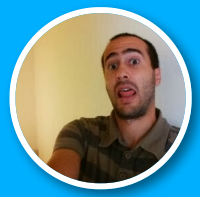

Deep Dive Kotlin :
du Hello World au
ByteCode



Emmanuel Vinas

Expert Android & Java

@emmanuelvinas <https://twitter.com/emmanuelvinas>

emmanuel@monkeypatch.io

<mailto:emmanuel@monkeypatch.io>

Igor Laborie

Expert Web & Java

@ilaborie <https://twitter.com/ilaborie>

igor@monkeypatch.io <mailto:igor@monkeypatch.io>

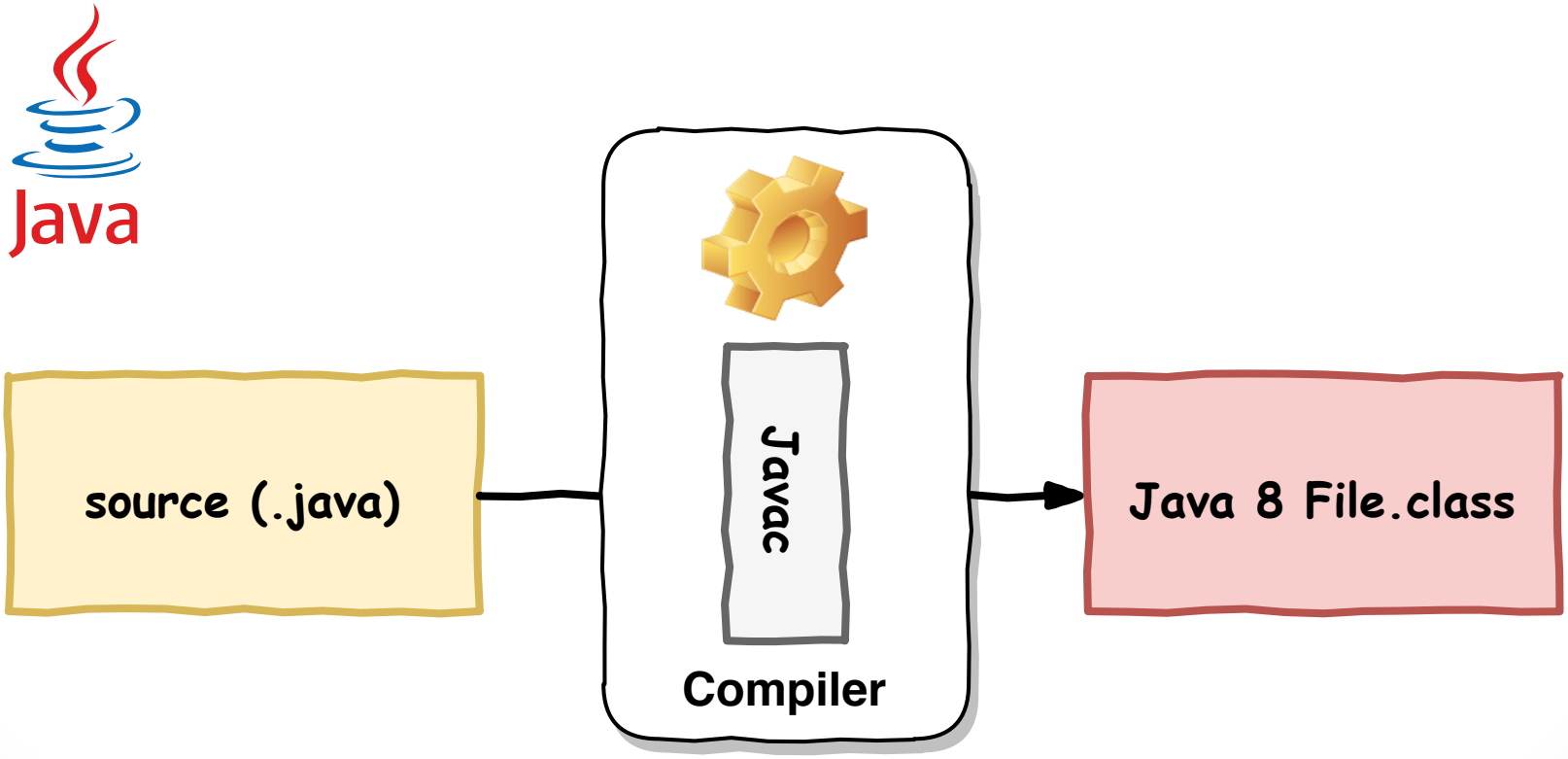
<Monkey Patch/>

Roadmap

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

ByteCode Java ?

javac



HelloWorld.java


```java
public class HelloWorld {

    public static void main(String[] args) {
        System.out.println("Hello RivieraDev");
    }
}
```

```
javac HelloWorld.java
```

Java ByteCode binary


```
hexdump -C HelloWorld.class
```

```
00000000  ca fe ba be 00 00 00 34  00 1d 0a 00 06 00 0f 09  |.......4........|
00000010  00 10 00 11 08 00 12 0a  00 13 00 14 07 00 15 07  |................|
00000020  00 16 01 00 06 3c 69 6e  69 74 3e 01 00 03 28 29  |.....<init>...()|
00000030  56 01 00 04 43 6f 64 65  01 00 0f 4c 69 6e 65 4e  |V...Code...LineN|
00000040  75 6d 62 65 72 54 61 62  6c 65 01 00 04 6d 61 69  |umberTable...mai|
00000050  6e 01 00 16 28 5b 4c 6a  61 76 61 2f 6c 61 6e 67  |n...([Ljava/lang|
00000060  2f 53 74 72 69 6e 67 3b  29 56 01 00 0a 53 6f 75  |/String;)V...Sou|
00000070  72 63 65 46 69 6c 65 01  00 0f 48 65 6c 6c 6f 57  |rceFile...HelloW|
00000080  6f 72 6c 64 2e 6a 61 76  61 0c 00 07 00 08 07 00  |orld.java.......|
00000090  17 0c 00 18 00 19 01 00  0c 48 65 6c 6c 6f 20 44  |.........Hello D|
000000a0  65 76 6f 78 78 07 00 1a  0c 00 1b 00 1c 01 00 19  |evoxx...........|
000000b0  5f 30 30 5f 68 65 6c 6c  6f 77 6f 72 6c 64 2f 48  |_00_helloworld/H|
000000c0  65 6c 6c 6f 57 6f 72 6c  64 01 00 10 6a 61 76 61  |elloWorld...java|
000000d0  2f 6c 61 6e 67 2f 4f 62  6a 65 63 74 01 00 10 6a  |/lang/Object...j|
000000e0  61 76 61 2f 6c 61 6e 67  2f 53 79 73 74 65 6d 01  |ava/lang/System.|
000000f0  00 03 6f 75 74 01 00 15  4c 6a 61 76 61 2f 69 6f  |..out...Ljava/io|
00000100  2f 50 72 69 6e 74 53 74  72 65 61 6d 3b 01 00 13  |/PrintStream;...|
00000110  6a 61 76 61 2f 69 6f 2f  50 72 69 6e 74 53 74 72  |java/io/PrintStr|
00000120  65 61 6d 01 00 07 70 72  69 6e 74 6c 6e 01 00 15  |eam...println...|
00000130  28 4c 6a 61 76 61 2f 6c  61 6e 67 2f 53 74 72 69  |(Ljava/lang/Stri|
00000140  6e 67 3b 29 56 00 21 00  05 00 06 00 00 00 00 00  |ng;)V.!.........|
00000150  02 00 01 00 07 00 08 00  01 00 09 00 00 00 1d 00  |................|
00000160  01 00 01 00 00 00 05 2a  b7 00 01 b1 00 00 00 01  |.......*........|
00000170  00 0a 00 00 00 06 00 01  00 00 00 03 00 09 00 0b  |................|
00000180  00 0c 00 01 00 09 00 00  00 25 00 02 00 01 00 00  |.........%......|
```

Explorons le ByteCode


```
javap -c HelloWorld.class
```

```
Compiled from "HelloWorld.java"
public class _00_helloworld.HelloWorld {
  public _00_helloworld.HelloWorld();
    Code:
       0: aload_0
       1: invokespecial #1                  // Method java/lang/Object."<init>":()V
       4: return

  public static void main(java.lang.String[]);
    Code:
       0: getstatic     #2                  // Field java/lang/System.out:Ljava/io/PrintStream;
       3: ldc           #3                  // String Hello RivieraDev
       5: invokevirtual #4                  // Method java/io/PrintStream.println:(Ljava/lang/String;)V
       8: return
}
```

À propos du ByteCode

Environ 200 opérations possibles (maxi. 256 opscodes)

Préfixe pour le type d'opérations (`i` pour entier, `d` pour double, ...)

Manipulation de la pile, des variables locales (`iconst_0`, `istore`, `iload`, ...)

Contrôle du flux des instructions (`if_icmpgt`, `goto`, ...)

Arithmétique et conversion de type (`iadd`, `iinc`, `i2d`, ...)

Manipulation d'objets (`invokevirtual`, `invokedynamic`, ...)

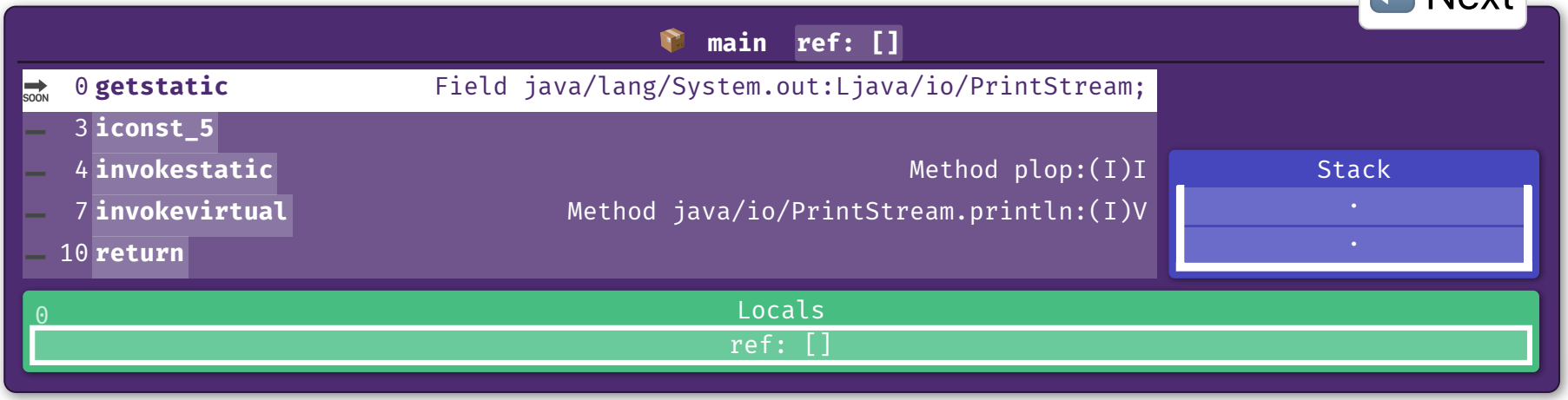
Autres (`athrow`, ...)

Jouons un peu

▶ Constant Pool

▼ Frames

Next



Liens

- Mastering Java Bytecode at the Core of the JVM <https://zeroturnaround.com/rebellabs/rebel-labs-report-mastering-java-bytecode-at-the-core-of-the-jvm/>
- Introduction to Java Bytecode <https://mahmoudanouti.wordpress.com/2018/03/20/introduction-to-java-bytecode/>
- The Java® Virtual Machine Specification <https://docs.oracle.com/javase/specs/jvms/se10/html/index.html>
- The Java Virtual Machine Instruction Set <https://docs.oracle.com/javase/specs/jvms/se10/html/jvms-6.html>
- Suivez le lapin blanc : Exploration au coeur de la JVM <https://www.youtube.com/watch?v=rB0EIXf05nU>
- Byte Buddy <http://bytebuddy.net/#/>
- asm <http://asm.ow2.org/>

> *"Soyez curieux, regardez comment ça marche avec* `javap -c` *!*

Introduction Kotlin

Historique


Dévoilé par JetBrains
v1.1 : coroutines, JS, ...
Support par Google
v1.2 : multiplatform
corountine-flow
v1.4 ?
???
2011 - 2015
2016
2017
2018
2019
...
v1.0
Support annoncé par Spring
v1.3
Coroutine stable
Kotlin Native beta
Kotlin Foundation

