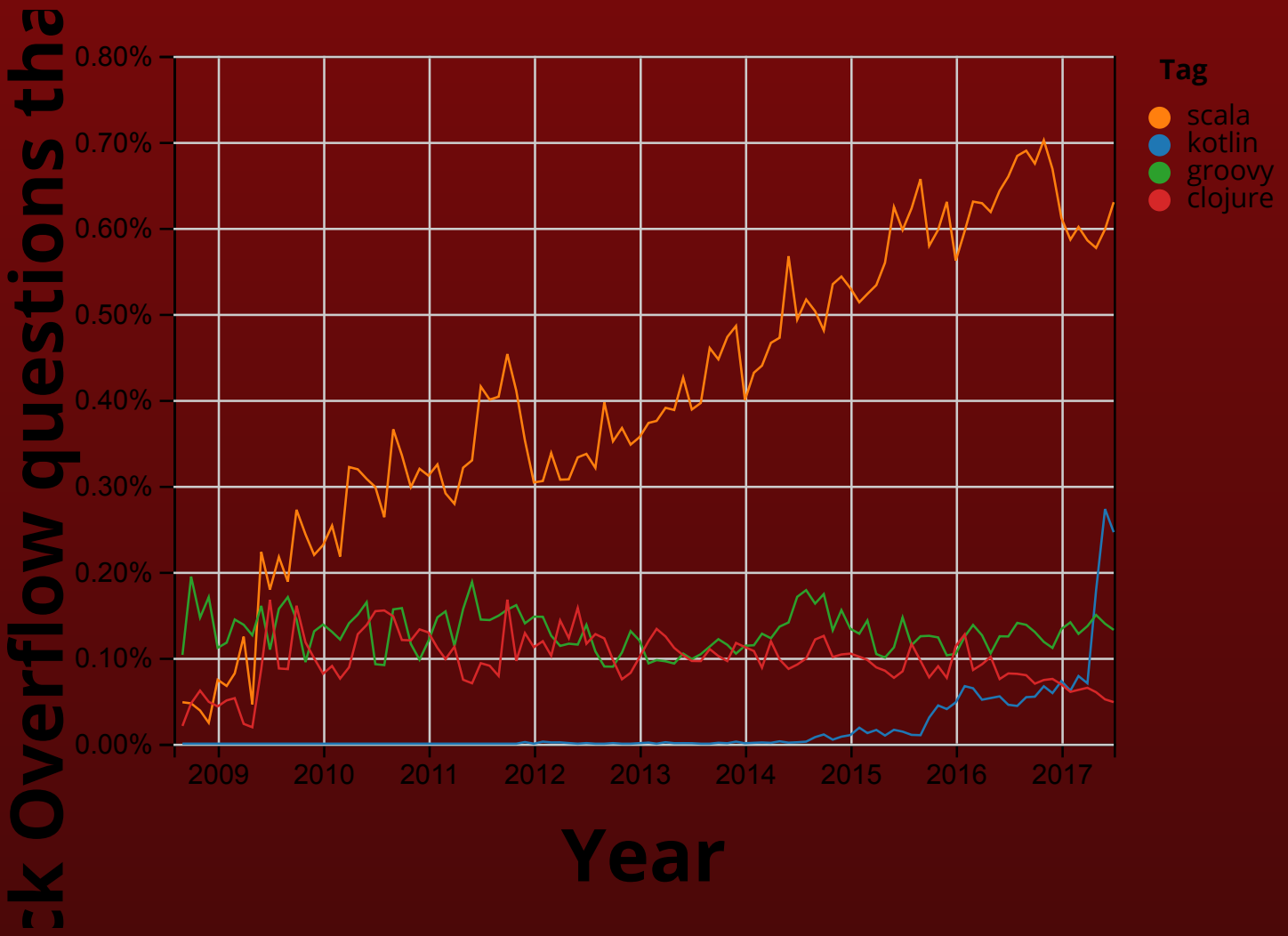
Natif

- Faire des applications sans JVM
- Partager du code avec iOS
- WebAssembly

Bilan

- 💎 JVM
- 😎 Le byte code c'est cool
- 🔮 Généralement, ça ne suffit pas pour prédire les performances
- ⚖️ Mesurez !

Tendance



Stackoverflow insights

Kotlin vs Java

- C'est déjà mature
- ✊ Code plus expressif, plus sûr, plus simple
- 🤝 Interopérable avec Java
- 👍 Outillage (editeur, gradle, maven)
- 👍 Ecosystème et communauté
- 🚀 Évolution rapide
- 🐣 Code multiplatform

> Kotlin réussit une belle alchimie entre pragmatisme, puissance, sûreté, accessibilité.

Liens

- Référence
- Blog
- Forum
- Slack
- Koans
- KEEP - Kotlin Evolution and Enhancement Process

Liens presentation

- Slides en HTML: http://bit.ly/KotlinDevoxxFR
- Slides en PDF: http://bit.ly/kotlinDevoxxFRpdf
- kotlin-perf
- Kotlin by example

Bibliothèques

- kotlinx.serialization
- kotlinx.coroutines
- KotlinTest
- Javalin
- RxKotlin
- ∧rrow
- Kotlin is Awesome

Questions ?

Pensez au votes et aux retours