

# 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>>

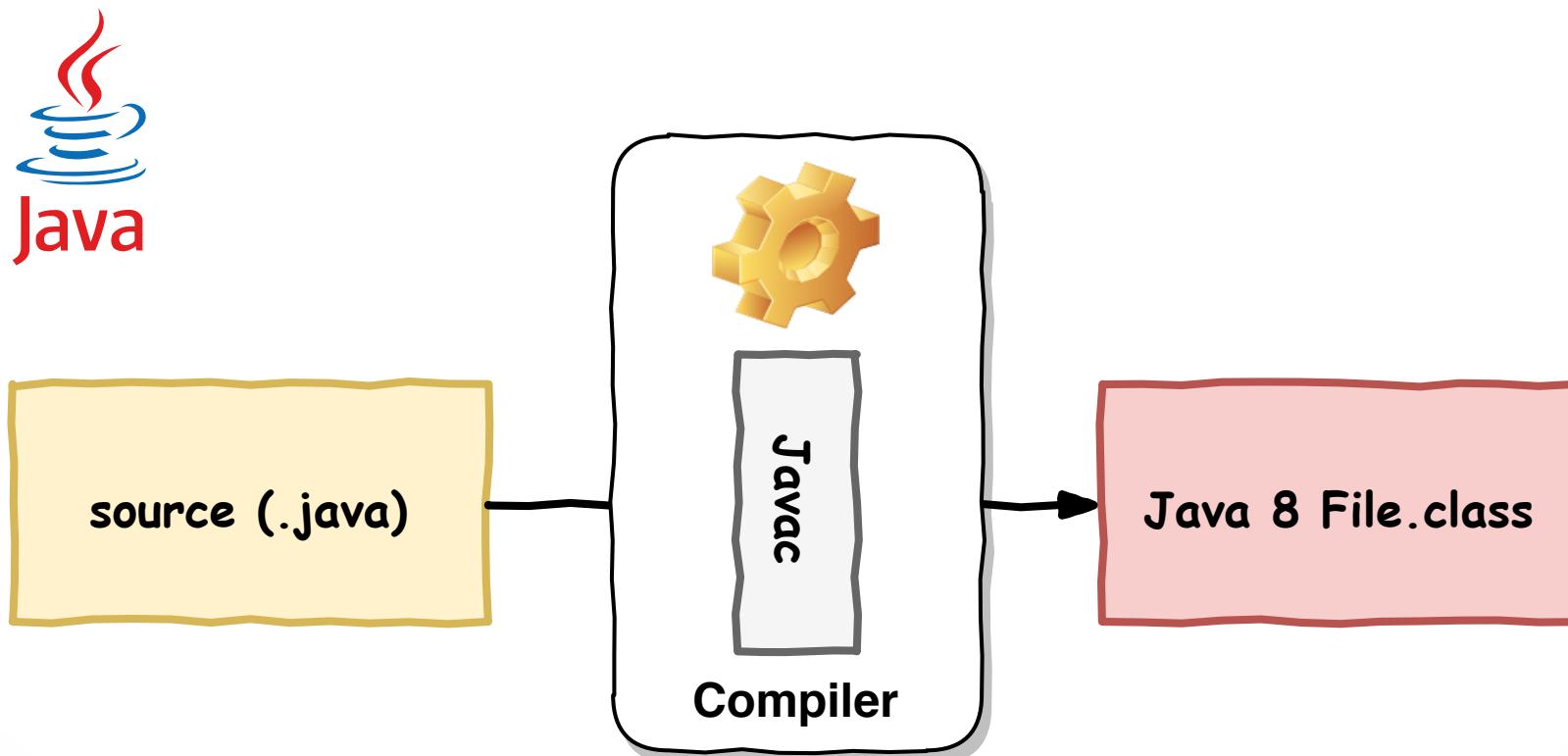
igor@monkeypatch.io <<mailto:igor@monkeypatch.io>>





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

# ByteCode Java ?



# HelloWorld.java

#6

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

```
javac HelloWorld.java
```

# Java ByteCode binary

#7



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 00 |ng;)V.!.....|
00000150  02 00 01 00 07 00 08 00 01 00 09 00 00 00 00 1d 00 | |
```

# Explorons le ByteCode

#8



```
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
        8: return
}
```

Environ 200 opérations possibles (maxi. 256 opcodes)

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

Manipulation de la pile, des variables locales (iconst\_0, istore, iload, ...)

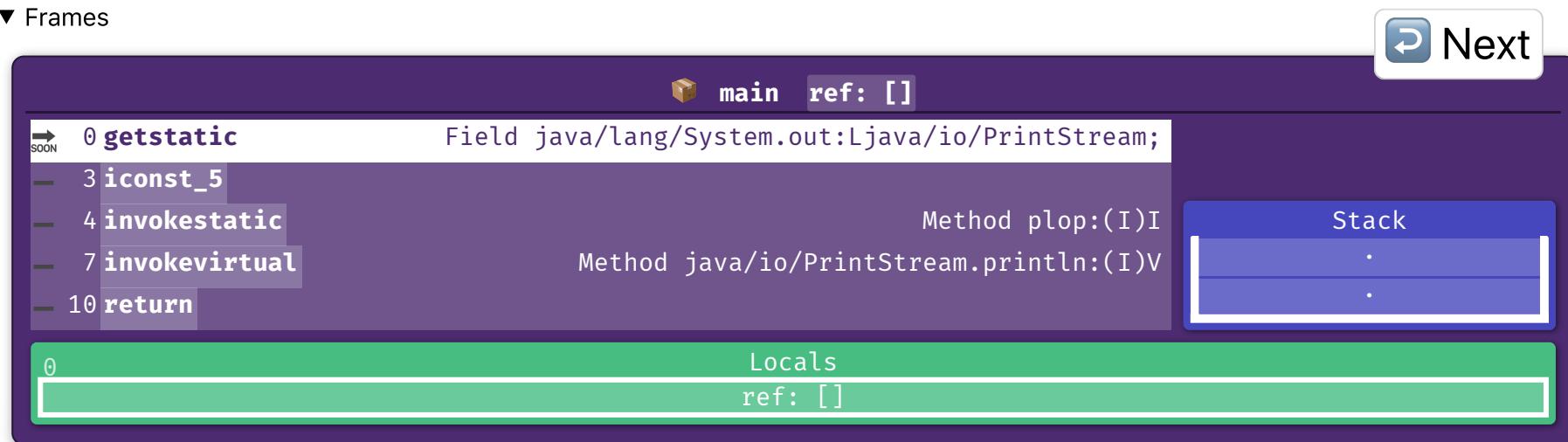
Contrôle du flux des instructions (if\_icmpgt, goto, ...)

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

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

Autres (athrow, ...)

- ▶ Constant Pool
- ▼ Frames



## » 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=rB0ElXf05nU>>

## » Byte Buddy <<http://bytebuddy.net/#/>>

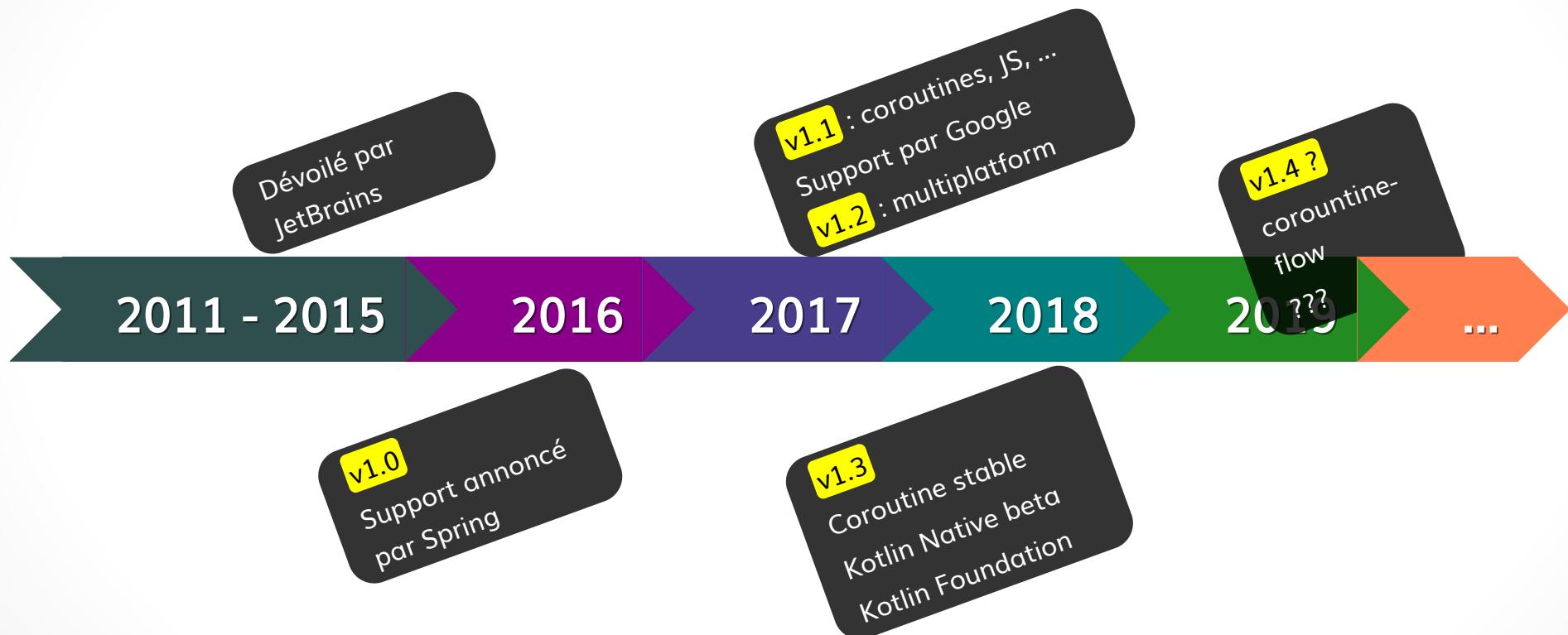
## » asm <<http://asm.ow2.org/>>

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

# Introduction Kotlin

# Historique

#13

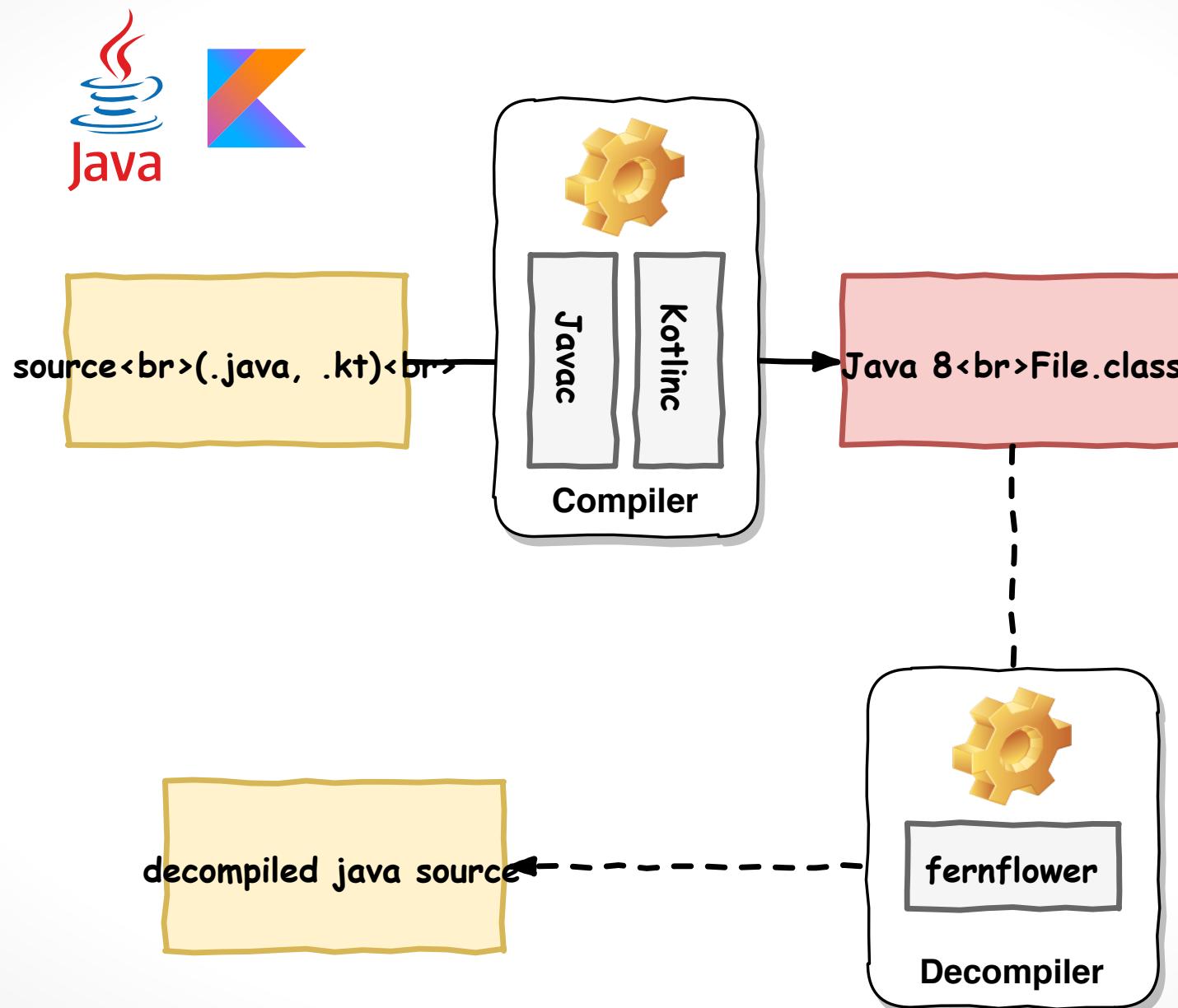