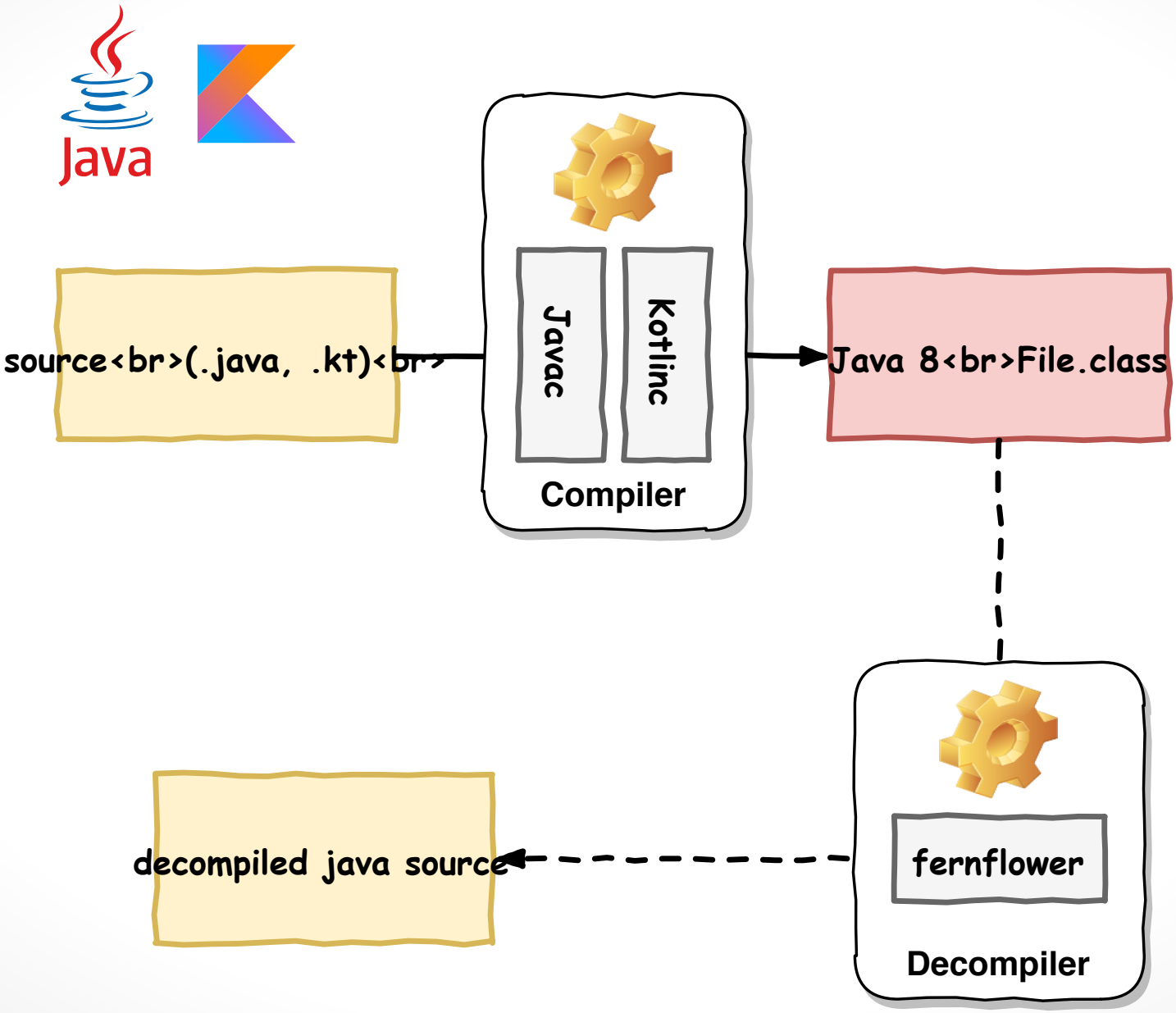
Cibles

HelloWorld.kt


```kotlin
fun main(args: Array<String>) {
    println("Hello RivieraDev")
}
```

```
kotlinc HelloWorld.kt
```

kotlinc


Java
source
(.java, .kt)
Javac
Kotlinc
Compiler
Java 8
File.class
fernflower
Decompiler
decompiled java source

hexdump


```
00000000  ca fe ba be 00 00 00 32  00 2b 01 00 0c 48 65 6c  |.......2.+ ... Hel|
00000010  6c 6f 57 6f 72 6c 64 4b  74 07 00 01 01 00 10 6a  |loWorldKt......j|
00000020  61 76 61 2f 6c 61 6e 67  2f 4f 62 6a 65 63 74 07  |ava/lang/Object.|
00000030  00 03 01 00 04 6d 61 69  6e 01 00 03 28 29 56 01  |.....main ... ()V.|
00000040  00 16 28 5b 4c 6a 61 76  61 2f 6c 61 6e 67 2f 53  |..([Ljava/lang/S|
00000050  74 72 69 6e 67 3b 29 56  0c 00 05 00 06 0a 00 02  |tring;)V........|
00000060  00 08 01 00 10 48 65 6c  6c 6f 20 52 69 76 69 65  |.....Hello Rivie|
00000070  72 61 44 65 76 08 00 0a  01 00 10 6a 61 76 61 2f  |raDev......java/|
00000080  6c 61 6e 67 2f 53 79 73  74 65 6d 07 00 0c 01 00  |lang/System.....|
00000090  03 6f 75 74 01 00 15 4c  6a 61 76 61 2f 69 6f 2f  |.out ... Ljava/io/|
000000a0  50 72 69 6e 74 53 74 72  65 61 6d 3b 0c 00 0e 00  |PrintStream;....|
000000b0  0f 09 00 0d 00 10 01 00  13 6a 61 76 61 2f 69 6f  |.........java/io|
000000c0  2f 50 72 69 6e 74 53 74  72 65 61 6d 07 00 12 01  |/PrintStream....|
000000d0  00 07 70 72 69 6e 74 6c  6e 01 00 15 28 4c 6a 61  |..println ... (Lja|
000000e0  76 61 2f 6c 61 6e 67 2f  4f 62 6a 65 63 74 3b 29  |va/lang/Object;)|
000000f0  56 0c 00 14 00 15 0a 00  13 00 16 01 00 11 4c 6b  |V.............Lk|
00000100  6f 74 6c 69 6e 2f 4d 65  74 61 64 61 74 61 3b 01  |otlin/Metadata;.|
00000110  00 02 6d 76 03 00 00 00  01 03 00 00 00 0f 01 00  |..mv............|
00000120  02 62 76 03 00 00 00 00  03 00 00 00 03 01 00 01  |.bv.............|
00000130  6b 03 00 00 00 02 01 00  02 64 31 01 00 13 c0 80  |k........d1.....|
00000140  06 0a c0 80 0a 02 10 02  1a 06 10 c0 80 1a 02 30  |...............0|
00000150  01 01 00 02 64 32 01 00  00 01 00 0d 48 65 6c 6c  |....d2......Hell|
00000160  6f 57 6f 72 6c 64 2e 6b  74 01 00 04 43 6f 64 65  |oWorld.kt ... Code|
00000170  01 00 0f 4c 69 6e 65 4e  75 6d 62 65 72 54 61 62  | ... LineNumberTab|
00000180  6c 65 01 00 0a 53 6f 75  72 63 65 46 69 6c 65 01  |le ... SourceFile.|
00000190  00 14 53 6f 75 72 63 65  44 65 62 75 67 45 78 74  |..SourceDebugExt|
000001a0  65 6e 73 69 6f 6e 01 00  19 52 75 6e 74 69 6d 65  |ension ... Runtime|
```

Java ByteCode


```
Compiled from "HelloWorld.kt"
public final class HelloWorldKt {
  public static final void main();
    Code:
       0: ldc           #11                 // String Hello RivieraDev
       2: astore_0
       3: iconst_0
       4: istore_1
       5: getstatic     #17                 // Field java/lang/System.out:Ljava/io/PrintStream;
       8: aload_0
       9: invokevirtual #23                 // Method java/io/PrintStream.println:(Ljava/lang/Object;)V
      12: return

  public static void main(java.lang.String[]);
    Code:
       0: invokestatic  #9                  // Method main:()V
       3: return
}
```

HelloWorld-java


```java
import kotlin.Metadata;

@Metadata(
        mv = {1, 1, 15},
        bv = {1, 0, 3},
        k = 2,
        d1 = {"\u0000\b\n\u0000\n\u0002\u0010\u0002\n\u0000\u001a\u0006\u0010\u0000\u001a\u00020\u0001¨\u00
        d2 = {"main", "", "kotlinDee"}
)
public final class HelloWorldKt {
   public static final void main() {
      String var0 = "Hello RivieraDev";
      boolean var1 = false;
      System.out.println(var0);
   }

   // $FF: synthetic method
   public static void main(String[] var0) {
      main();
   }
}
```

Bilan HelloWorld.kt

👮 Kotlin ajoute des contrôles

du coup, on a besoin de JARs en plus

	jar	version	taille
kotlin-stdlib	kotlin-stdlib	1.3.31	1.3M
kotlin-stdlib-jdk7	kotlin-stdlib-jdk7	1.3.31	3.1k
kotlin-stdlib-jdk8	kotlin-stdlib-jdk8	1.3.31	13k
kotlin-reflect	kotlin-reflect	1.3.31	2.8M
lombok	lombok	1.18.6	1.7M
spring-core	spring-core	5.1.6	1.3M

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

```kotlin
fun greeting(who: Someone) {
    println("Hello $who!")
    println("Hello ${who.firstName} ${who.lastName}!")
    println("""
        multi
        line "$who"
        string""".trimIndent())
}
```

string-template.java


```java
import kotlin.Metadata;
import kotlin.jvm.internal.Intrinsics;
import kotlin.text.StringsKt;
import org.jetbrains.annotations.NotNull;

@Metadata(
         mv = {1, 1, 15},
         bv = {1, 0, 3},
         k = 2,
         d1 = {"\u0000\u000e\n\u0000\n\u0002\u0010\u0002\n\u0000\n\u0002\u0018\u0002\n\u0000\u001a\u000e\u00
         d2 = {"greeting", "", "who", "LSomeone;", "kotlinDee"}
)
public final class GreetingKt {
   public static final void greeting(@NotNull Someone who) {
      Intrinsics.checkParameterIsNotNull(who, "who");
      String var1 = "Hello " + who + '!';
      boolean var2 = false;
      System.out.println(var1);
      var1 = "Hello " + who.getFirstName() + ' ' + who.getLastName() + '!';
      var2 = false;
      System.out.println(var1);
      var1 = StringsKt.trimIndent("\n        multi\n        line \"" + who + "\"\n        string");
      var2 = false;
      System.out.println(var1);
   }
}
```

ByteCode de string-template


```
Compiled from "greeting.kt"
public final class GreetingKt {
  public static final void greeting(Someone);
    Code:
       0: aload_0
       1: ldc           #9                  // String who
       3: invokestatic  #15                 // Method kotlin/jvm/internal/Intrinsics.checkParameterIsNotNul
       6: new           #17                 // class java/lang/StringBuilder
       9: dup
      10: invokespecial #21                 // Method java/lang/StringBuilder."<init>":()V
      13: ldc           #23                 // String Hello
      15: invokevirtual #27                 // Method java/lang/StringBuilder.append:(Ljava/lang/String;)Lj
      18: aload_0
      19: invokevirtual #30                 // Method java/lang/StringBuilder.append:(Ljava/lang/Object;)Lj
      22: bipush        33
      24: invokevirtual #33                 // Method java/lang/StringBuilder.append:(C)Ljava/lang/StringBu
      27: invokevirtual #37                 // Method java/lang/StringBuilder.toString:()Ljava/lang/String;
      30: astore_1
      31: iconst_0
      32: istore_2
      33: getstatic     #43                 // Field java/lang/System.out:Ljava/io/PrintStream;
      36: aload_1
      37: invokevirtual #49                 // Method java/io/PrintStream.println:(Ljava/lang/Object;)V
      40: new           #17                 // class java/lang/StringBuilder
      43: dup
      44: invokespecial #21                 // Method java/lang/StringBuilder."<init>":()V
      47: ldc           #23                 // String Hello
```

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

numeric.java


```java
import kotlin.Metadata;
import org.jetbrains.annotations.Nullable;

@Metadata(
        mv = {1, 1, 15},
        bv = {1, 0, 3},
        k = 2,
        d1 = {"\u0000\u001a\n\u0000\n\u0002\u0010\u0006\n\u0002\b\u0006\n\u0002\u0010\t\n\u0002\b\u0003\n\u
        d2 = {"aDouble", "", "getADouble", "()Ljava/lang/Double;", "setADouble", "(Ljava/lang/Double;)V", "
)
public final class NumericKt {
   private static final int anInt = 42;
   private static final long aLong = 42L;
   @Nullable
   private static Double aDouble;

   public static final int getAnInt() {
      return anInt;
   }

   public static final long getALong() {
      return aLong;
   }

   @Nullable
   public static final Double getADouble() {
      return aDouble;
```

ByteCode de numeric


```
Compiled from "numeric.kt"
public final class NumericKt {
  public static final int getAnInt();
    Code:
       0: getstatic     #11                 // Field anInt:I
       3: ireturn

  public static final long getALong();
    Code:
       0: getstatic     #19                 // Field aLong:J
       3: lreturn

  public static final java.lang.Double getADouble();
    Code:
       0: getstatic     #26                 // Field aDouble:Ljava/lang/Double;
       3: areturn

  public static final void setADouble(java.lang.Double);
    Code:
       0: aload_0
       1: putstatic     #26                 // Field aDouble:Ljava/lang/Double;
       4: return

  static {};
    Code:
       0: bipush        42
       2: putstatic     #11                 // Field anInt:I
```

C'est simple

Plus de `;` *

😍 String templating

😘 Plus de types primitifs (avant la compilation)

🧐 Inférence de type

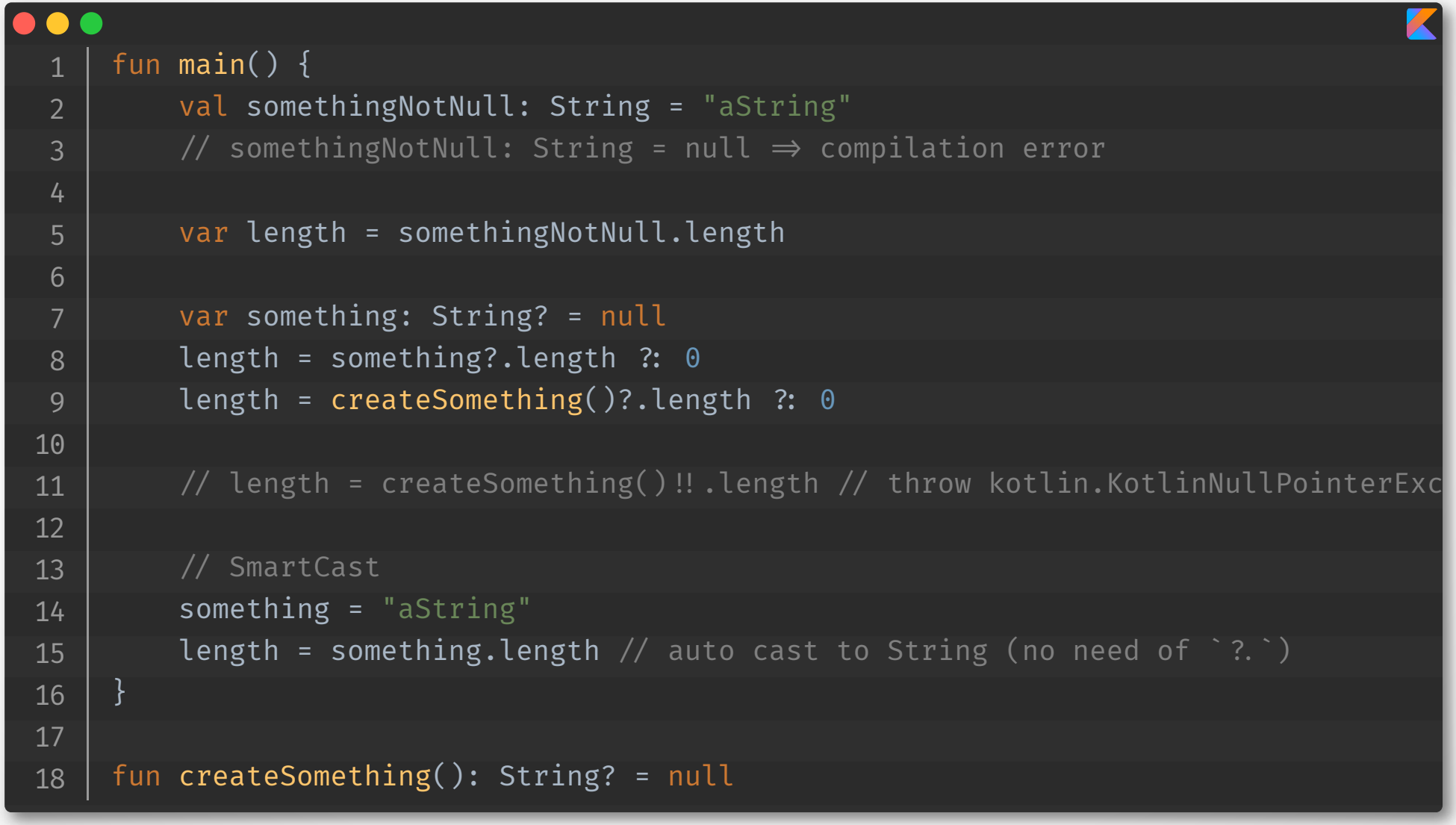
🤝 On peut mélanger du code Java et Kotlin

null-safety

> *"I call it my billion-dollar mistake. It was the invention of the `null` reference in 1965. [...]. 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 <https://www.infoq.com/presentations/Null-References-The-Billion-Dollar-Mistake-Tony-Hoare>

null-safety.kt



```kotlin
fun main() {
    val somethingNotNull: String = "aString"
    // somethingNotNull: String = null ⇒ compilation error

    var length = somethingNotNull.length

    var something: String? = null
    length = something?.length ?: 0
    length = createSomething()?.length ?: 0

    // length = createSomething()!!.length // throw kotlin.KotlinNullPointerExc

    // SmartCast
    something = "aString"
    length = something.length // auto cast to String (no need of `?.`)
}

fun createSomething(): String? = null
```

null-safety.java


```java
import kotlin.Metadata;
import org.jetbrains.annotations.Nullable;

@Metadata(
        mv = {1, 1, 15},
        bv = {1, 0, 3},
        k = 2,
        d1 = {"\u0000\u000e\n\u0000\n\u0002\u0010\u000e\n\u0000\n\u0002\u0010\u0002\n\u0000\u001a\b\u0010\u
        d2 = {"createSomething", "", "main", "", "kotlinDee"}
)
public final class Null_safetyKt {
   public static final void main() {
      String somethingNotNull = "aString";
      int length = somethingNotNull.length();
      String something = (String)null;
      int length = false;
      String var10000 = createSomething();
      length = var10000 ≠ null ? var10000.length() : 0;
      something = "aString";
      length = something.length();
   }

   // $FF: synthetic method
   public static void main(String[] var0) {
      main();
   }

```

Bilan null-safety

🎸 Elvis operator: `?:`

🙌 plus de `NullPointerException`

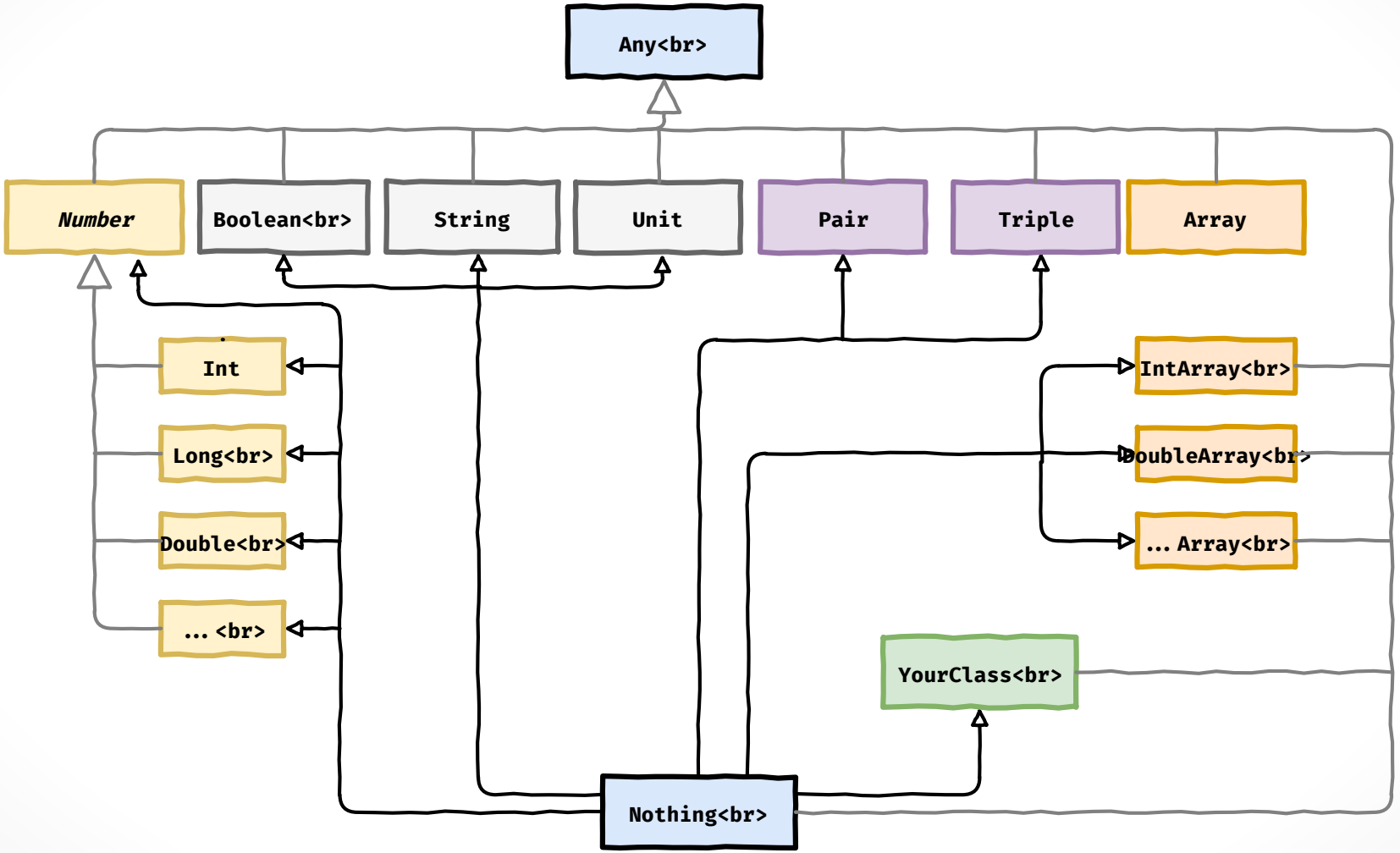
⚠️ quand on appelle du Java

➽ Le compilateur Kotlin peut interpréter des annotions de nullabilité (JSR-305, Android, ...) <https://kotlinlang.org/docs/reference/java-interop.html#nullability-annotations>

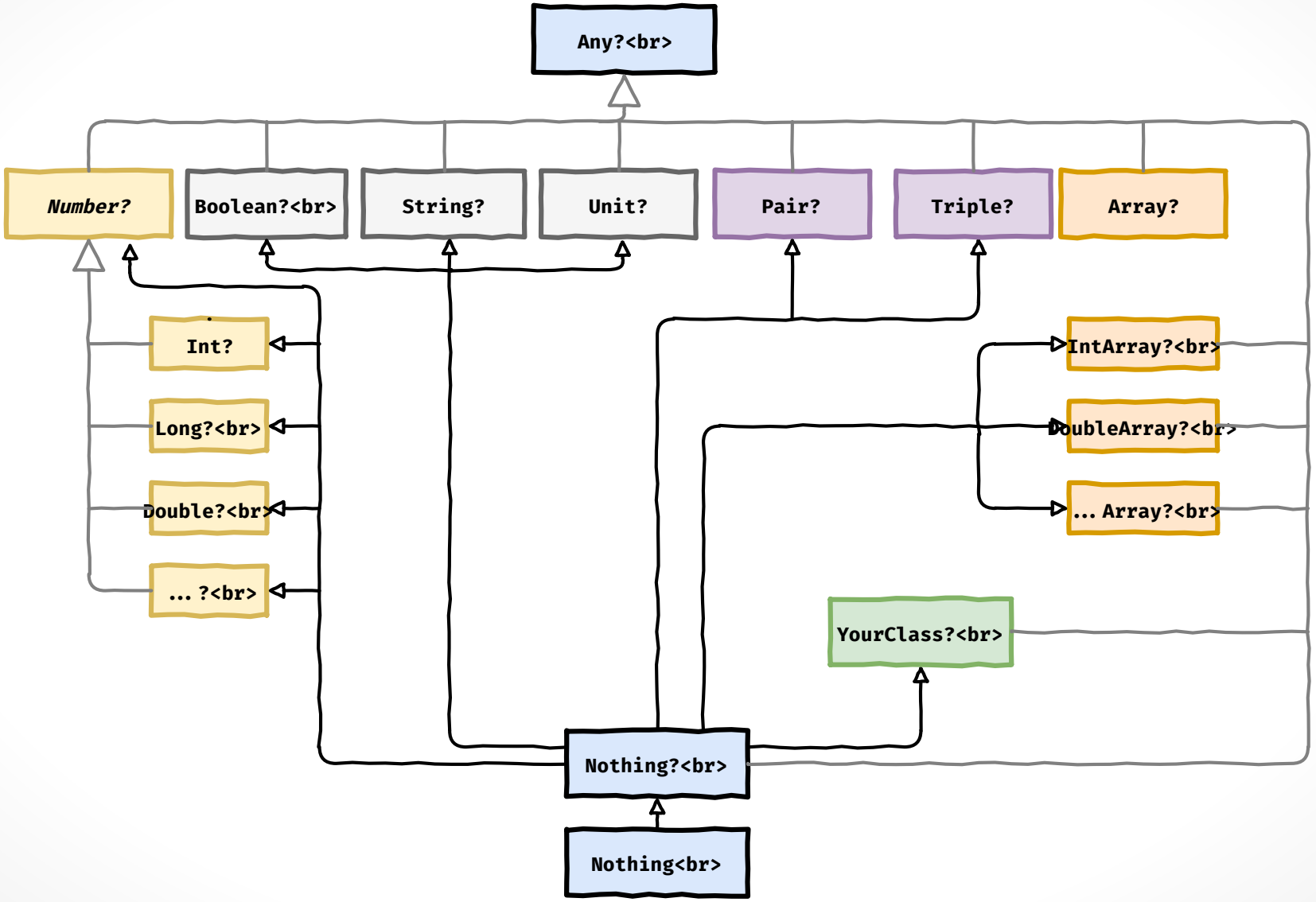
Pas de `Optional`, si nécessaire voir ➽ Arrow <https://arrow-kt.io/>

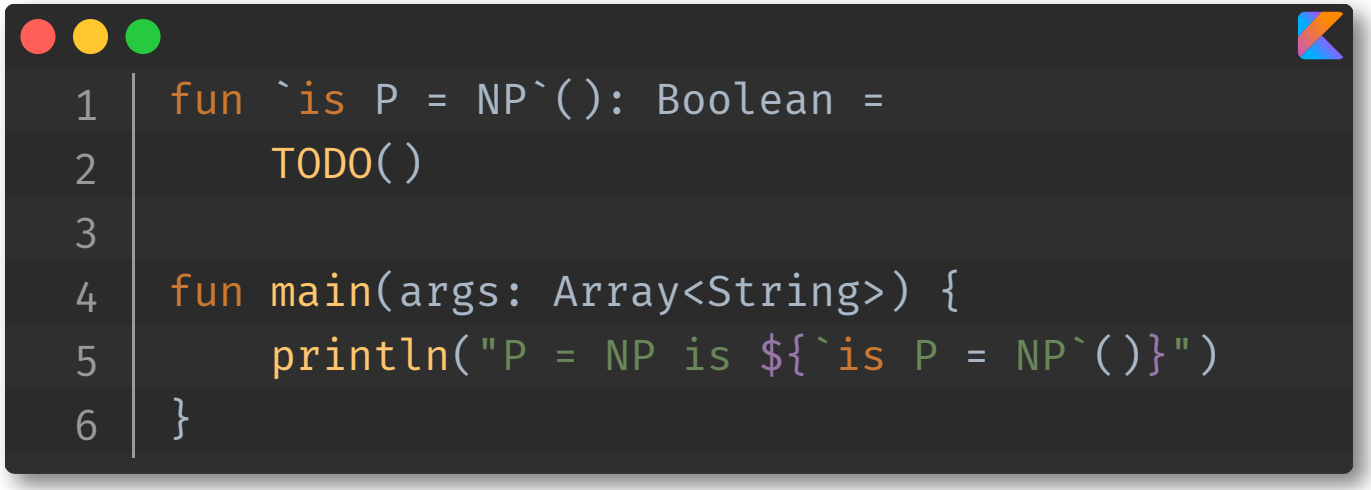
Les types

Types basiques



Types basiques nullable





```kotlin
fun `is P = NP`(): Boolean =
    TODO()

fun main(args: Array<String>) {
    println("P = NP is ${`is P = NP`()}")
}
```