```
package _00_helloworld

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

```
kotlinc HelloWorld.kt
```



	00000000	00000010	00000020	00000030	00000040	00000050	00000060	00000070	00000080	00000090	000000a0	000000b0	000000c0	000000d0	000000e0	000000f0	00000100	00000110	00000120	00000130	00000140	00000150	00000160	00000170	.....2.4 ... _00	_helloworld/Hell	oWorldKt.....ja	va/lang/Object..	....main ... ([Lja	va/lang/String;)	V..#Lorg/jetbrai	ns/annotations/N	otNull; ... args..	....kotlin/jvm/i	nternal/Intrinsic	cs.....checkPar	ameterIsNotNull.	.'(Ljava/lang/Ob	ject;Ljava/lang/	String;)V.....	.....Hello Rivi	eraDev.....java	/lang/System....	.. out ... Ljava/io	/PrintStream; ...	.....java/i	o/PrintStream ...	... println ... (Lj
	ca fe ba be 00 00 00 32 00 34 01 00 1b 5f 30 30  .....2.4 ... _00	5f 68 65 6c 6c 6f 77 6f 72 6c 64 2f 48 65 6c 6c  _helloworld/Hell	6f 57 6f 72 6c 64 4b 74 07 00 01 01 00 10 6a 61  oWorldKt.....ja	76 61 2f 6c 61 6e 67 2f 4f 62 6a 65 63 74 07 00  va/lang/Object..	03 01 00 04 6d 61 69 6e 01 00 16 28 5b 4c 6a 61  ....main ... ([Lja	76 61 2f 6c 61 6e 67 2f 53 74 72 69 6e 67 3b 29  va/lang/String;)	56 01 00 23 4c 6f 72 67 2f 6a 65 74 62 72 61 69  V..#Lorg/jetbrai	6e 73 2f 61 6e 6e 6f 74 61 74 69 6f 6e 73 2f 4e  ns/annotations/N	6f 74 4e 75 6c 6c 3b 01 00 04 61 72 67 73 08 00  otNull; ... args..	08 01 00 1e 6b 6f 74 6c 69 6e 2f 6a 76 6d 2f 69  ....kotlin/jvm/i	6e 74 65 72 6e 61 6c 2f 49 6e 74 72 69 6e 73 69  nternal/Intrinsic	63 73 07 00 0a 01 00 17 63 68 65 63 6b 50 61 72  cs.....checkPar	61 6d 65 74 65 72 49 73 4e 6f 74 4e 75 6c 6c 01  ameterIsNotNull.	00 27 28 4c 6a 61 76 61 2f 6c 61 6e 67 2f 4f 62  .'(Ljava/lang/Ob	6a 65 63 74 3b 4c 6a 61 76 61 2f 6c 61 6e 67 2f  ject;Ljava/lang/	53 74 72 69 6e 67 3b 29 56 0c 00 0c 00 0d 0a 00  String;)V.....	0b 00 0e 01 00 10 48 65 6c 6c 6f 20 52 69 76 69  .....Hello Rivi	65 72 61 44 65 76 08 00 10 01 00 10 6a 61 76 61  eraDev.....java	2f 6c 61 6e 67 2f 53 79 73 74 65 6d 07 00 12 01  /lang/System....	00 03 6f 75 74 01 00 15 4c 6a 61 76 61 2f 69 6f   .. out ... Ljava/io	2f 50 72 69 6e 74 53 74 72 65 61 6d 3b 0c 00 14  /PrintStream; ...	00 15 09 00 13 00 16 01 00 13 6a 61 76 61 2f 69  .....java/i	6f 2f 50 72 69 6e 74 53 74 74 72 65 61 6d 07 00 18  o/PrintStream ...	01 00 07 70 72 69 6e 74 6c 6e 01 00 15 28 4c 6a   ... println ... (Lj																								

```
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/jvm/internal/Intrinsics.checkNotNullParameterI
            6: ldc           #17                 // String Hello RivieraDev
            8: astore_1
            9: iconst_0
            10: istore_2
            11: getstatic     #23                // Field java/lang/System.out:Ljava/io/PrintStream;
            14: aload_1
            15: invokevirtual #29               // Method java/io/PrintStream.println:(Ljava/lang/Object)
            18: return
}
```





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

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



Performances ?

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

➡ <https://github.com/JetBrains/kotlin-benchmarks>

<<https://github.com/JetBrains/kotlin-benchmarks>>

➡ <https://github.com/MonkeyPatchlo/kotlin-perf>

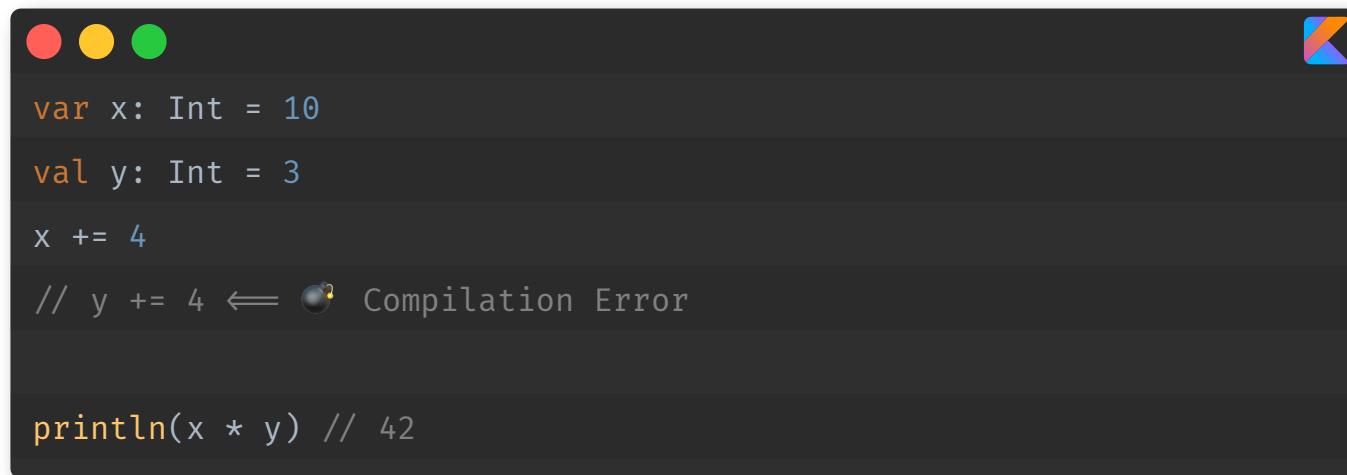
<<https://github.com/MonkeyPatchlo/kotlin-perf>>

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

testJava  66490 ops/s

testKotlin  72393 ops/s

# Les bases



```
var x: Int = 10
val y: Int = 3
x += 4
// y += 4 == ⚡ Compilation Error

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



A screenshot of a dark-themed code editor window. The title bar has three colored window control buttons (red, yellow, green) on the left and a blue 'K' logo on the right. The main area contains the following Kotlin code:

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

# string-template.java

#25

# ByteCode de string-template

#26

```
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: invokespecial #15                // Method kotlin/jvm/internal/Intrinsics.checkNotNullParameterI
            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/Str
            18: aload_0
            19: invokevirtual #30               // Method java/lang/StringBuilder.append:(Ljava/lang/Object)
            22: bipush         33
            24: invokevirtual #33               // Method java/lang/StringBuilder.append:(C)Ljava/lang/S
            27: invokevirtual #37               // Method java/lang/StringBuilder.toString:()Ljava/lang/
            30: astore_1
            31: getstatic      #43                // Field java/lang/System.out:Ljava/io/PrintStream;
            34: aload_1
            35: invokevirtual #49               // Method java/io/PrintStream.println:(Ljava/lang/Object)
            38: new            #17                // class java/lang/StringBuilder
            41: dup
            42: invokespecial #21               // Method java/lang/StringBuilder."<init>":()V
```

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

```
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\u0006\u0010\u0000\u001a\u0020\u0001"},
    d2 = {"tryNumeric", ""}
)
public final class NumericKt {
    public static final void tryNumeric() {
        int anInt = true;
        long aLong = 42L;
        Double aDouble = (Double)null;
    }
}
```