Hiérarchie des types

🤝 le `TODO()` est l'ami du TDD

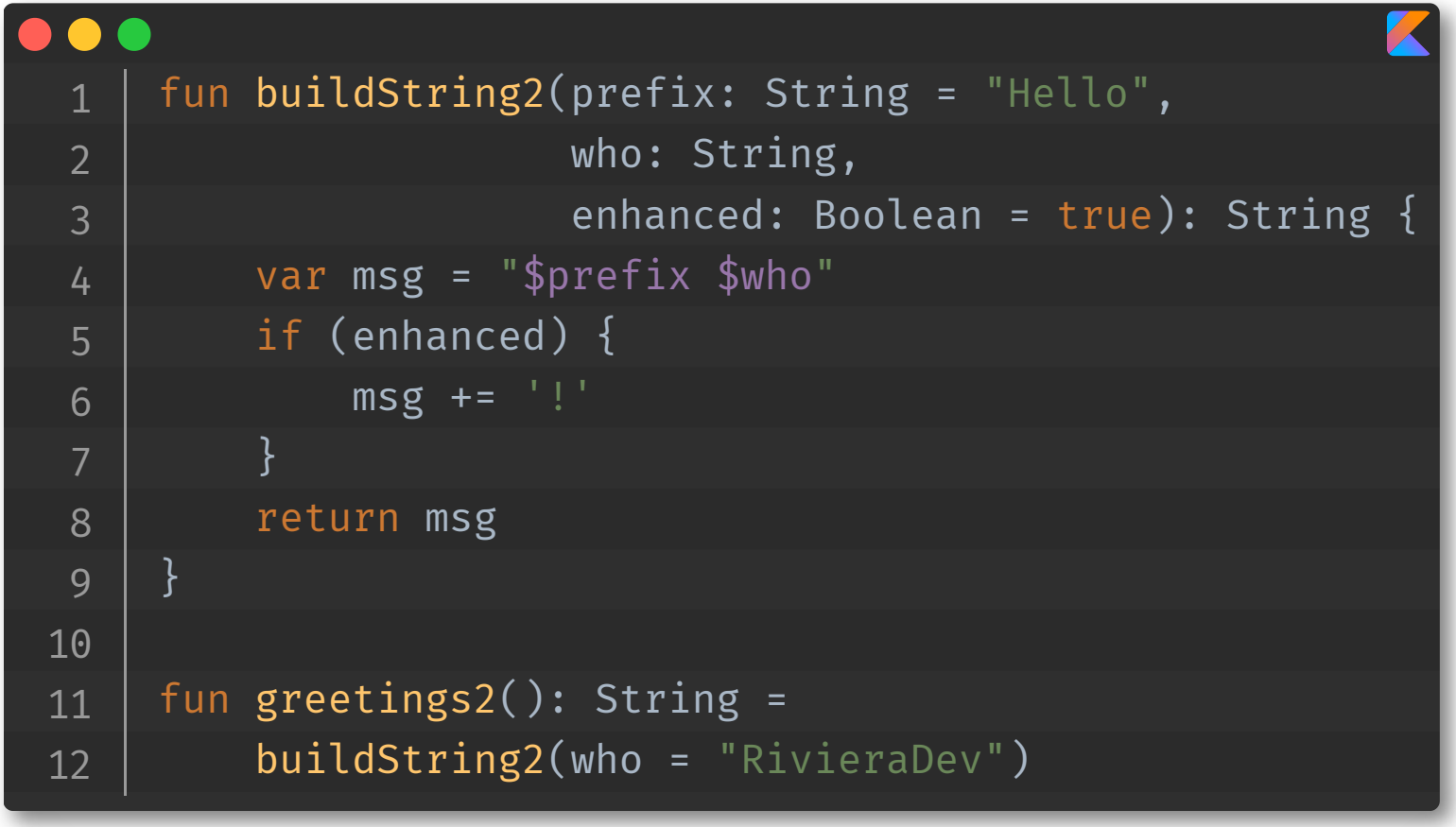
Les fonctions

```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 = "RivieraDev", prefix = "Hello"
```

named-params.java


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

@Metadata(
        mv = {1, 1, 15},
        bv = {1, 0, 3},
        k = 2,
        d1 = {"\u0000\u0012\n\u0000\n\u0002\u0010\u000e\n\u0002\b\u0003\n\u0002\u0010\u000b\n\u0002\b\u0002
        d2 = {"buildString", "", "prefix", "who", "enhanced", "", "greetings", "kotlinDee"}
)
public final class Named_paramsKt {
   @NotNull
   public static final String buildString(@NotNull String prefix, @NotNull String who, boolean enhanced) {
      Intrinsics.checkParameterIsNotNull(prefix, "prefix");
      Intrinsics.checkParameterIsNotNull(who, "who");
      String msg = prefix + ' ' + who;
      if (enhanced) {
         msg = msg + '!';
      }

      return msg;
   }

   @NotNull
   public static final String greetings() {
      String var0 = "Hello";
```

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 = "RivieraDev")
```

Comment ça marche


```kotlin
// In my-app.jar
fun main() {
    println(compute())
}

// In lib-v1.0.0
fun compute(result: Int = 1) = result

// In lib-v1.0.1
fun compute(result: Int = 42) = result
```

Compile `my-app` avec `lib-v1.0.0`

```
java my-app.jar -cp lib-v1.0.0.jar
```

```
java my-app.jar -cp lib-v1.0.1.jar
```

Résultat ?

default-value.java


```java
import kotlin.Metadata;

@Metadata(
        mv = {1, 1, 15},
        bv = {1, 0, 3},
        k = 2,
        d1 = {"\u0000\b\n\u0000\n\u0002\u0010\u0002\n\u0000\u001a\u0006\u0010\u0000\u001a\u00020\u0001¨\u00
        d2 = {"main", "", "kotlinDee"}
)
public final class My_appKt {
    public static final void main() {
        int var0 = LibKt.compute$default(0, 1, (Object)null);
        boolean var1 = false;
        System.out.println(var0);
    }

    // $FF: synthetic method
    public static void main(String[] var0) {
        main();
    }
}
```

ByteCode de default-value


```java
import kotlin.Metadata;

@Metadata(
        mv = {1, 1, 15},
        bv = {1, 0, 3},
        k = 2,
        d1 = {"\u0000\n\n\u0000\n\u0002\u0010\b\n\u0002\b\u0002\u001a\u0010\u0010\u0000\u001a\u00020\u00012
        d2 = {"compute", "", "result", "kotlinDee"}
)
public final class LibKt {
    public static final int compute(int result) {
        return result;
    }

    // $FF: synthetic method
    public static int compute$default(int var0, int var1, Object var2) {
        if ((var1 & 1) ≠ 0) {
            var0 = 42;
        }

        return compute(var0);
    }
}
```

Kotlin c'est fun !

Toujours typer le retour de vos fonctions

(sauf si c'est évident et une surcharge comme le `toString`)

Kotlin est plus concis 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 ambiguïtés

Les lambdas

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

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

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

// store function reference
val sumLam = ::sumf

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

function.java

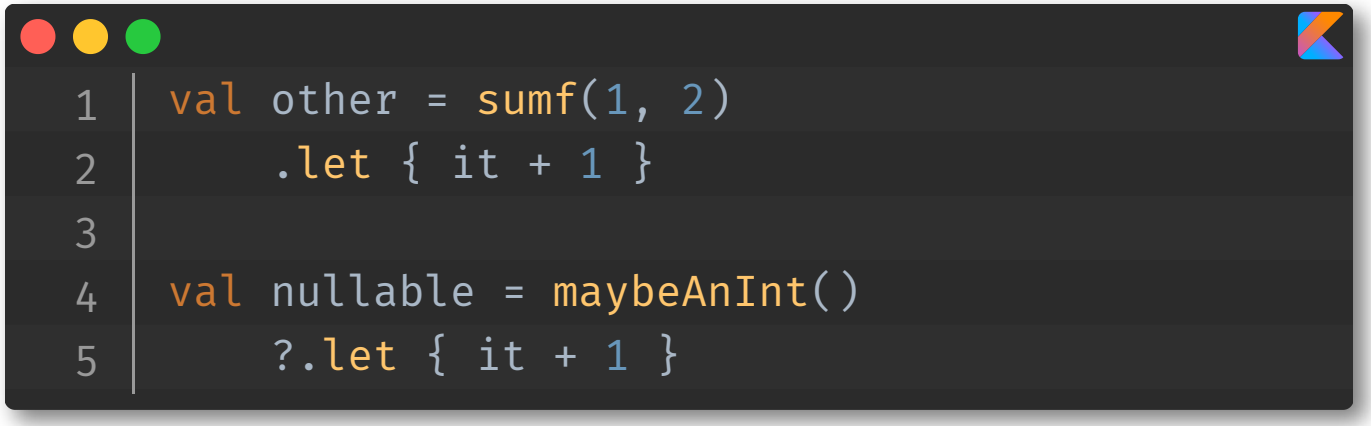

```java
import kotlin.Metadata;
import kotlin.jvm.functions.Function2;
import kotlin.jvm.internal.Intrinsics;
import kotlin.reflect.KFunction;
import org.jetbrains.annotations.NotNull;

@Metadata(
        mv = {1, 1, 15},
        bv = {1, 0, 3},
        k = 2,
        d1 = {"\u0000\u001e\n\u0000\n\u0002\u0010\b\n\u0002\b\u0005\n\u0002\u0018\u0002\n\u0002\u0018\u0002
        d2 = {"sum5", "", "getSum5", "()I", "sum6", "getSum6", "sumLam", "Lkotlin/reflect/KFunction2;", "Lk
)
public final class FunctionKt {
   private static final int sum5;
   @NotNull
   private static final KFunction sumLam;
   private static final int sum6;

   public static final int compute(int x, int y, @NotNull Function2 operation) {
      Intrinsics.checkParameterIsNotNull(operation, "operation");
      return ((Number)operation.invoke(x, y)).intValue();
   }

   public static final int sumf(int x, int y) {
      return x + y;
   }
```

```kotlin
// Declare lambda
val suml: (Int, Int) -> Int = { x: Int, y: Int ->
    x + y
}

// call compute with suml lambda
val sum3 = compute(1, 2, suml)

// call compute with lamda
val sum4 = compute(1, 3) { x, y ->
    x + y
}
```

lambda.java


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

@Metadata(
        mv = {1, 1, 15},
        bv = {1, 0, 3},
        k = 2,
        d1 = {"\u0000\u0012\n\u0000\n\u0002\u0010\b\n\u0002\b\u0005\n\u0002\u0018\u0002\n\u0002\b\u0003\"\u
        d2 = {"sum3", "", "getSum3", "()I", "sum4", "getSum4", "suml", "Lkotlin/Function2;", "getSuml", "()
)
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() {
      return suml;
   }

   public static final int getSum3() {
      return sum3;
   }

   public static final int getSum4() {
```

let.kt



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

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

let.java


```java
import kotlin.Metadata;
import org.jetbrains.annotations.Nullable;

@Metadata(
        mv = {1, 1, 15},
        bv = {1, 0, 3},
        k = 2,
        d1 = {"\u0000\n\n\u0000\n\u0002\u0010\b\n\u0002\b\u0007\"\u0015\u0010\u0000\u001a\u0004\u0018\u0001
        d2 = {"nullable", "", "getNullable", "()Ljava/lang/Integer;", "Ljava/lang/Integer;", "other", "getO
)
public final class LetKt {
   private static final int other;
   @Nullable
   private static final Integer nullable;

   public static final int getOther() {
      return other;
   }

   @Nullable
   public static final Integer getNullable() {
      return nullable;
   }

   static {
      int var0 = FunctionKt.sumf(1, 2);
      boolean var1 = false;
```

Les lambdas

⚠️ pas de `return`

pensez à mettre vos lambda comme dernier argument

voir aussi les `apply`, `also`, `run` , `use`, `with`

➽ the tldr; on Kotlin's let, apply, also, with and run functions

<https://proandroiddev.com/the-tldr-on-kotlins-let-apply-also-with-and-run-functions-6253f06d152b>

Les classes

astronomy.kt

```kotlin
interface AstronomicalBody {
    val name: String
}

data class Planet(override val name: String,
                  val moons: List<Moon> = emptyList()) : AstronomicalBody {
    init {
        require(name.isNotEmpty())
    }

    operator fun plus(moon: Moon): Planet {
        return this.copy(moons = moons + moon)
    }
}

data class Moon(override val name: String) : AstronomicalBody

object SolarSystem {
    val moon = Moon(name = "Moon")
    val earth = Planet(name = "Earth") + moon

    val bodies: List<AstronomicalBody> = listOf(
        earth,
        Planet(name = "Jupiter")
    )
}

```

astronomy.java


```java
import java.util.Collection;
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;
import org.jetbrains.annotations.Nullable;

@Metadata(
        mv = {1, 1, 15},
        bv = {1, 0, 3},
        k = 1,
        d1 = {"\u00000\n\u0002\u0018\u0002\n\u0002\u0018\u0002\n\u0000\n\u0002\u0010\u000e\n\u0000\n\u0002\
        d2 = {"LPlanet;", "LAstronomicalBody;", "name", "", "moons", "", "LMoon;", "(Ljava/lang/String;Ljav
)
public final class Planet implements AstronomicalBody {
   @NotNull
   private final String name;
   @NotNull
   private final List moons;

   @NotNull
   public final Planet plus(@NotNull Moon moon) {
      Intrinsics.checkParameterIsNotNull(moon, "moon");
      return copy$default(this, (String)null, CollectionsKt.plus((Collection)this.moons, moon), 1, (Object)
   }
```

Héritage en Kotlin


```kotlin
open class SmallBody {
    open fun sizeRange(): IntRange = 0..10
}

// Inherit SmallBody
data class Comet(val name: String): SmallBody()