```
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/lang/Double
        11: astore_3
        12: return
    }
```

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>>

References-The-Billion-Dollar-Mistake-Tony-Hoare>

```
fun main(args: Array<String>) {  
  
    val somethingNotNull: String = "aString"  
    // somethingNotNull: String = null => compilation error  
  
    var length = somethingNotNull.length  
  
    var something: String? = null  
    length = something?.length ?: 0  
  
    length = something()?.length ?: 0  
  
    // length = something()!! .length // throw kotlin.NullPointerException  
  
    // SmartCast  
    something = "aString"  
    length = something.length  
}  
  
fun something(): String? = null
```

```
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\u0002\u0010\u0011\n\u0002\u0010\u0000", "main", "", "args", "", "", "([Ljava/lang/String;)V", "something"}
)
public final class NullSafetyKt {
    public static final void main(@NotNull String[] args) {
        Intrinsics.checkNotNull(args, "args");
        String somethingNotNull = "aString";
        int length = somethingNotNull.length();
        String something = (String)null;
        int length = false;
        String var10000 = something();
        length = var10000 != null ? var10000.length() : 0;
        var10000 = something();
    }
}
```

🎸 Elvis operator: ?:

🙌 plus de NullPointerException

⚠ quand on appelle du Java

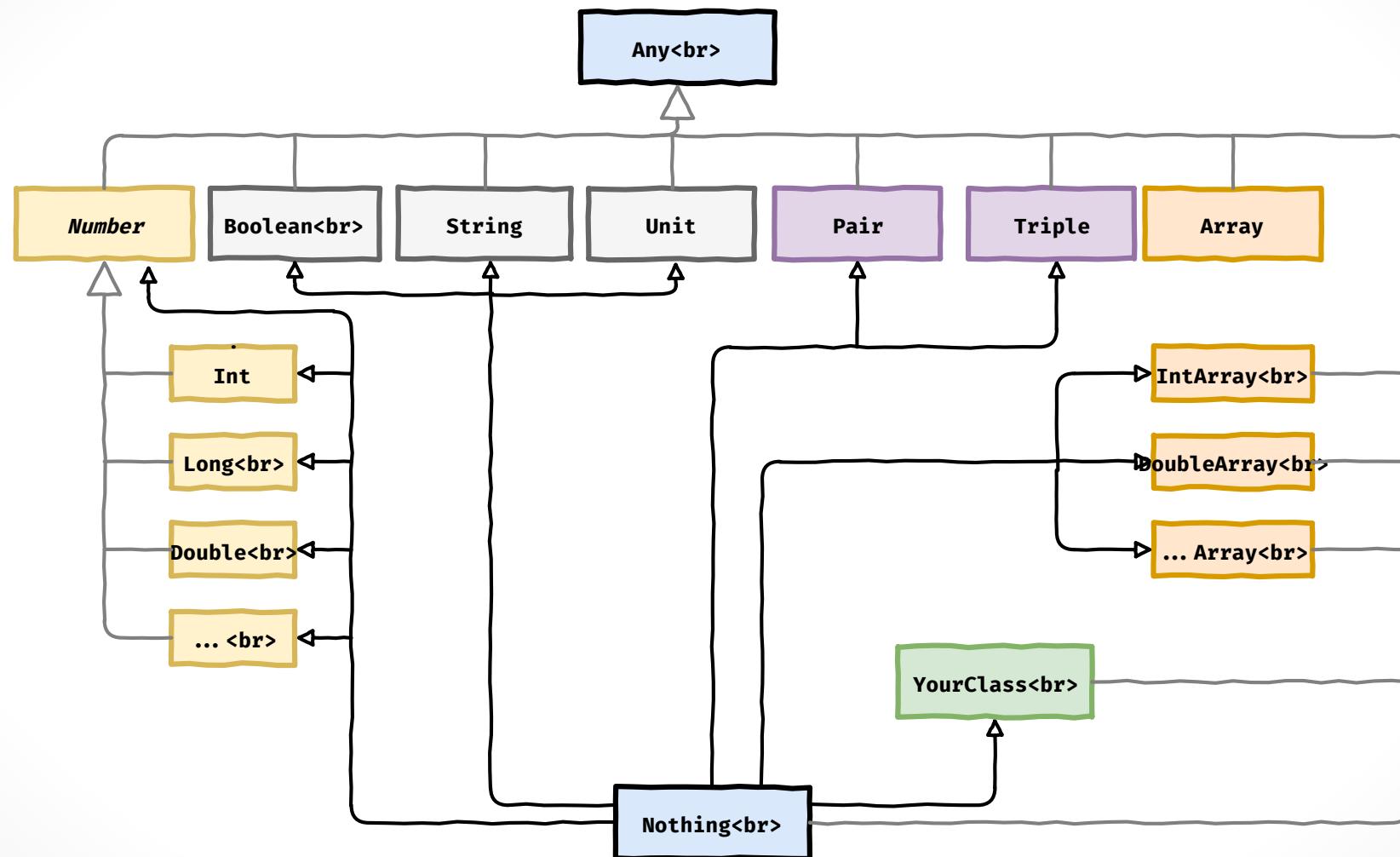
➡ Le compilateur Kotlin peut interpréter des annotations de nullabilité  
(JSR-305, Android, ...) <<https://kotlinlang.org/docs/reference/java-interop.html#nullability-annotations>>

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

# Les types

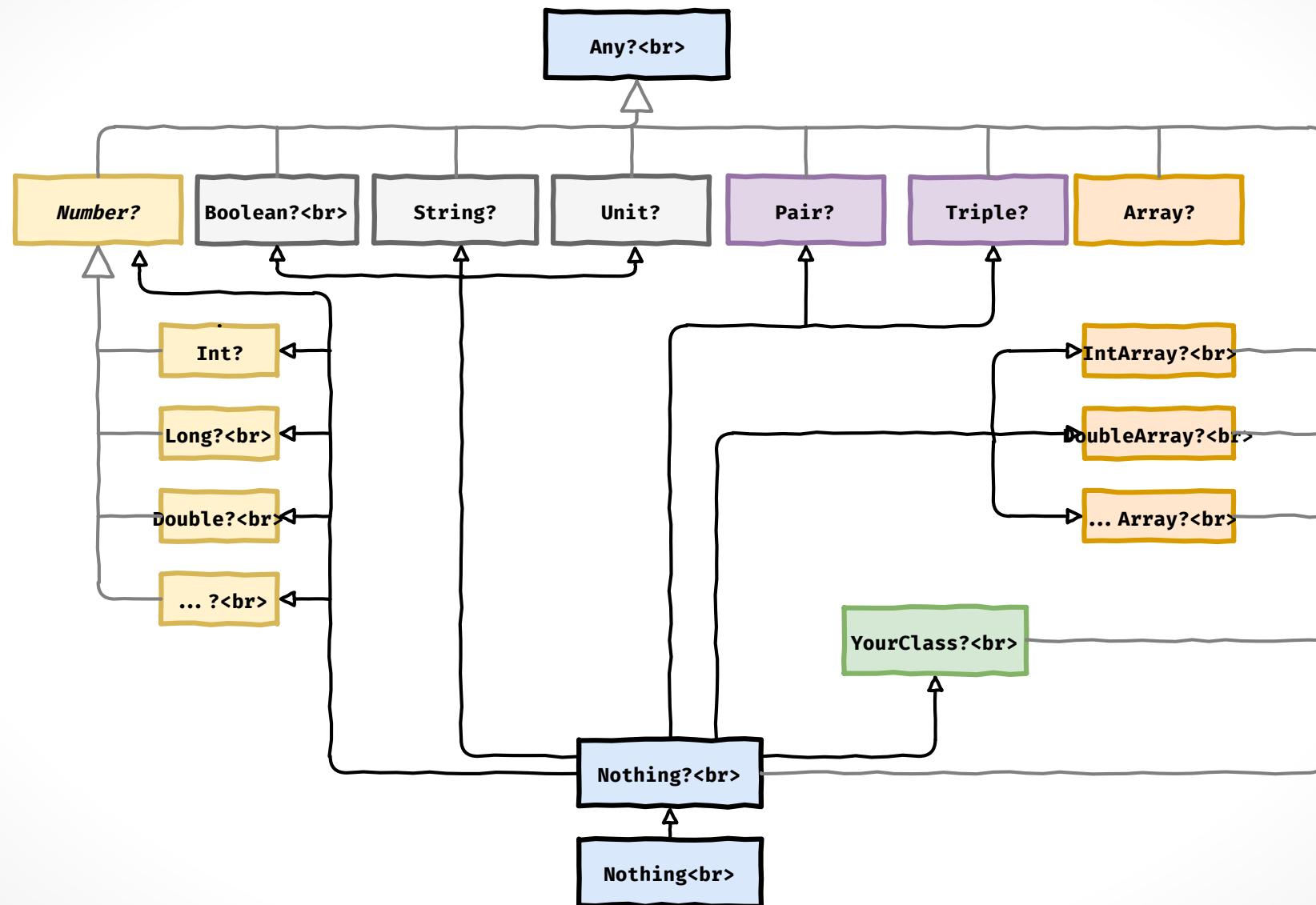
# Types basiques

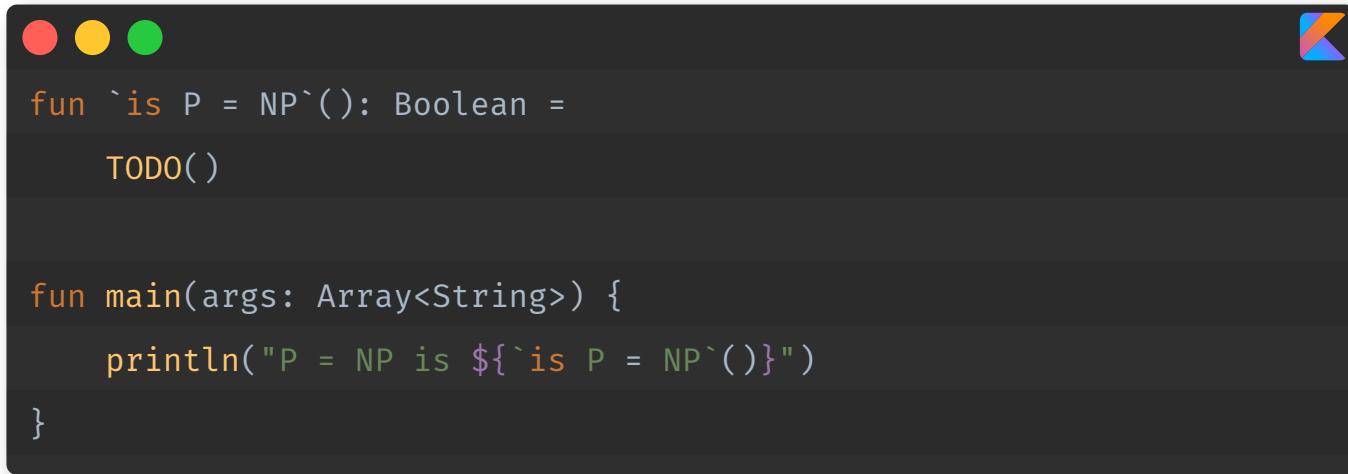
#37



# Types basiques nullable

#38





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

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

🤝 le `TODO()` est l'ami du TDD

# Les fonctions

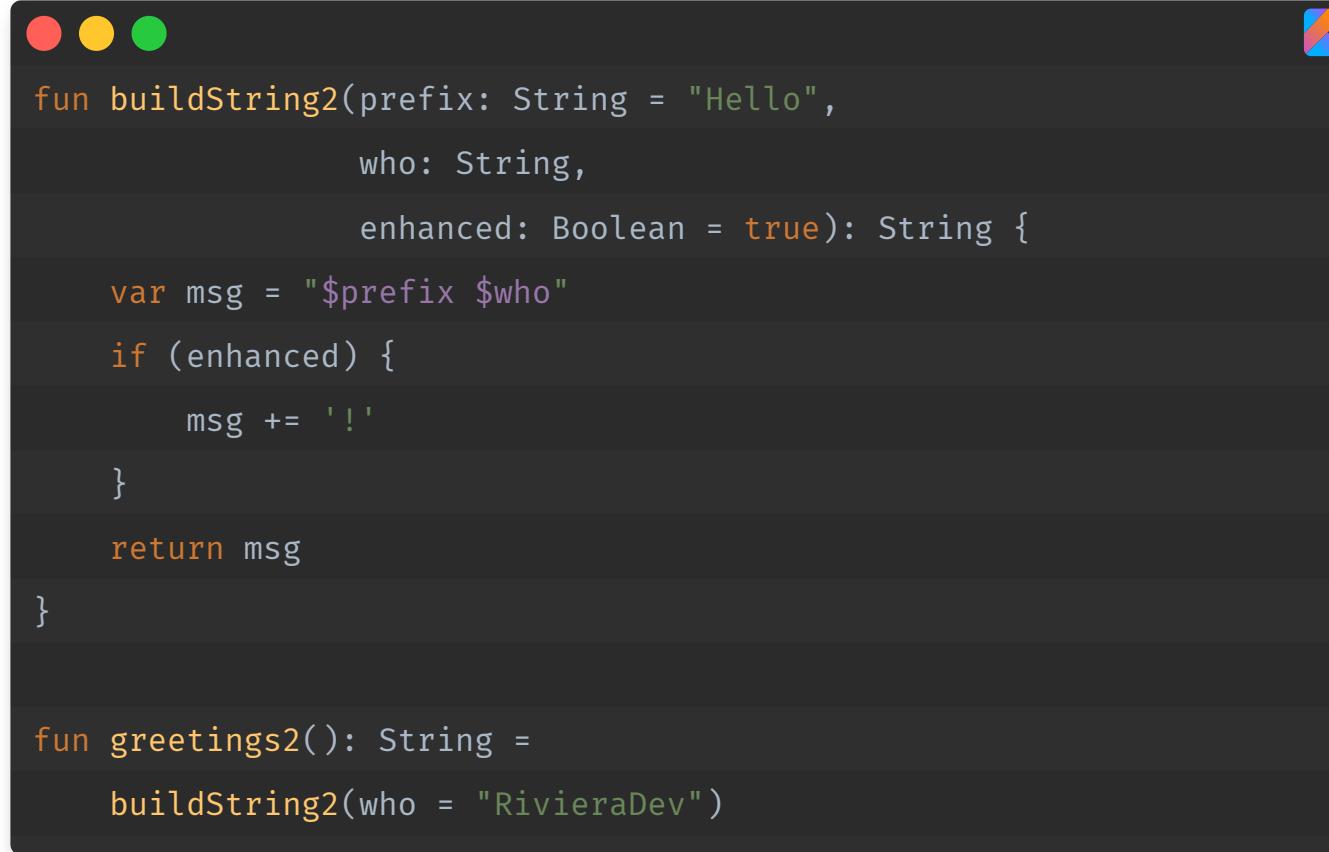
```
fun buildString(prefix: String,  
               who: String,  
               enhanced: Boolean): String {  
    var msg = "$prefix $who"  
    if (enhanced) {  
        msg += ' !'  
    }  
    return msg  
}  
  
fun greetings(): String =  
    buildString(enhanced = true, who = "RivieraDev",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

```
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\u0003\n\u0002\u0010\u000b\n\u0000\u001a"},
    d2 = {"buildString", "", "prefix", "who", "enhanced", "", "greetings"}
)
public final class NamedKt {
    @NotNull
    public static final String buildString(@NotNull String prefix, @NotNull String who, boolean enhan
        Intrinsics.checkNotNull(prefix, "prefix");
        Intrinsics.checkNotNull(who, "who");
        String msg = "" + prefix + ' ' + who;
        if (enhanced) {
            msg = msg + '!';
        }

    return msg;
}
```



```
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")
```

```
// 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
```

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

Résultat ?

TODO: bytecode myapp

TODO: bytecode lib

## ✨ Conseils

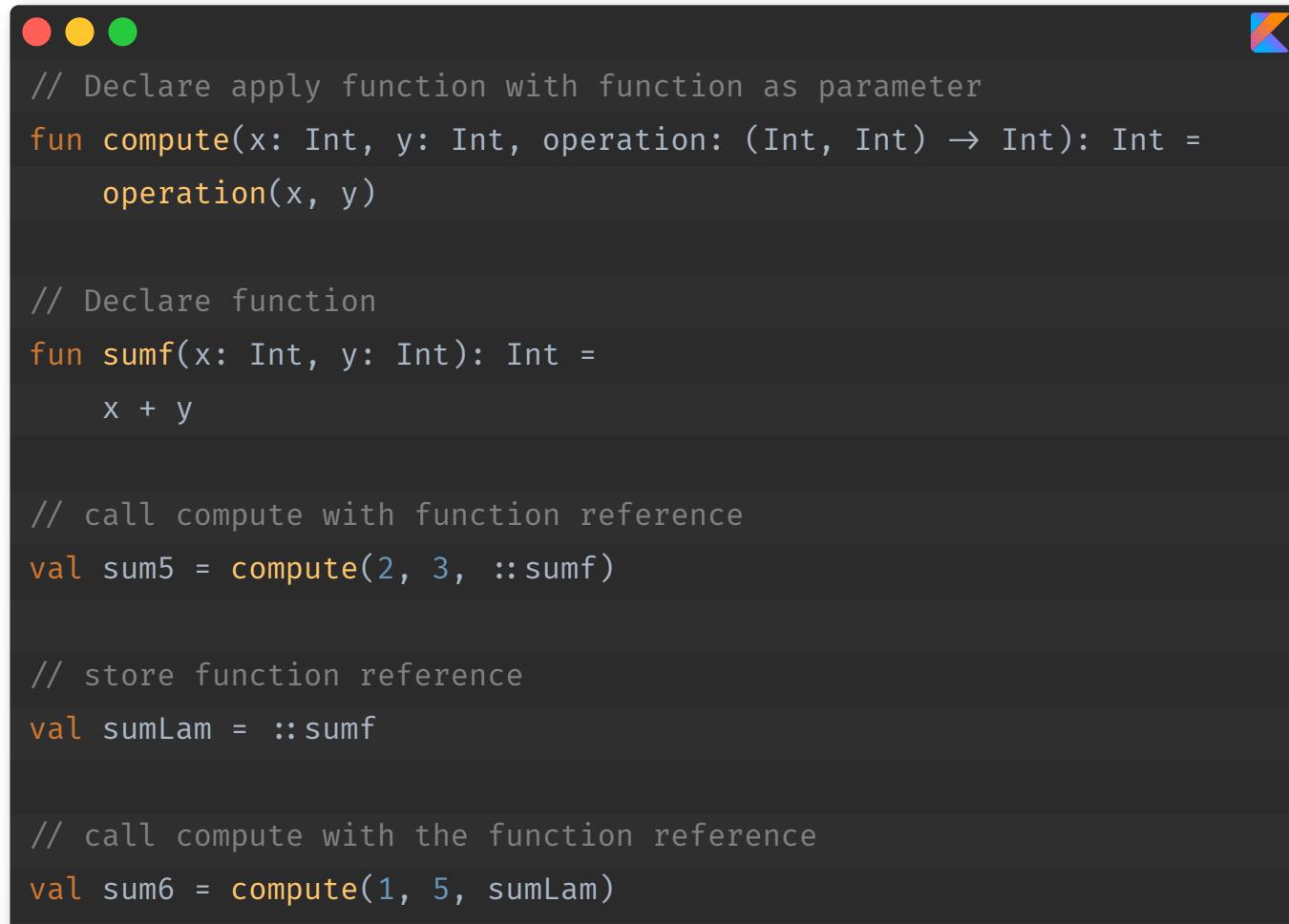
Toujours typer le retour de vos fonctions  
(sauf si c'est évident et une surcharge comme le `toString`)  
Kotlin est plus concis que Java => évitez de faire des fonctions trop longues

- Sautéz une ligne après le `=`
- Utilisez le passage des arguments par nom quand ça lève des ambiguïtés



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

# Les lambdas



```
// 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

#51

```
package _04_lambda;

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\u0001c\n\u0000\n\u0002\u0010\b\n\u0002\b\u0005\n\u0002\u0018\u0002\n\u0002\u0018\u0000",
    d2 = {"sum5", "", "getSum5", "()I", "sum6", "getSum6", "sumLam", "Lkotlin/reflect/KFunction2;", ""},
)
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.checkNotNullParameter(operation, "operation");
        return ((Number)operation.invoke(x, y)).intValue();
    }
}
```

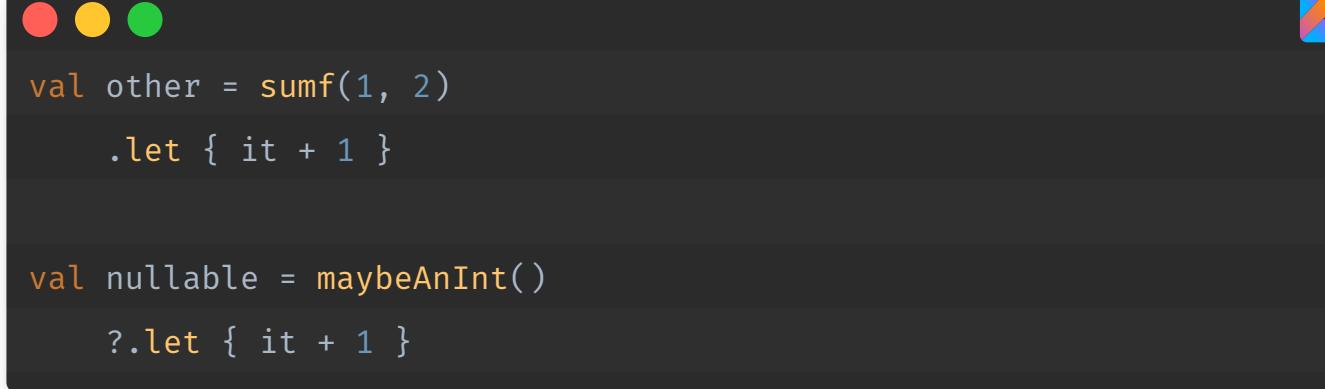
```
// Declare lambda  
  
val suml: (Int, Int) -> Int = { x: Int, y: Int ->  
    x + y  
}  
  
// call compute with suml lambda  
  
val sum3 = compute(1, 2, suml)  
  