// Inherit SmallBody
data class Asteroid(val name: String): SmallBody() {
    override fun sizeRange(): IntRange = 0..4
}

fun main(args: Array<String>) {
    val bodies = listOf(Comet("Halley"), Asteroid("Adeona"))

    bodies.forEach { body →
        println("$body: ${body.sizeRange()}")
    }
}
```

Génériques

⚠️ Les contrôles de types génériques ne sont faits qu'au moment de la compilation

Covariant: `out`, en java `? extends T`

Contravariant: `in`, en java `? super T`

Projection `*` correspondant à `Any?` ou `Nothing`

Borne supérieure

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

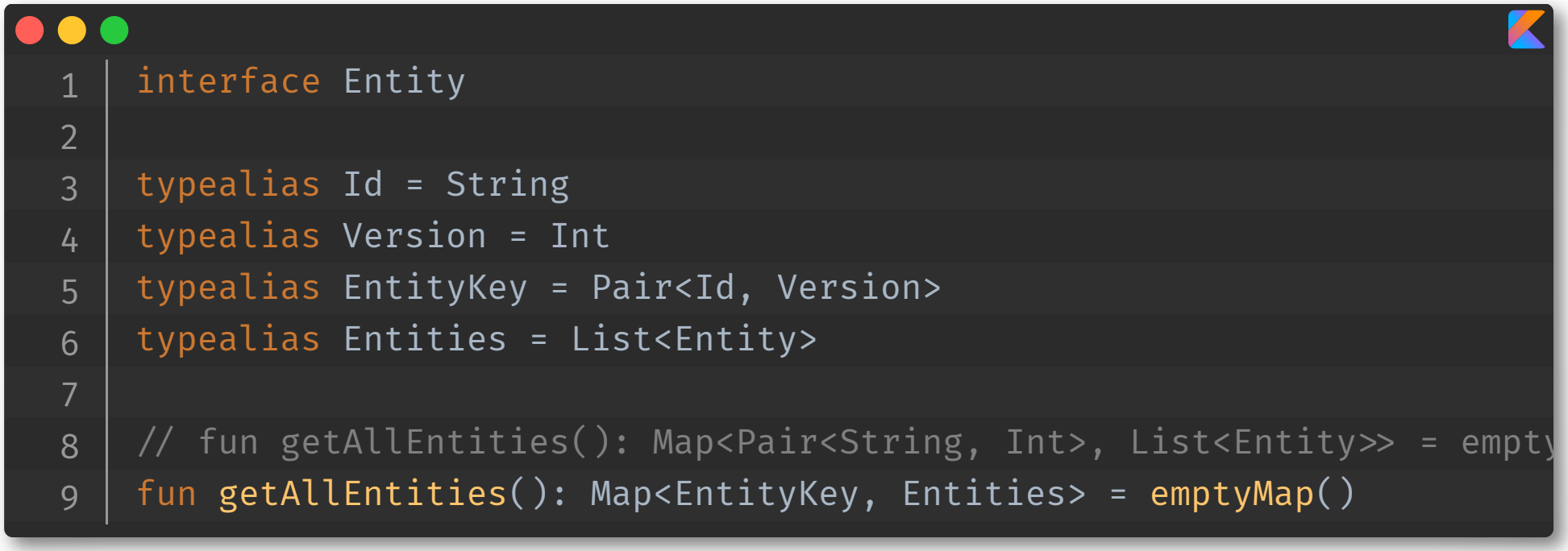
Les détails: ➳ https://kotlinlang.org/docs/reference/generics.html

<https://kotlinlang.org/docs/reference/generics.html>

Sealed


```kotlin
sealed class JsonValue

data class JsonObject(val attributes: Map<String, JsonValue>) : JsonV
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>
typealias Entities = List<Entity>

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

ByteCode d'alias


```
Compiled from "typealias.kt"
public final class TypealiasKt {
  public static final java.util.Map<kotlin.Pair<java.lang.String, java.lang.Integer>, java.util.List<Entity
    Code:
       0: invokestatic  #13                 // Method kotlin/collections/MapsKt.emptyMap:()Ljava/util/Map;
       3: areturn
}
```

Classe, le bilan

🤩 `data class`

🤔 Mais pourquoi on n'a pas ça en Java ?

➽ JEP draft: Records and Sealed Types <https://openjdk.java.net/jeps/8222777>

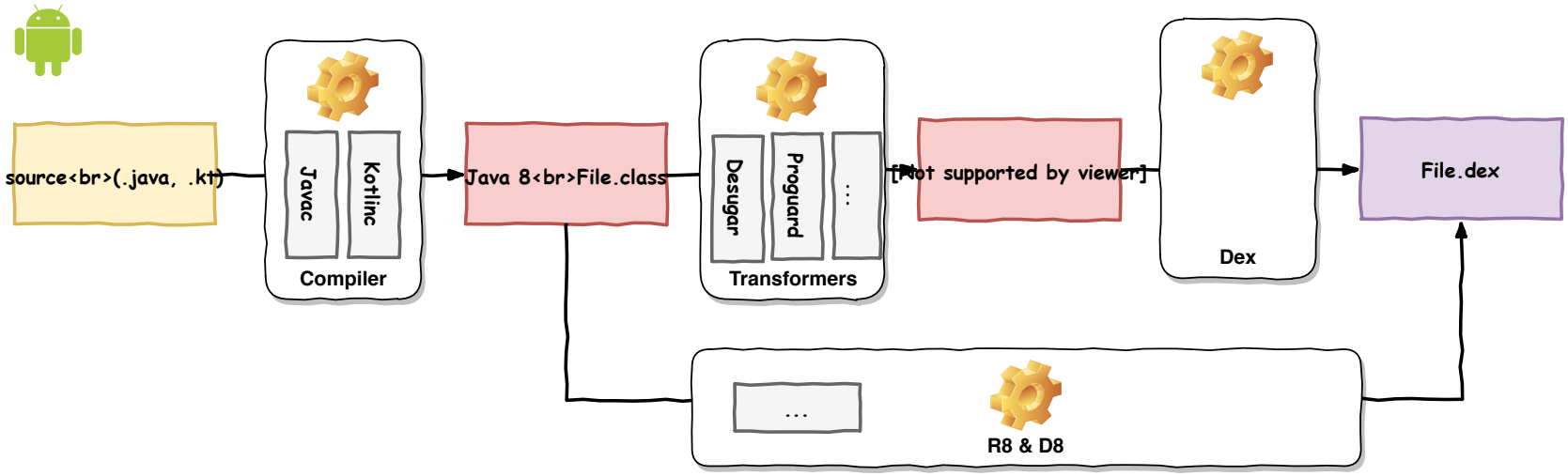
Une seule classe par fichier n'est pas utile

🤓 `sealed` permet de faire des types algébriques de données (Algebraic Data Type)

Pause

ByteCode Android

Compilation pour Android



Dalvik EXecutable format

➽ Dalvik Executable format <https://source.android.com/devices/tech/dalvik/dex-format>

```
1 java -jar ./scripts/lib/d8.jar --release \
2              --output ./target/android/dex \
3              ./target/android/hello.jar
```

```
1  00000000  64 65 78 0a 30 33 35 00  06 50 f0 61 50 10 7c c0  |dex.035..P.aP.|.|
2  00000010  b2 f9 77 2d 54 df 60 f3  ac dd b0 10 eb 55 53 e1  |..w-T.`......US.|
3  00000020  28 f6 10 00 70 00 00 00  78 56 34 12 00 00 00 00  |(...p...xV4.....|
4  00000030  00 00 00 00 4c f5 10 00  12 17 00 00 70 00 00 00  |....L.......p...|
5  00000040  31 03 00 00 b8 5c 00 00  2a 08 00 00 7c 69 00 00  |1....\..*...|i..|
6  00000050  30 03 00 00 74 cb 00 00  20 1a 00 00 f4 e4 00 00  |0...t... .......|
7  00000060  75 02 00 00 f4 b5 01 00  94 f1 0e 00 94 04 02 00  |u...............|
8  00000070  4a f1 07 00 4c f1 07 00  75 f1 07 00 a4 f1 07 00  |J...L...u.......|
9  00000080  c2 f1 07 00 e0 f1 07 00  00 f2 07 00 23 f2 07 00  |............#...|
10 00000090  71 f2 07 00 b9 f2 07 00  07 f3 07 00 37 f3 07 00  |q...........7...|
11 000000a0  69 f3 07 00 90 f3 07 00  bc f3 07 00 e3 f3 07 00  |i...............|
12 000000b0  27 f4 07 00 71 f4 07 00  8f f4 07 00 ad f4 07 00  |'...q...........|
13 000000c0  cb f4 07 00 ed f4 07 00  0f f5 07 00 2c f5 07 00  |............,...|
14 000000d0  49 f5 07 00 79 f5 07 00  b1 f5 07 00 e9 f5 07 00  |I...y...........|
15 000000e0  1a f6 07 00 51 f6 07 00  7b f6 07 00 a6 f6 07 00  |....Q...{.......|
16 000000f0  d1 f6 07 00 0a f7 07 00  47 f7 07 00 84 f7 07 00  |........G.......|
17 00000100  c1 f7 07 00 02 f8 07 00  43 f8 07 00 84 f8 07 00  |........C.......|
18 00000110  f1 f8 07 00 12 f9 07 00  41 f9 07 00 6e f9 07 00  |........A...n...|
19 00000120  b0 f9 07 00 dd f9 07 00  0a fa 07 00 39 fa 07 00  |............9...|
20 00000130  68 fa 07 00 ab fa 07 00  e0 fa 07 00 25 fb 07 00  |h...........%...|
```

dexdump

```
~/.android-sdk/build-tools/23.0.1/dexdump -d \
      ./target/android/dex/classes.dex \
      > ./target/android/dex/classes.dex.dump
```

```
Processing './target/android/dex/classes.dex'...
Opened './target/android/dex/classes.dex', DEX version '035'
Class #0            -
  Class descriptor  : 'L_00_helloworld/HelloWorldKt;'
  Access flags      : 0x0011 (PUBLIC FINAL)
  Superclass        : 'Ljava/lang/Object;'
  Interfaces        -
  Static fields     -
  Instance fields   -
  Direct methods    -
    #0              : (in L_00_helloworld/HelloWorldKt;)
      name          : 'main'
      type          : '([Ljava/lang/String;)V'
      access        : 0x0019 (PUBLIC STATIC FINAL)
      code          -
      registers     : 2
      ins           : 1
      outs          : 2
      insns size    : 13 16-bit code units
020494:                                        |[020494] 00 helloworld.HelloWorldKt.main:([Ljava/lang/Stri
```

```
sh ./scripts/lib/dextools/d2j-dex2smali.sh \
      ./target/android/dex/classes.dex -f \
      -o ./target/android/smali
```

```
.class public final L_00_helloworld/HelloWorldKt;
.super Ljava/lang/Object;
.source "HelloWorld.kt"

.annotation system Ldalvik/annotation/SourceDebugExtension;
  value = "SMAP\nHelloWorld.kt\nKotlin\n*S Kotlin\n*F\n+ 1 HelloWorld.kt\n_00_helloworld/HelloWorldKt\n*L\n
.end annotation
.annotation runtime Lkotlin/Metadata;
  bv = {
    1,
    0,
    2
  }
  d1 = {
    "\u0000\u0012\n\u0000\n\u0002\u0010\u0002\n\u0000\n\u0002\u0010\u0011\n\u0002\u0010\u000e\n\u0000\u001a
  }
  d2 = {
    "main",
    "",
    "args",
```

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)
}
```

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

Mais aussi des `while` et `do while`

et des `break`, `continue`, et label

```kotlin
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 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
      23: goto          11
      26: iload_1
      27: ireturn
}
```

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 récursivité


```
Compiled from "rec-factorial.kt"
public final class 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 recFactorial:(I)I
      16: imul
      17: ireturn
}
```

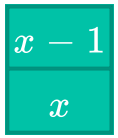
Récursion non terminale

$$x! = x \times (x-1) \times ... \times 2 \times 1$$
$$1! = 0! = 1$$

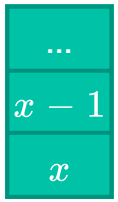
$fact(x)$

$x \times fact(x-1)$

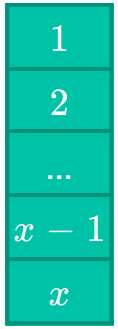
$x \times (x-1) \times fact(x-2)$



$x \times (x-1) \times (x-2) \times ...$



$x \times (x-1) \times (x-2) \times ... \times 2 \times 1$



Récursion terminale

$$fact(x) = fact(x, 1)$$

$$fact(x - 1, x \times 1)$$

$$fact(x - 2, x \times (x - 1))$$

$$fact(..., x \times (x - 1) \times (x - 2) \times ...)$$

$$fact(1, x \times (x - 1) \times (x - 2) \times ... \times 2)$$

⚠️ Nécessite une optimisation par le compilateur

tailrec-factorial.kt

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

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

    return aux(n, 1)
}
```