// call compute with lamda  
  
val sum4 = compute(1, 3) { x, y ->  
    x + y  
}
```

```
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\u0005\n\u0002\u0018\u0002\n\u0002\b\u0002\",
    d2 = {"sum3", "", "getSum3", "()", "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;
    }
}
```



The screenshot shows a dark-themed code editor window with three colored circular icons (red, yellow, green) in the top-left corner and a small 'Kotlin' logo in the top-right corner. The code editor displays two examples of Kotlin code:

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

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

```
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\u001a\r\u0010\b\u001a\u0004\u0018\u00010\
    d2 = {"nullable", "", "getNullable", "()Ljava/lang/Integer;", "Ljava/lang/Integer;", "other", "ge
)
public final class LetKt {
    private static final int other;
    @Nullable
    private static final Integer nullable;

    @Nullable
    public static final Integer maybeAnInt() {
        throw (Throwable)(new NotImplementedErr((String)null, 1, (DefaultConstructorMarker)null));
    }
}
```

⚠ pas de `return`

pensez à mettre vos lambda comme dernier argument

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

➡ the tl;dr; 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

```
● ● ●
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 earth = Planet(name = "Earth")
    val moon = Moon(name = "Moon")

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



```
class SmallBody {  
    fun sizeRange(): IntRange = 0..10  
}  
  
// Inherit SmallBody  
data class Comet(val name: String)  
  
// Inherit SmallBody  
data class Asteroid(val name: String) {  
    fun sizeRange(): IntRange = 0..4  
}  
  
fun main(args: Array<String>) {  
    val bodies = listOf(Comet("Halley"), Asteroid("Adeona"))  
  
    // bodies.forEach { body ->  
    //     println("$body: ${body.sizeRange()}" )  
    // }  
}
```

⚠ 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



```
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 class JsonValue

data class JsonObject(val attributes: Map<String, JsonValue>) : JsonValue()

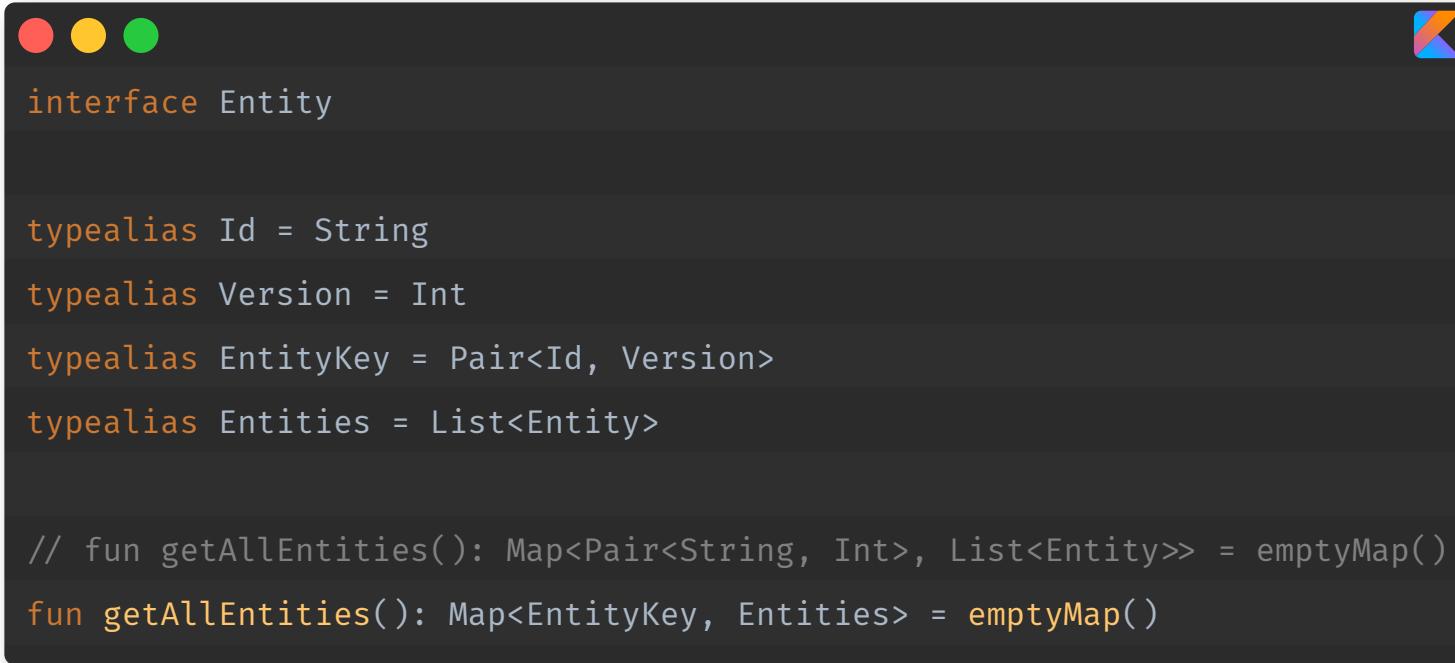
data class JSONArray(val values: List<JsonValue>) : JsonValue()

data class JsonString(val value: String) : JsonValue()

data class JsonNumber(val value: Number) : JsonValue()

data class JsonBoolean(val value: Boolean) : JsonValue()

object JsonNull : JsonValue()
```



The screenshot shows a dark-themed code editor window. At the top left are three circular icons: red, yellow, and green. At the top right is the official Kotlin logo. The code itself is as follows:

```
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>> = emptyMap()
fun getAllEntities(): Map<EntityKey, Entities> = emptyMap()
```

```
Compiled from "typealias.kt"
public final class _06_class_2.TypealiasKt {
    public static final java.util.Map<kotlin.Pair<java.lang.String, java.lang.Integer>, java.util.List<
        Code:
            0: invokestatic #12                  // Method kotlin/collections/MapsKt.emptyMap:()Ljava/util/
            3: areturn
}
```

😊 `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

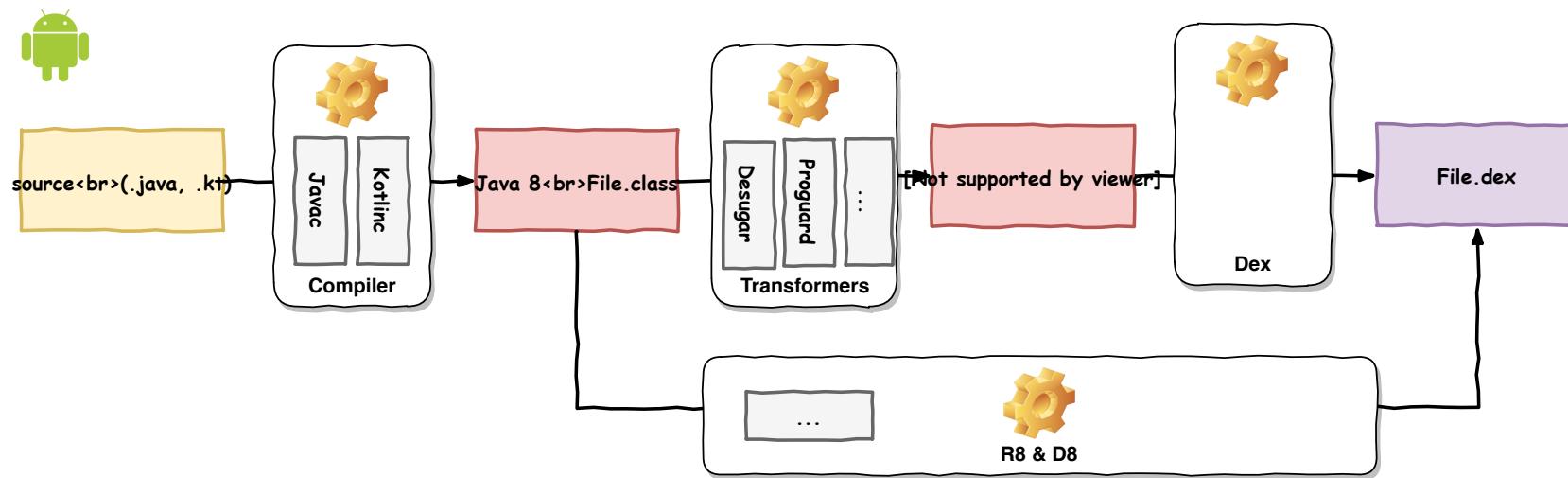
🤓 `sealed` permet de faire des types algébriques de données  
(Algebraic Data Type)

# Pause

# ByteCode Android

# Compilation pour Android

#68



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



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



000000000	64	65	78	0a	30	33	35	00	06	50	f0	61	50	10	7c	c0	dex.035..P.aP. ..
00000010	b2	f9	77	2d	54	df	60	f3	ac	dd	b0	10	eb	55	53	e1	..w-T.`.....US.
00000020	28	f6	10	00	70	00	00	00	78	56	34	12	00	00	00	00	(...p...xV4....
00000030	00	00	00	00	4c	f5	10	00	12	17	00	00	70	00	00	00	.....L.....p...
00000040	31	03	00	00	b8	5c	00	00	2a	08	00	00	7c	69	00	00	1....\..*... i..
00000050	30	03	00	00	74	cb	00	00	20	1a	00	00	f4	e4	00	00	0...t... .....
00000060	75	02	00	00	f4	b5	01	00	94	f1	0e	00	94	04	02	00	u..... .....
00000070	4a	f1	07	00	4c	f1	07	00	75	f1	07	00	a4	f1	07	00	J...L...u.....
00000080	c2	f1	07	00	e0	f1	07	00	00	f2	07	00	23	f2	07	00	.....#...
00000090	71	f2	07	00	b9	f2	07	00	07	f3	07	00	37	f3	07	00	q.....7...
000000a0	69	f3	07	00	90	f3	07	00	bc	f3	07	00	e3	f3	07	00	i.....
000000b0	27	f4	07	00	71	f4	07	00	8f	f4	07	00	ad	f4	07	00	'...q.....
000000c0	cb	f4	07	00	ed	f4	07	00	0f	f5	07	00	2c	f5	07	00	....., ...
000000d0	49	f5	07	00	79	f5	07	00	b1	f5	07	00	e9	f5	07	00	I...y.....
000000e0	1a	f6	07	00	51	f6	07	00	7b	f6	07	00	a6	f6	07	00	....Q...{ .....
000000f0	d1	f6	07	00	0a	f7	07	00	47	f7	07	00	84	f7	07	00	.....G.....
00000100	c1	f7	07	00	02	f8	07	00	43	f8	07	00	84	f8	07	00	.....C.....

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

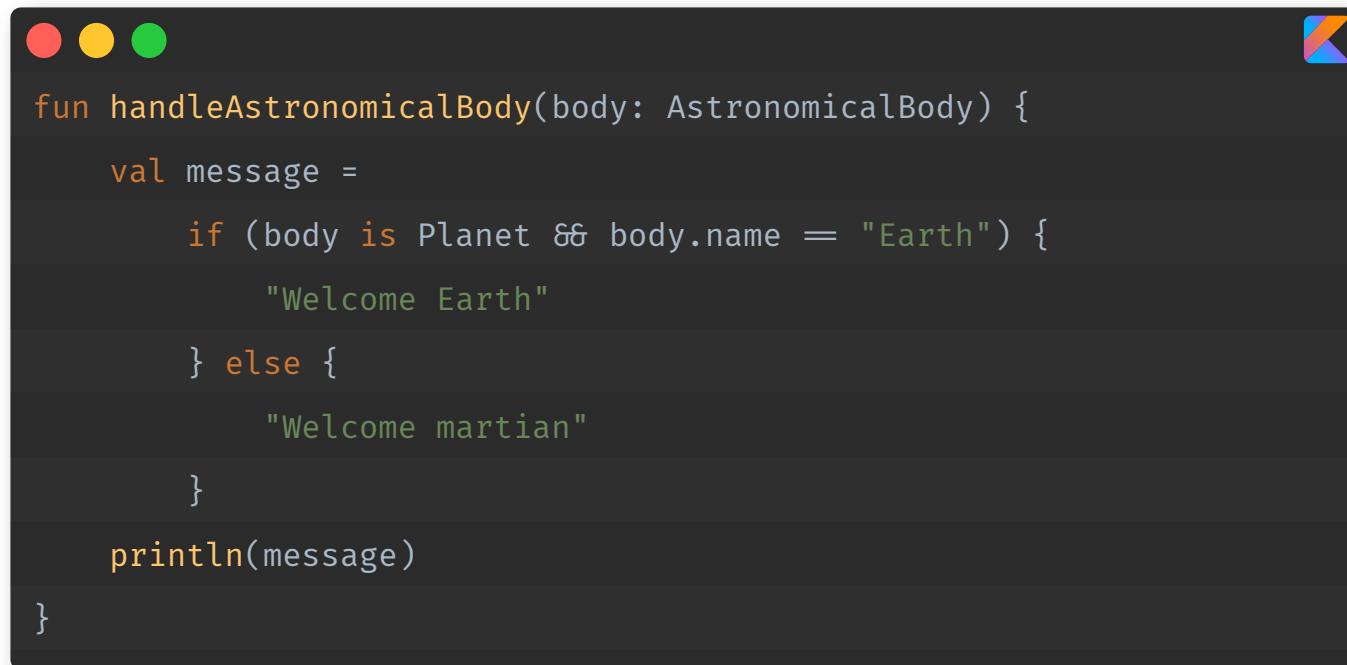
```
Processing './target/android/dex/classes.dex'...
Opened './target/android/dex/classes.dex', DEX version '035'
Class #0
  Class descriptor : 'L_00_helloworld/HelloWorldKt;'
  Access flags     : 0x0011 (PUBLIC FINAL)
  Superclass       : 'Ljava/lang/Object;'
  Interfaces       : -
  Static fields    : -
  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
```

```
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"

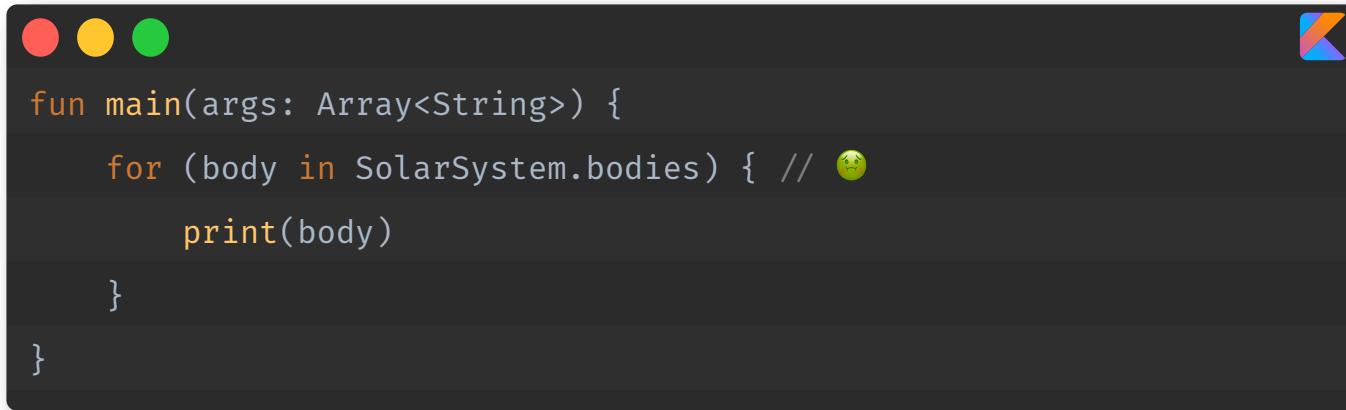
.annotation system Ldalvik/annotation/SourceDebugExtension;
    value = "SMAP\nHelloWorld.kt\nKotlin\n*S Kotlin\n*F\n+ 1 HelloWorld.kt\n_00_helloworld/HelloWorldK
.end annotation
.annotation runtime Lkotlin/Metadata;
    bv = {
        1,
        0,
        2
    }
    d1 = {
        "\u0000\u0012\n\u0000\u0002\u0010\u0002\u0000\u0000\n\u0002\u0010\u0011\n\u0002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

# Autres structures



A screenshot of a dark-themed code editor window. The title bar at the top has three colored circles (red, yellow, green) on the left and the Kotlin logo on the right. The main area contains the following Kotlin code:

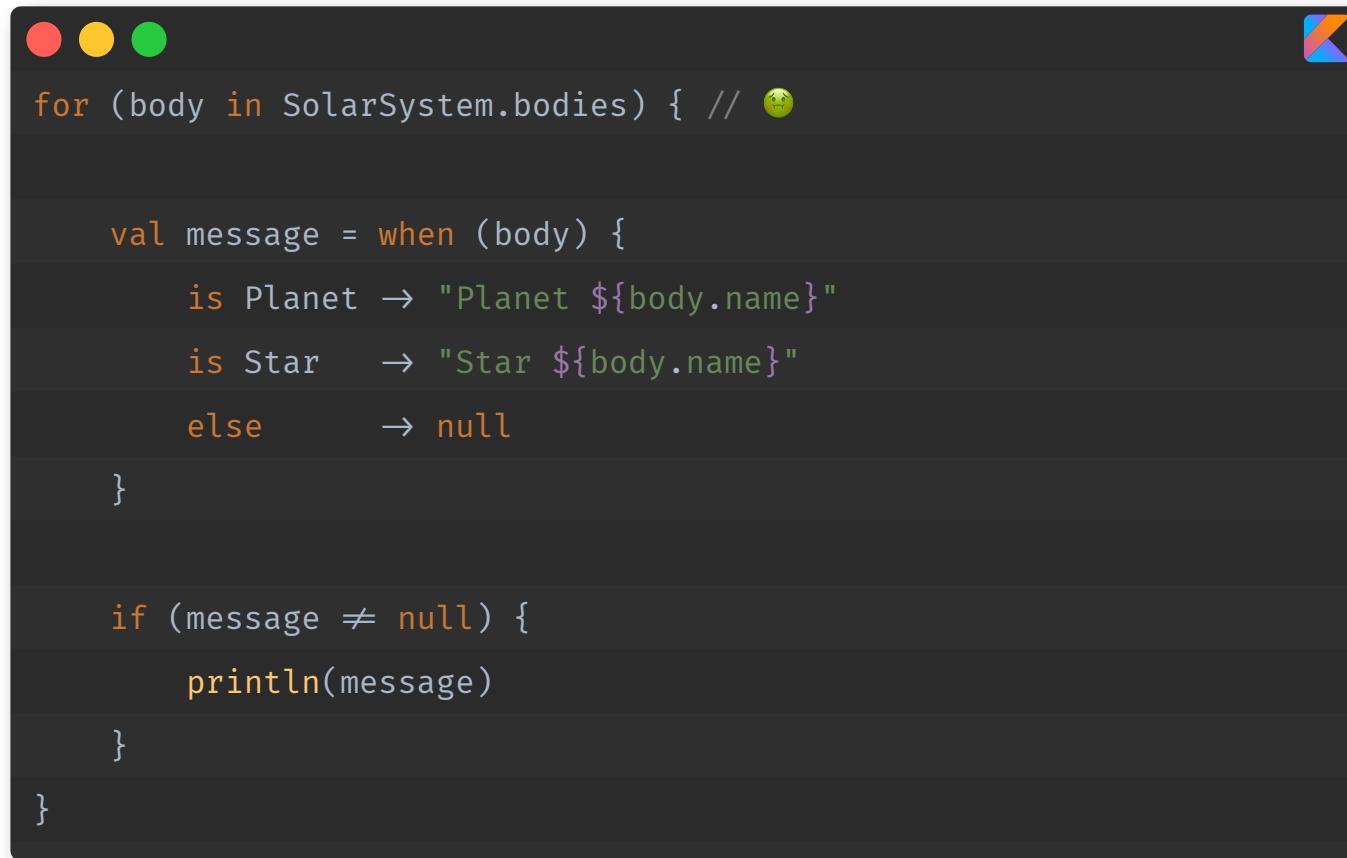
```
fun handleAstronomicalBody(body: AstronomicalBody) {
    val message =
        if (body is Planet && body.name == "Earth") {
            "Welcome Earth"
        } else {
            "Welcome martian"
        }
    println(message)
}
```



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

Mais aussi des `while` et `do while`

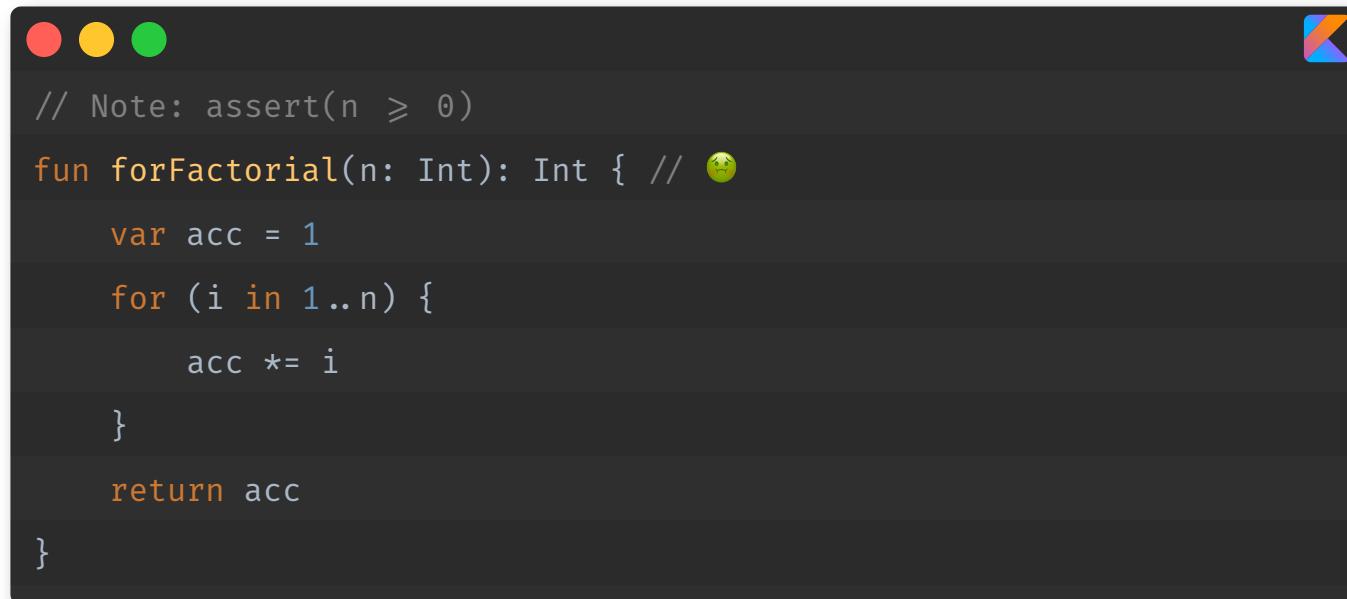
et des `break`, `continue`, et `label`



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

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

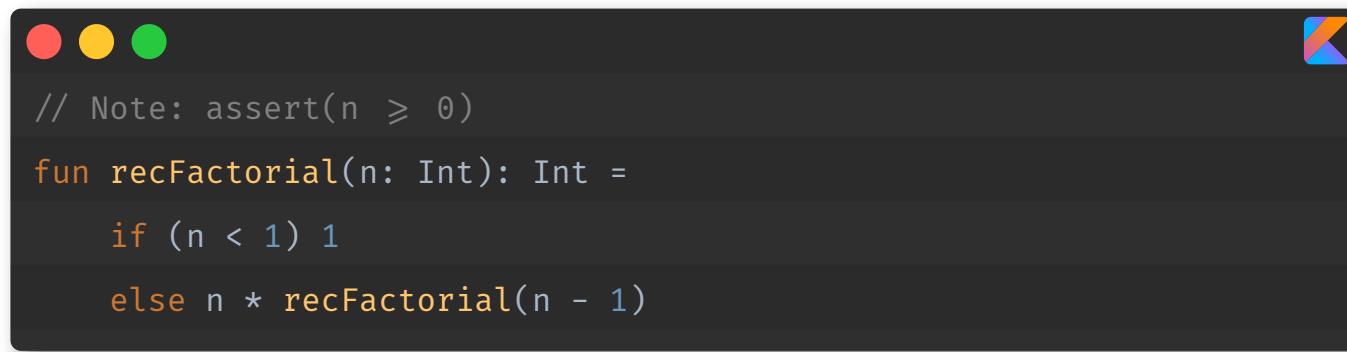
    if (message != null) {
        println(message)
    }
}
```



```
// Note: assert(n >= 0)

fun forFactorial(n: Int): Int { // 🤔
    var acc = 1
    for (i in 1..n) {
        acc *= i
    }
    return acc
}
```