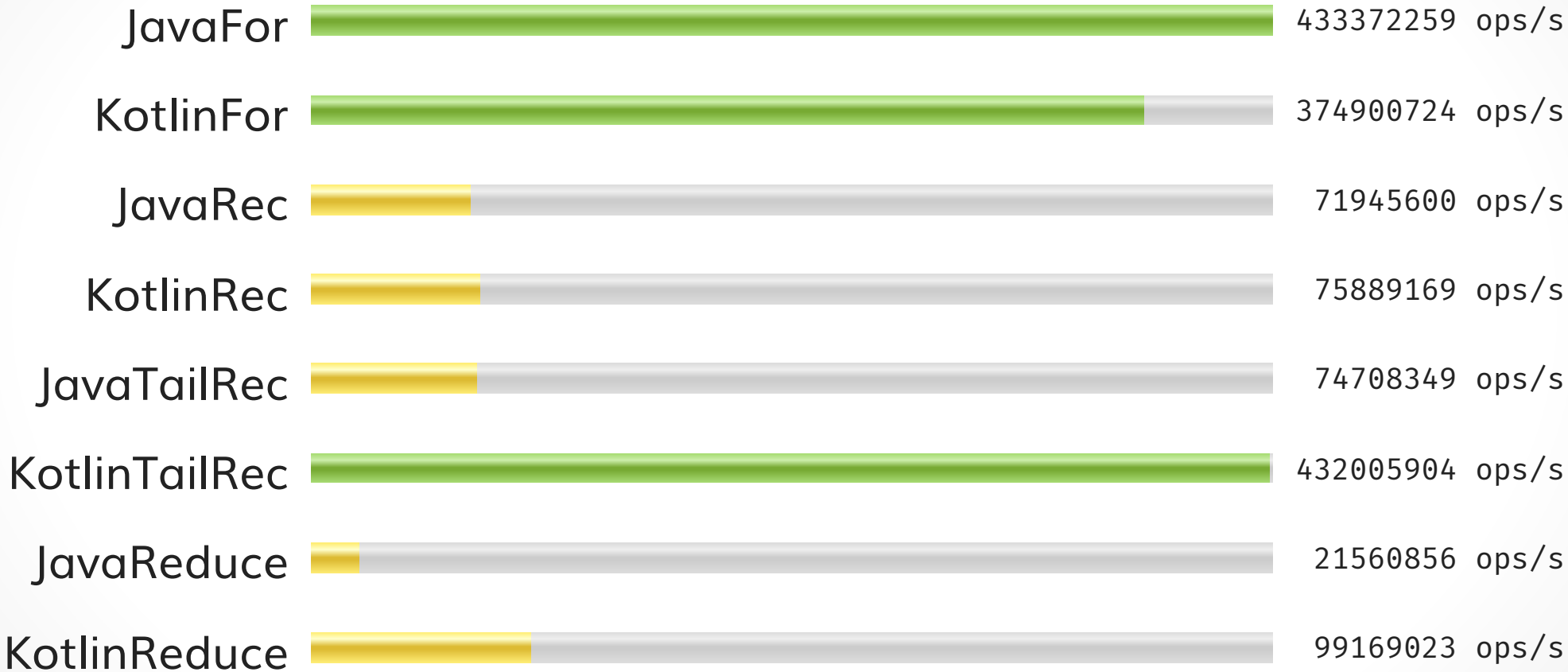
ByteCode factoriel avec recursivité terminale 1/2


```
public final class Tailrec_factorialKt {
  public static final int tailRecFactorial(int);
    Code:
       0: getstatic     #12                 // Field Tailrec_factorialKt$tailRecFactorial$1.INSTANCE:LTailr
       3: astore_1
       4: aload_1
       5: iload_0
       6: iconst_1
       7: invokevirtual #16                 // Method Tailrec_factorialKt$tailRecFactorial$1.invoke:(II)I
      10: ireturn
}
```

ByteCode factoriel avec recursivité terminale 2/2


```
Compiled from "tailrec-factorial.kt"
final class Tailrec_factorialKt$tailRecFactorial$1 extends kotlin.jvm.internal.Lambda implements kotlin.jvm
  public static final Tailrec_factorialKt$tailRecFactorial$1 INSTANCE;

  public java.lang.Object invoke(java.lang.Object, java.lang.Object);
    Code:
       0: aload_0
       1: aload_1
       2: checkcast     #11                 // class java/lang/Number
       5: invokevirtual #15                 // Method java/lang/Number.intValue:()I
       8: aload_2
       9: checkcast     #11                 // class java/lang/Number
      12: invokevirtual #15                 // Method java/lang/Number.intValue:()I
      15: invokevirtual #18                 // Method invoke:(II)I
      18: invokestatic  #24                 // Method java/lang/Integer.valueOf:(I)Ljava/lang/Integer;
      21: areturn

  public final int invoke(int, int);
    Code:
       0: iload_1
       1: iconst_1
       2: if_icmpge     9
       5: iload_2
       6: goto          25
       9: aload_0
      10: checkcast     #2                  // class Tailrec_factorialKt$tailRecFactorial$1
      13: pop
```

Performances sur 10!



Bilan structures

`when` peut être utilisé avec

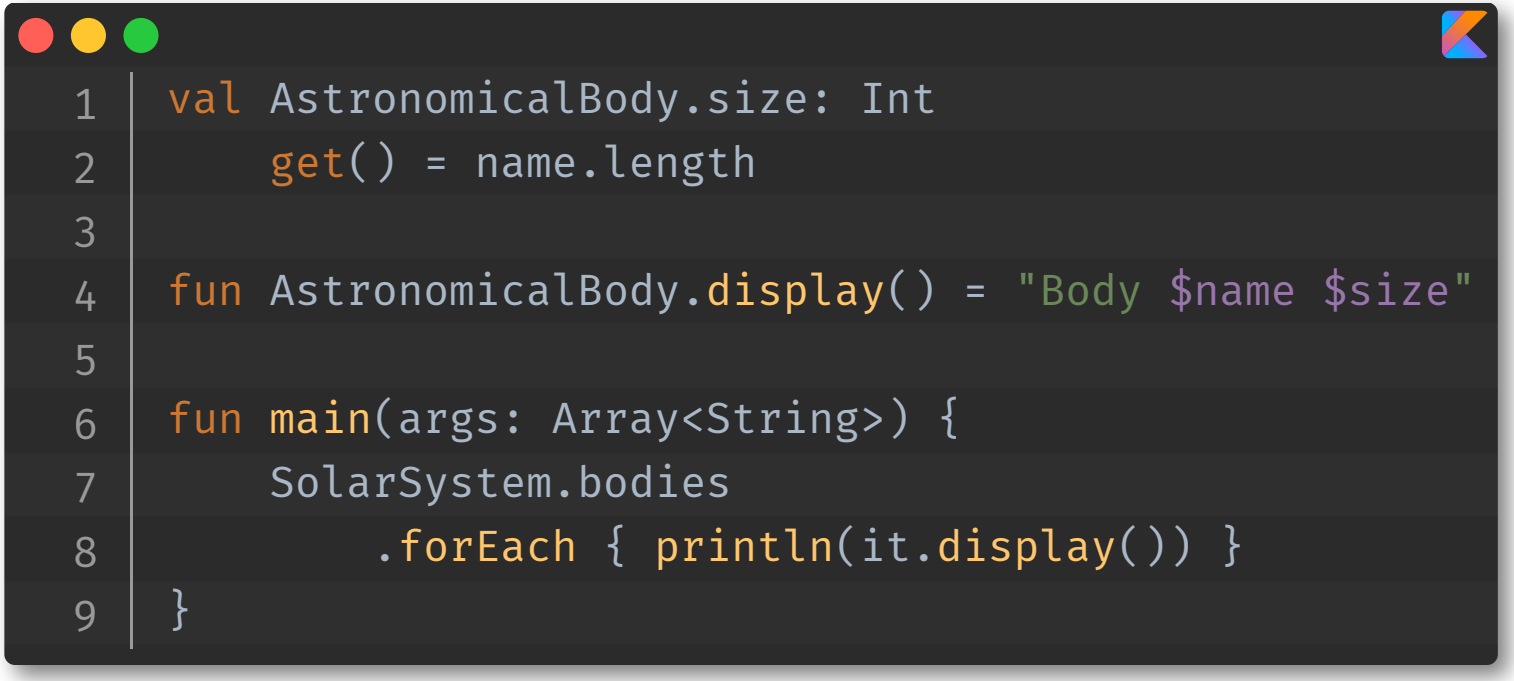
- des constantes
- plusieurs valeurs séparées par `,`
- une expression
- avec `is` et un type (avec un 'smart cast')

✨ Tips

privilégier les `when` si vous avez plus de 2 cas

si vous faites des fonctions récursives, faites les `tailrec`

Extensions de fonctions



```kotlin
val AstronomicalBody.size: Int
    get() = name.length

fun AstronomicalBody.display() = "Body $name $size"

fun main(args: Array<String>) {
    SolarSystem.bodies
        .forEach { println(it.display()) }
}
```

extension.java


```java
import java.util.Iterator;
import kotlin.Metadata;
import kotlin.jvm.internal.Intrinsics;
import org.jetbrains.annotations.NotNull;

@Metadata(
        mv = {1, 1, 15},
        bv = {1, 0, 3},
        k = 2,
        d1 = {"\u0000 \n\u0000\n\u0002\u0010\b\n\u0002\u0018\u0002\n\u0002\b\u0003\n\u0002\u0010\u0002\n\u0
        d2 = {"size", "", "LAstronomicalBody;", "getSize", "(LAstronomicalBody;)I", "main", "", "args", "",
)
public final class ExtensionKt {
   public static final int getSize(@NotNull AstronomicalBody $this$size) {
      Intrinsics.checkParameterIsNotNull($this$size, "$this$size");
      return $this$size.getName().length();
   }

   @NotNull
   public static final String display(@NotNull AstronomicalBody $this$display) {
      Intrinsics.checkParameterIsNotNull($this$display, "$this$display");
      return "Body " + $this$display.getName() + ' ' + getSize($this$display);
   }

   public static final void main(@NotNull String[] args) {
      Intrinsics.checkParameterIsNotNull(args, "args");
      Iterable $this$forEach$iv = (Iterable)SolarSystem.INSTANCE.getBodies();
```

Extension

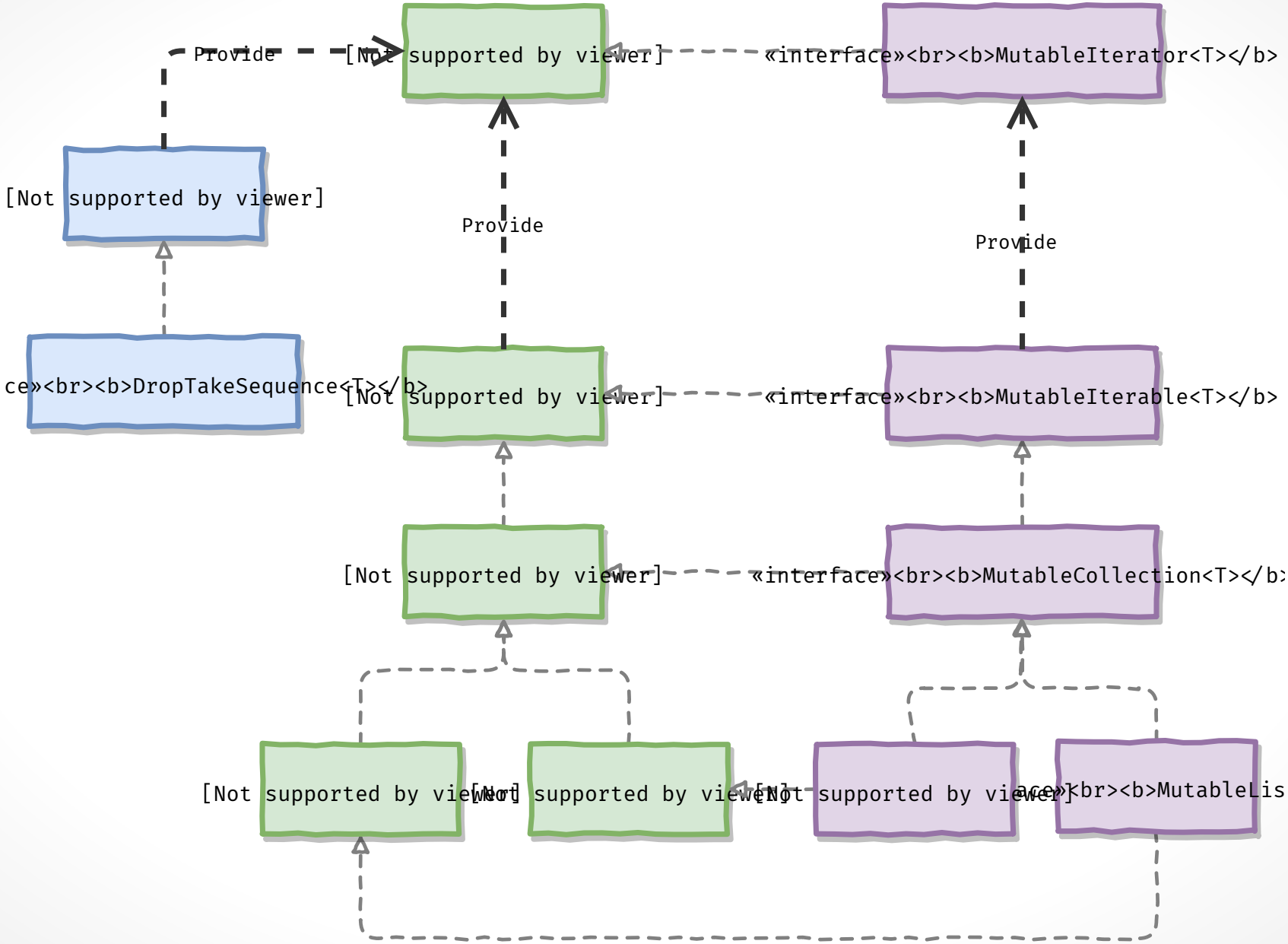
Permet d'enrichir les APIs Java

- ➳ Android KTX <https://developer.android.com/kotlin/ktx>
- ➳ Spring <https://docs.spring.io/spring/docs/current/spring-framework-reference/languages.html#kotlin-spring-projects-in-kotlin>
- ➳ RxKotlin <https://github.com/ReactiveX/RxKotlin>
- ...