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



A screenshot of a macOS terminal window. The window has a dark theme with red, yellow, and green close buttons. In the top right corner is the official Kotlin logo. The code inside the terminal is:

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

# ByteCode factoriel avec récursivité

#79

```
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 recFactorial:(I)I
            16: imul
            17: ireturn
}
```

$$x! = x \times (x - 1) \times \dots \times 2 \times 1$$

$$1! = 0! = 1$$

$fact(x)$

$x \times fact(x - 1)$

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

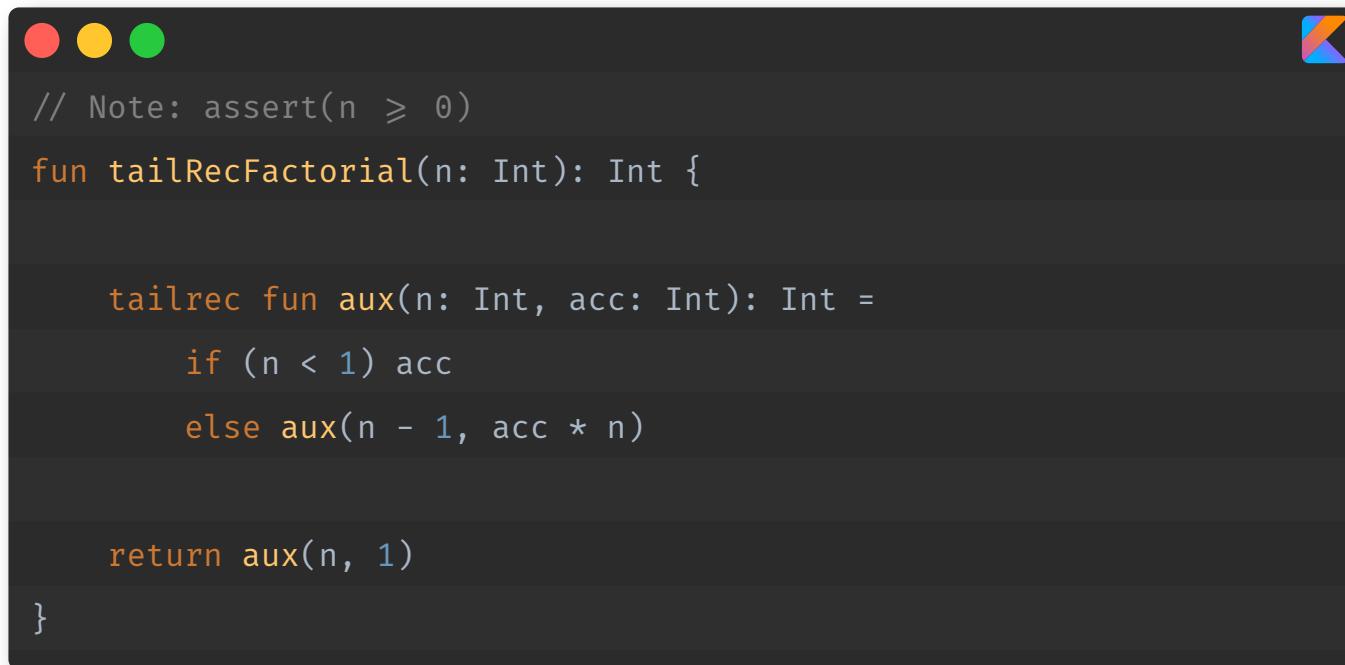
$x \times (x - 1) \times (x - 2) \times \dots$

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



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

⚠ Nécessite une optimisation par le compilateur



The screenshot shows a dark-themed code editor window. In the top right corner, there are three colored circular icons (red, yellow, green) and the Kotlin logo. The code itself is as follows:

```
// 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)
}
```

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

# ByteCode factoriel avec recursivité terminale

#84

2/2

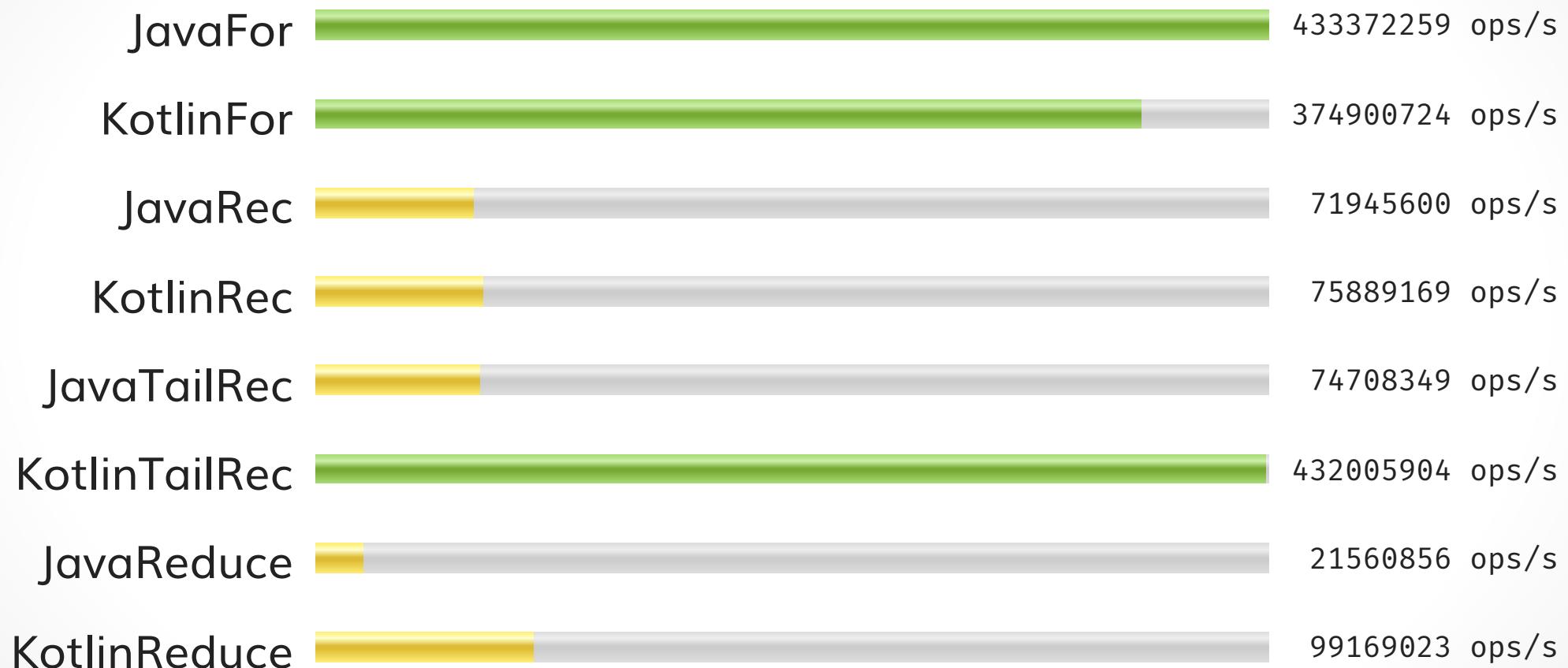
```
Compiled from "tailrec-factorial.kt"
final class _09_structures.recusion.Tailrec_factorialKt$tailRecFactorial$1 extends kotlin.jvm.internal
    public static final _09_structures.recusion.Tailrec_factorialKt$tailRecFactorial$1 INSTANCE;

    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: iconst_1
            6: goto       25
```