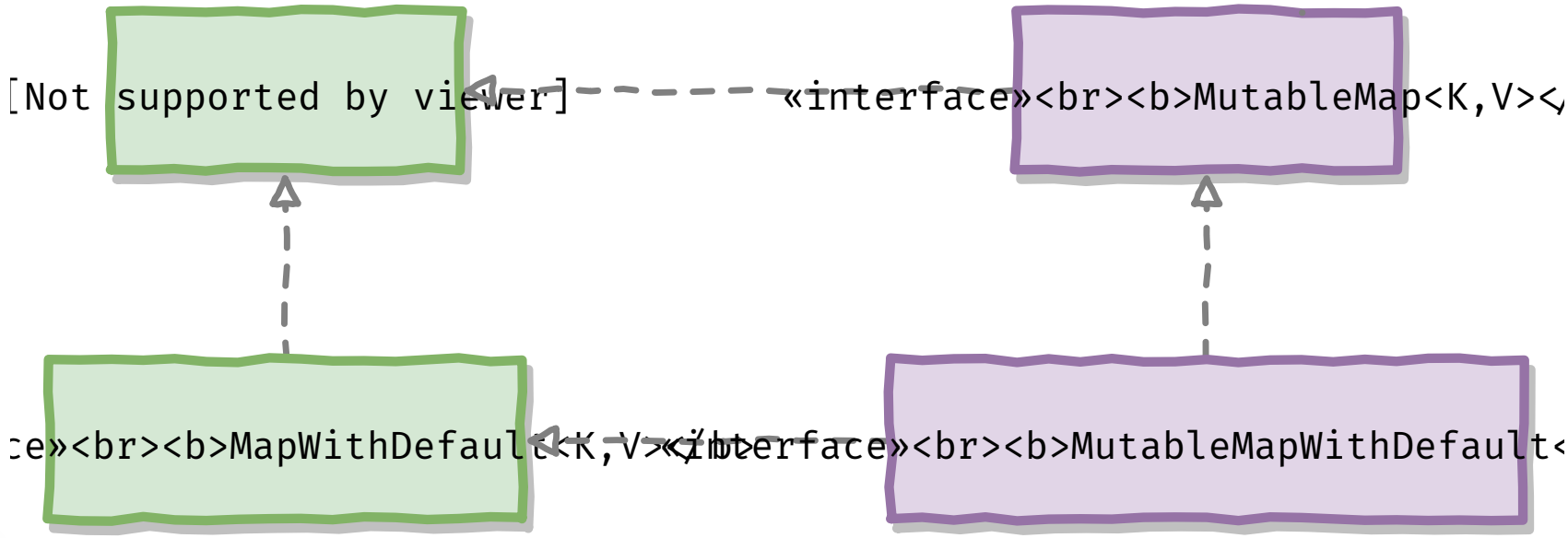
Permet la *SoC* (Separation of Concerns)

Les collections

Collections



Les Maps


[Not supported by viewer]
«interface»
<b>MutableMap<K,V></
ce»
<b>MapWithDefault<K,V></
«interface»
<b>MutableMapWithDefault<

```kotlin
val s = SolarSystem.bodies
    .filterIsInstance<Planet>()
    .flatMap { planet → planet.moons } // 😻
    .filterNot { it.name.startsWith("S/") }
    .sortedBy { it.name }
//        .fold("") { acc, moon →
//            (if (acc == "") "" else "$acc,\n") + moon.name
//        }
    .joinToString(",\n") { it.name }

println(s)
```

break-immutable.kt


```kotlin
fun main(args: Array<String>) {
    val moons = (1..9).map { Moon("Moon #$it") }.toList()

    println(moons.javaClass) // class java.util.ArrayList

    moons.javaClass.methods
        .find { it.name == "add" && it.parameterCount == 1 }
        ?.invoke(moons, Moon("XXX"))

    println(moons.joinToString("\n"))
    // Moon(name=Moon #1)
    // Moon(name=Moon #2)
    // Moon(name=Moon #3)
    // Moon(name=Moon #4)
    // Moon(name=Moon #5)
    // Moon(name=Moon #6)
    // Moon(name=Moon #7)
    // Moon(name=Moon #8)
    // Moon(name=Moon #9)
    // Moon(name=XXX)
}
```

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

println(s)
```

Performance des séquences 1/2

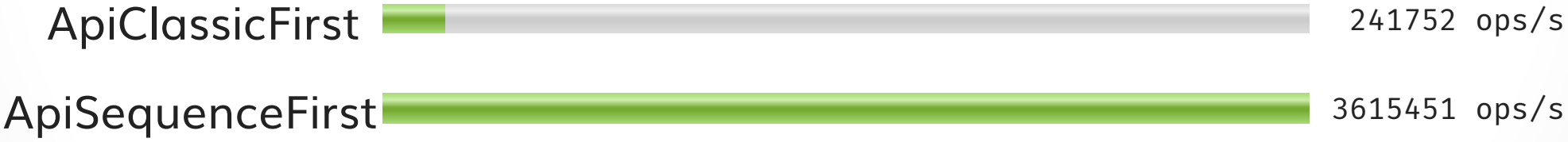
Benchmark	Mode	Cnt	Score	Error	Units
ApiClassic	thrpt	200	44535.029	3550.944	ops/s
ApiSequence	thrpt	200	23652.238	1967.535	ops/s

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 séquences 2/2

Benchmark	Mode	Cnt	Score	Error	Units
ApiClassicFirst	thrpt	200	241752.062	5022.663	ops/s
ApiSequenceFirst	thrpt	200	3615451.391	454502.198	ops/s



Bilan collection

💪 Super on a de l'immutabilité, des `map`, `flatMap`, `fold`, `aggregate`,...

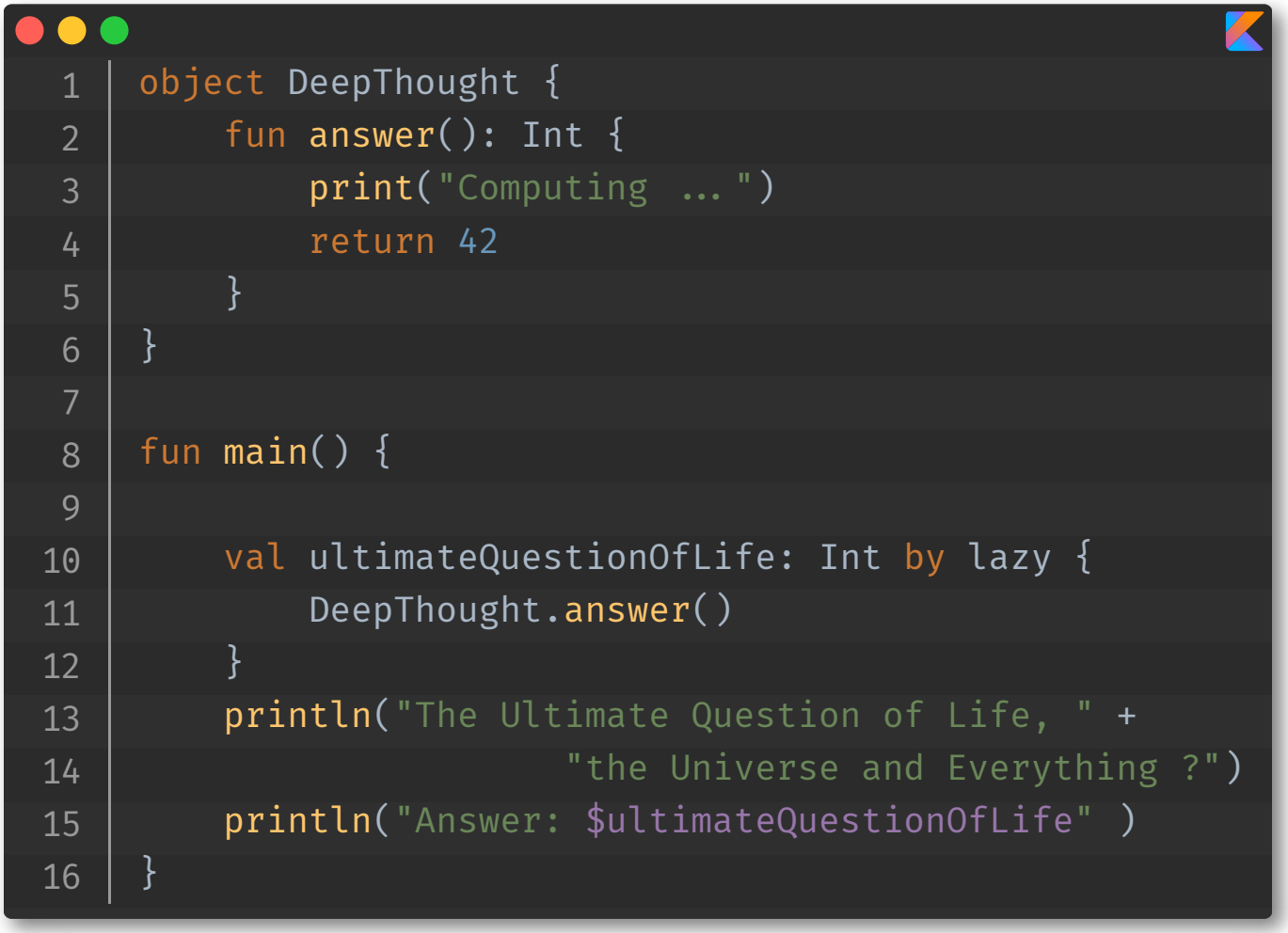
😐 Mais ça reste des collections Java

API standard avec `Range`, `Pair`, et `Triple`

📏 Avant d'utiliser les `Sequence`, faites des mesures

Les delegates

```kotlin
import kotlin.properties.ReadOnlyProperty
import kotlin.reflect.KProperty

fun main(args: Array<String>) {
    val value: String by MyDelegateClass()
    println(value)
}

class MyDelegateClass : ReadOnlyProperty<Nothing?, String> {
    override operator fun getValue(thisRef: Nothing?,
                                   property: KProperty<*>) = "Hello P
}
```

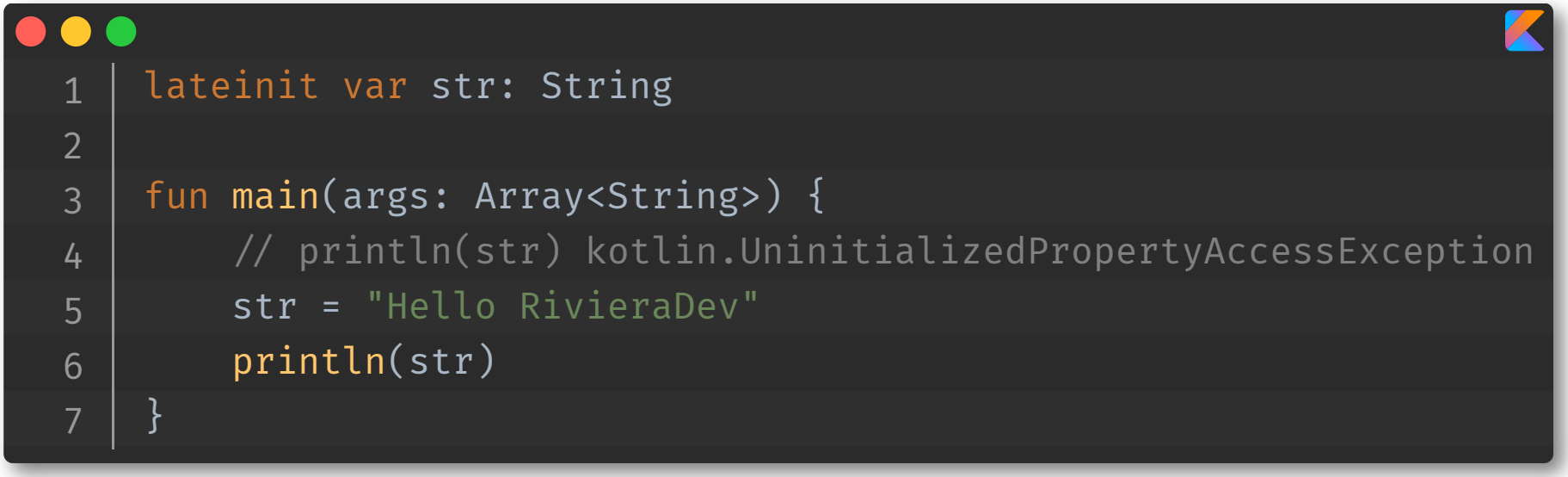


```kotlin
object DeepThought {
    fun answer(): Int {
        print("Computing ...")
        return 42
    }
}

fun main() {

    val ultimateQuestionOfLife: Int by lazy {
        DeepThought.answer()
    }
    println("The Ultimate Question of Life, " +
                    "the Universe and Everything ?")
    println("Answer: $ultimateQuestionOfLife" )
}
```

observable.kt

```kotlin
import kotlin.properties.Delegates

fun main(args: Array<String>) {

    var observable: String by Delegates.observable("Initial value") {
        _, old, new →
            println("$old → $new")
    }

    observable = "new value"
}
```

lateinit.kt



```kotlin
lateinit var str: String

fun main(args: Array<String>) {
    // println(str) kotlin.UninitializedPropertyAccessException
    str = "Hello RivieraDev"
    println(str)
}
```

Delegate

`Lazy` : utile pour les propriétés qui ne sont pas systématiquement utilisées.

⚠️ À manipuler avec précaution dans les activités Android (avec le cycle de vie, cela peut référencer une ancienne instance)

`Delegate` : Observable, Not null, ...

`lateinit` : évite les *null check* pour les propriétés qui ne peuvent être initialisées immédiatement (ex: référence de vues sur `Activity`, `Fragment`).

Ne peut pas être utilisé avec les types primitifs

inline

```kotlin
object Logger {
    var enable: Boolean = false

    inline fun log(msg: () -> String) {
        if (enable) {
            println(msg())
        }
    }
}

fun main() {
    Logger.log { "Hello" }
}
```

Logger.java


```
Compiled from "inline-fun.kt"
public final class Inline_funKt {
  public static final void main();
    Code:
       0: getstatic     #15                 // Field Logger.INSTANCE:LLogger;
       3: astore_0
       4: iconst_0
       5: istore_1
       6: aload_0
       7: invokevirtual #19                 // Method Logger.getEnable:()Z
      10: ifeq          27
      13: iconst_0
      14: istore_2
      15: ldc           #21                 // String Hello
      17: astore_2
      18: iconst_0
      19: istore_3
      20: getstatic     #27                 // Field java/lang/System.out:Ljava/io/PrintStream;
      23: aload_2
      24: invokevirtual #33                 // Method java/io/PrintStream.println:(Ljava/lang/Object;)V
      27: nop
      28: return

  public static void main(java.lang.String[]);
    Code:
       0: invokestatic  #9                  // Method main:()V
       3: return
}
```

```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)
}
```

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

@Metadata(
        mv = {1, 1, 15},
        bv = {1, 0, 3},
        k = 2,
        d1 = {"\u0000\u0014\n\u0000\n\u0002\u0010\u0002\n\u0000\n\u0002\u0010\u0011\n\u0002\u0010\u000e\n\u
        d2 = {"main", "", "args", "", "", "([Ljava/lang/String;)V", "kotlinDee"}
)
public final class ReifiedKt {
    public static final void main(@NotNull String[] args) {
        Intrinsics.checkParameterIsNotNull(args, "args");
        Pojo p1 = new Pojo();
        p1.setName("Plop1");
        boolean var2 = false;
        System.out.println(p1);
        JavaBeanBuilder this_$iv = JavaBeanBuilder.INSTANCE;
        int $i$f$createBean = false;
        Pojo p2 = (Pojo)this_$iv.createBean(Pojo.class);
        p2.setName("Plop2");
        boolean var6 = false;
        System.out.println(p2);
    }
}
```

```kotlin
inline class Latitude(val value: Double)
inline class Longitude(val value: Double)

data class Geoloc(val lat: Latitude, val lng: Longitude) {

    infix fun distanceTo(other: Geoloc): Double = TODO()
}

fun main() {
    val toulouse = Geoloc(Latitude(43.60426), Longitude(1.44367))
    val nice = Geoloc(Latitude(43.70313), Longitude(7.26608))

    val travel = toulouse distanceTo nice
    println(travel)
}
```

geoloc bytecode


```
Compiled from "geoloc.kt"
public final class Geoloc {
  public final double distanceTo(Geoloc);
    Code:
       0: aload_1
       1: ldc           #9                  // String other
       3: invokestatic  #15                 // Method kotlin/jvm/internal/Intrinsics.checkParameterIsNotNul
       6: iconst_0
       7: istore_2
       8: new           #17                 // class kotlin/NotImplementedError
      11: dup
      12: aconst_null
      13: iconst_1
      14: aconst_null
      15: invokespecial #21                 // Method kotlin/NotImplementedError."<init>":(Ljava/lang/Strin
      18: checkcast     #23                 // class java/lang/Throwable
      21: athrow

  public final double getLat();
    Code:
       0: aload_0
       1: getfield      #31                 // Field lat:D
       4: dreturn

  public final double getLng();
    Code:
       0: aload_0
```

Bilan inline

⚠️ les `inline`, `reified`, ... sont à manier avec précaution

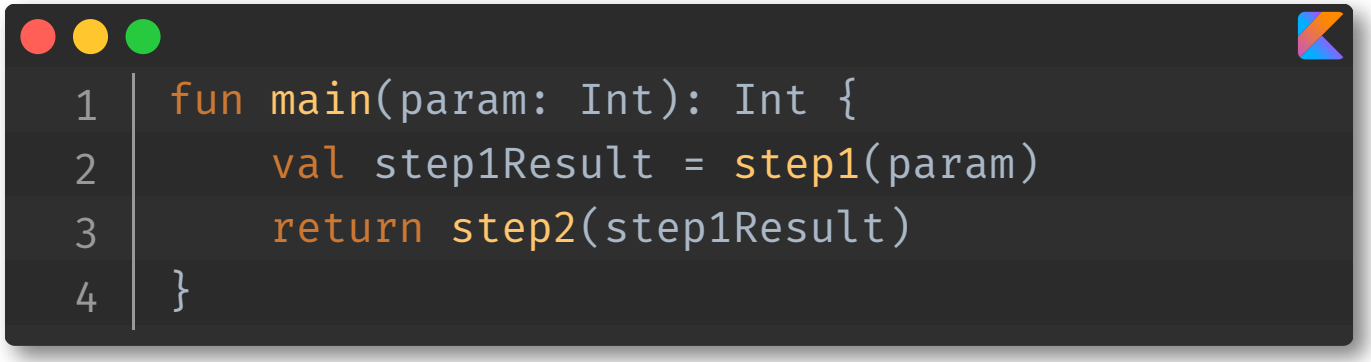
➳ Inline Functions <https://kotlinlang.org/docs/reference/inline-functions.html>

⚠️ les `inline class` sont encore expérimentales

➳ Inline classes <https://kotlinlang.org/docs/reference/inline-classes.html>

Coroutines

Callback hell problem



```kotlin
fun main(param: Int): Int {
    val step1Result = step1(param)
    return step2(step1Result)
}
```

Utilisons un callback


```kotlin
fun main(param: Int): Int {
    val step1Result = step1(param)
    return step2(step1Result)
}
```

```kotlin
fun main(param: Int): Int {
    return step1(param) { step1Result →
        step2(step1Result)
    }
```

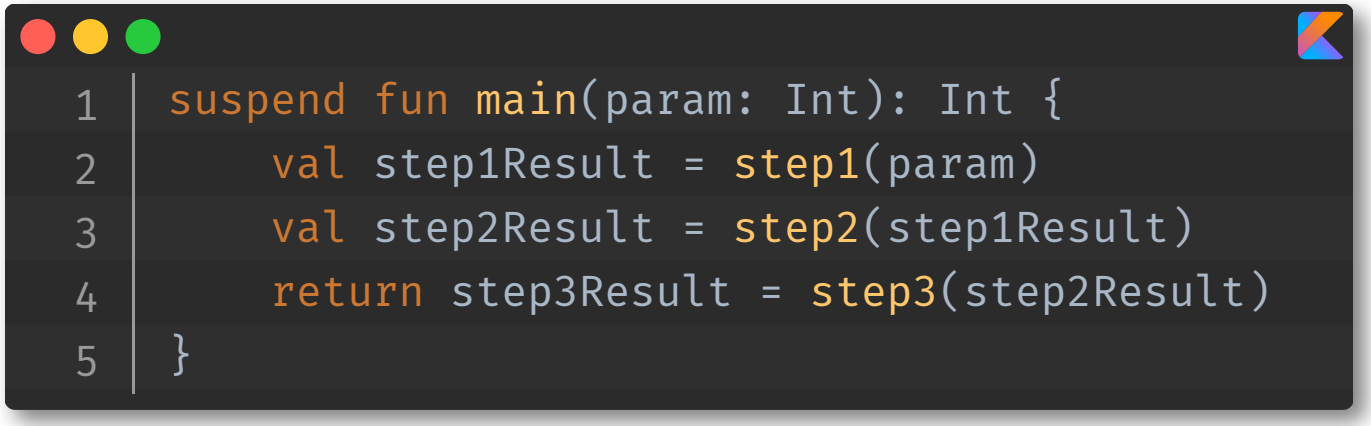
Ajoutons une étape


```kotlin
fun main(param: Int): Int {
    val step1Result = step1(param)
    val step2Result = step2(step1Result)
    return step3Result = step3(step2Result)
}
```

⇒

```kotlin
fun main(param: Int): Int {
    return step1(param) { step1Result →
        step2(step1Result) { step2Result →
            step3(step2Result)
        }
    }
```

suspend



```kotlin
suspend fun main(param: Int): Int {
    val step1Result = step1(param)
    val step2Result = step2(step1Result)
    return step3Result = step3(step2Result)
}
```

Continuation


```kotlin
suspend fun step2(step1Result: Int): Int = TODO()
```

```java
import kotlin.Metadata;
import kotlin.NotImplementedError;
import kotlin.coroutines.Continuation;
import kotlin.jvm.internal.DefaultConstructorMarker;
import org.jetbrains.annotations.NotNull;
import org.jetbrains.annotations.Nullable;

@Metadata(
        mv = {1, 1, 15},
        bv = {1, 0, 3},
        k = 2,
        d1 = {"\u0000\n\n\u0000\n\u0002\u0010\b\n\u0002\b\u0003\u001a\u0019\u0010\u0000\u001a\u00020\u00012
        d2 = {"step2", "", "step1Result", "(ILkotlin/coroutines/Continuation;)Ljava/lang/Object;", "devoxxf
)
public final class BaseKt {
    @Nullable
    public static final Object step2(int step1Result, @NotNull Continuation $completion) {
        boolean var2 = false;
```

➳ API Continuation <https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.coroutines.experimental/-continuation/index.html>

Plus sur les coroutines

- Channel (expérimental)
- Context
- dispatchers
- Jobs
- Scope

On atteind les ➳ limites du décompilateur <https://discuss.kotlinlang.org/t/decompiled-file-is-extremely-large-when-it-includes-coroutine-builder-functions-such-as-launch-async/10882>

Conclusion

Faible surcharge

Support officiel par Google

- Using Project Kotlin for Android <https://docs.google.com/document/d/1ReS3ep-hjxWA8kZi0YqDbEhCqTt29hG8P44aA9W0DM8/edit>
- Kotlin Guide <https://android.github.io/kotlin-guides/>
- android-ktx <https://github.com/android/android-ktx>
- Kotlin Android Extensions <https://kotlinlang.org/docs/tutorials/android-plugin.html>

Serveur

Supporté officiellement depuis ➳ Spring 5 <https://projects.spring.io/spring-framework/>, ➳ Spring Boot 2 <https://projects.spring.io/spring-boot/>

➳ SparkJava <https://sparktutorials.github.io/2017/01/28/using-spark-with-kotlin.html>, ➳ javalin <https://javalin.io/>

➳ Vert.x <http://vertx.io/docs/vertx-core/kotlin/>

➳ KTor <http://ktor.io/>

...

Web et Natif

Web

Partager du code commun

➽ Use Kotlin with npm, webpack and react <https://blog.jetbrains.com/kotlin/2017/04/use-kotlin-with-npm-webpack-and-react/>

Natif

Faire des applications sans JVM

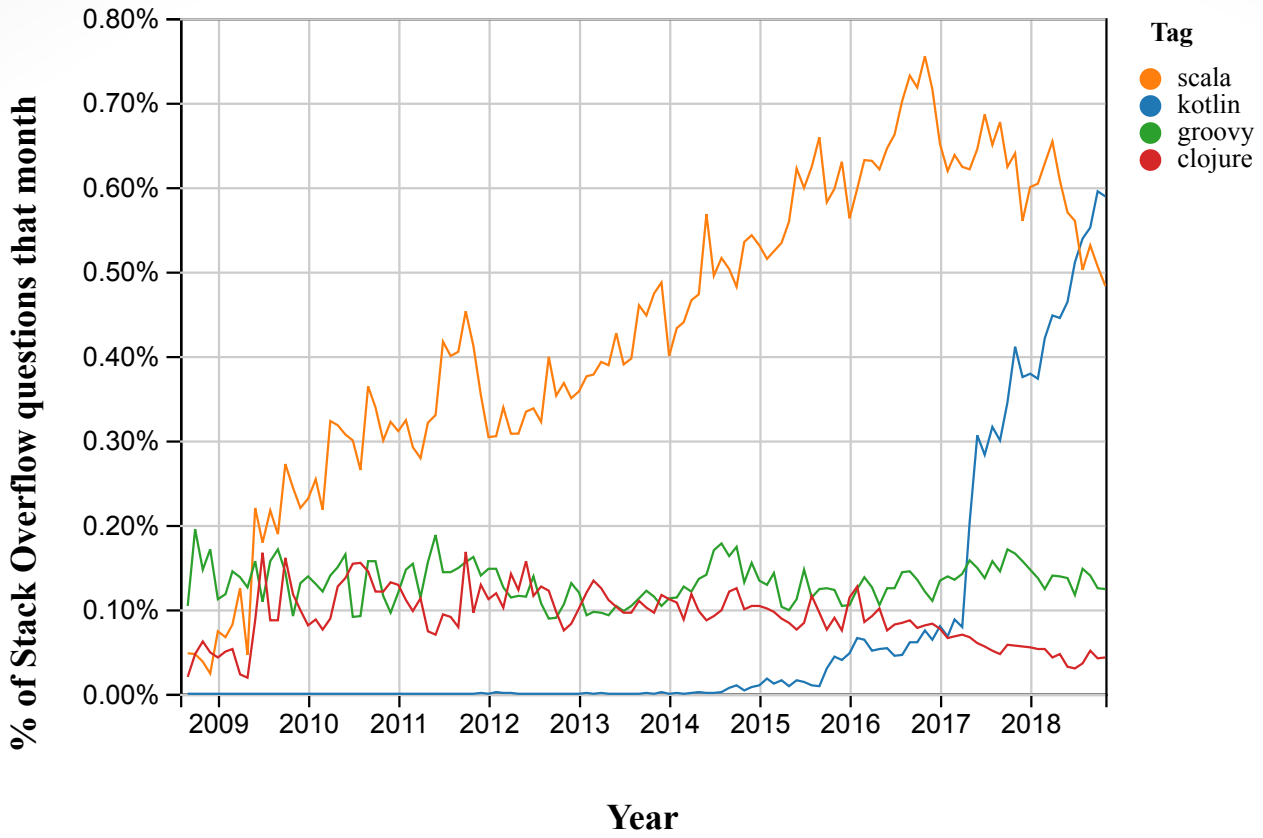
Partager du code avec iOS

WebAssembly

Bilan

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

Tendance



➳ Stackoverflow trends <https://insights.stackoverflow.com/trends?tags=kotlin%2Cscala%2Cgroovy%2Cclojure>

, et ➳ Stackoverflow insights <https://insights.stackoverflow.com/survey/2019#most-loved-dreaded-and-wanted>

➳ The State of the Octoverse <https://octoverse.github.com/projects#languages>

Kotlin vs Java

C'est déjà mature

✊ Code plus expressif, plus sûr, plus simple

🤝 Interopérable avec Java

👍 Outillage (éditeur, gradle, maven)

👍 Ecosystème et communauté

🚀 Évolution rapide

🐣 Code multiplatform

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

Merci

Questions ?