# Performances sur 10!

#85



`when` peut être utilisé avec

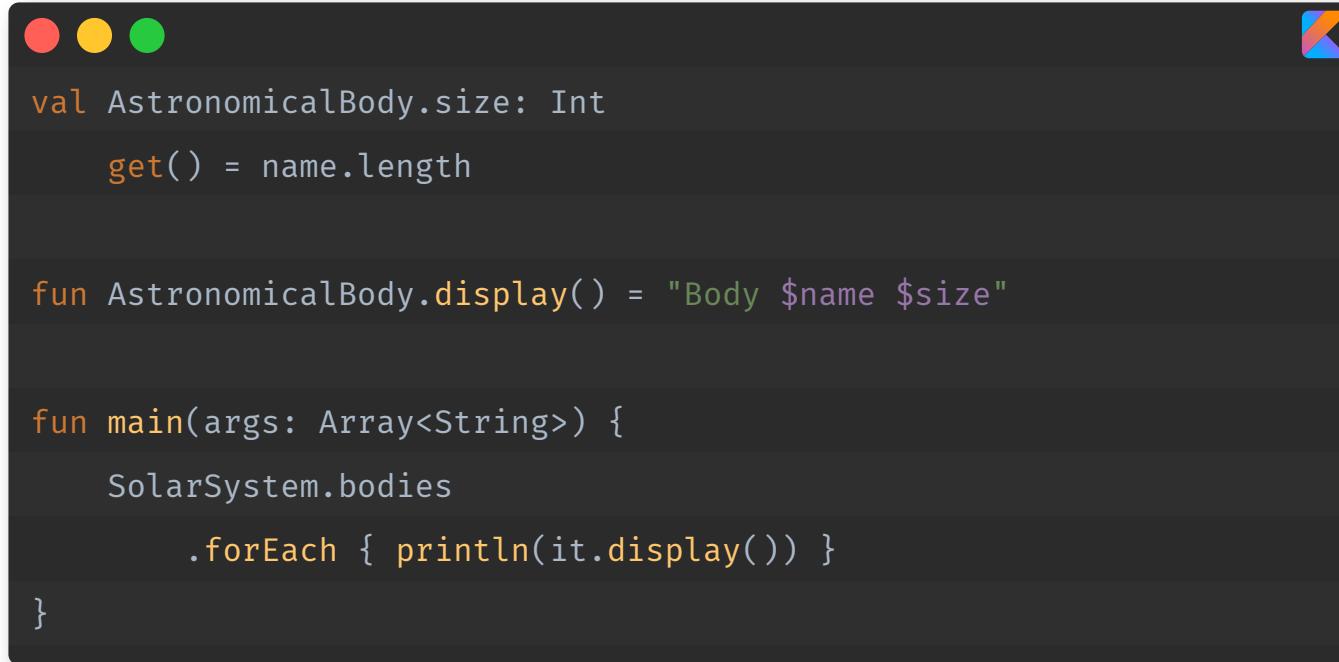
- des constantes
- plusieurs valeurs séparées par `,`
- une expression
- avec `is` et un type (avec un 'smart cast')

## ✨ Tips

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

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

# Extensions de fonctions



The screenshot shows a dark-themed code editor window. In the top right corner, there are three colored circular icons (red, yellow, green) and the Kotlin logo. The code itself is as follows:

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

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

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

```
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\n\u0002\b\u0003\n\u0002\u0010\u0000", "size", "", "Lastronomy/AstronomicalBody;", "getSize", "(Lastronomy/AstronomicalBody;)"},
    d2 = {"size", "", "Lastronomy/AstronomicalBody;"},
    k_4 = 1
)
public final class ExtensionKt {
    public static final int getSize(@NotNull AstronomicalBody $receiver) {
        Intrinsics.checkNotNullParameter($receiver, "$receiver");
        return $receiver.getName().length();
    }

    @NotNull
    public static final String display(@NotNull AstronomicalBody $receiver) {
```

Permet d'enrichir les APIs Java

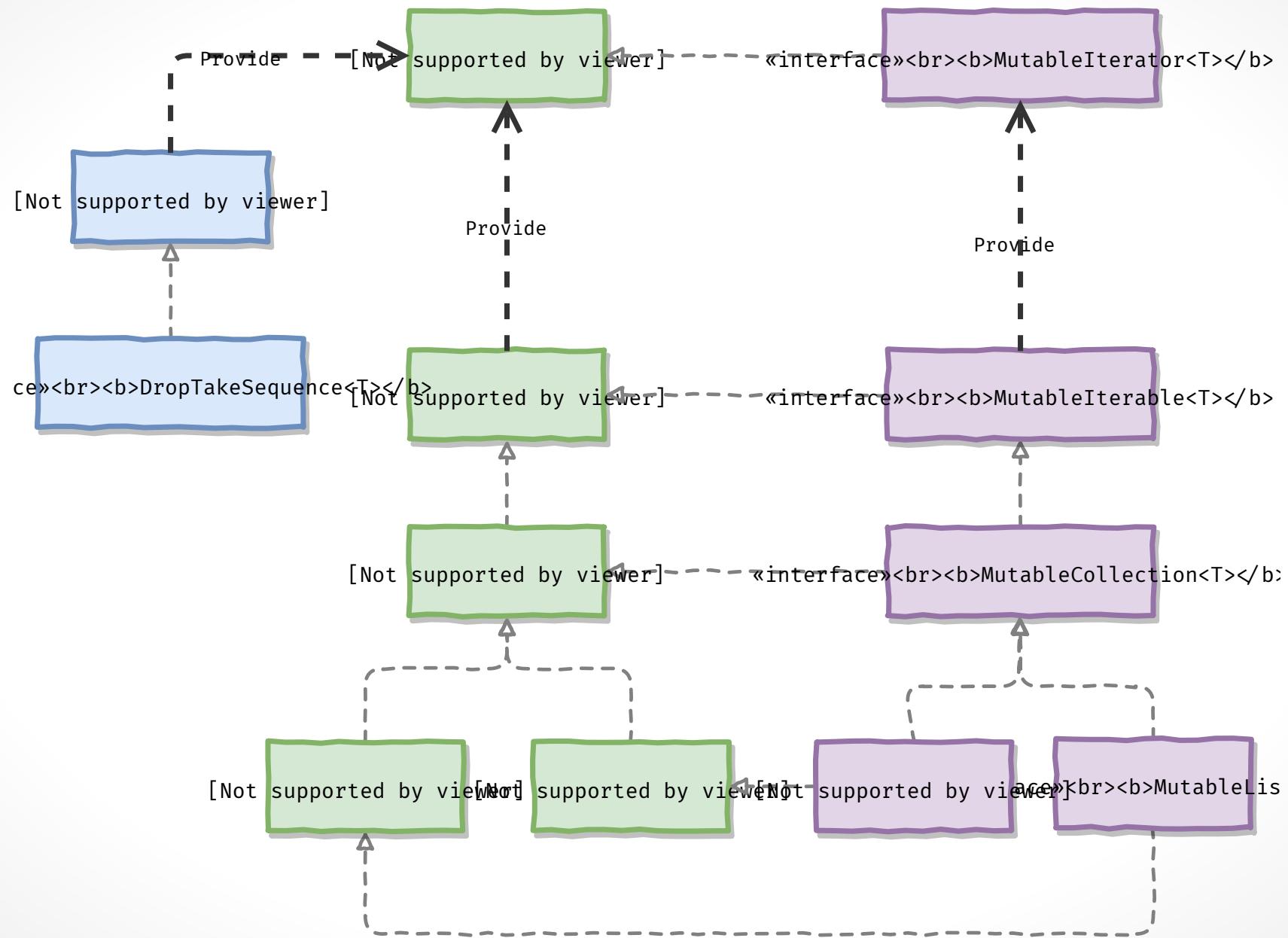
- ➡ **Android KTX** <<https://developer.android.com/kotlin/ktx>>
- ➡ **Spring** <<https://docs.spring.io/spring/docs/current/spring-framework-reference/languages.html#kotlin-spring-projects-in-kotlin>>
- ➡ **RxKotlin** <<https://github.com/ReactiveX/RxKotlin>>
- ...

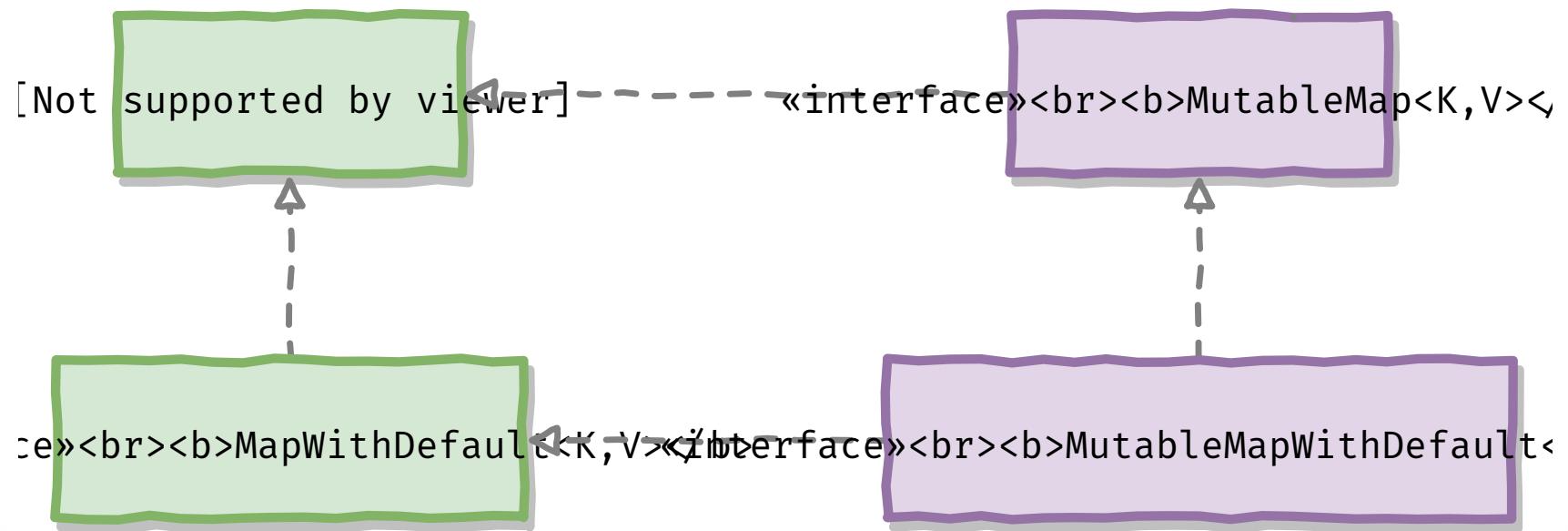
Permet la *SoC* (Separation of Concerns)

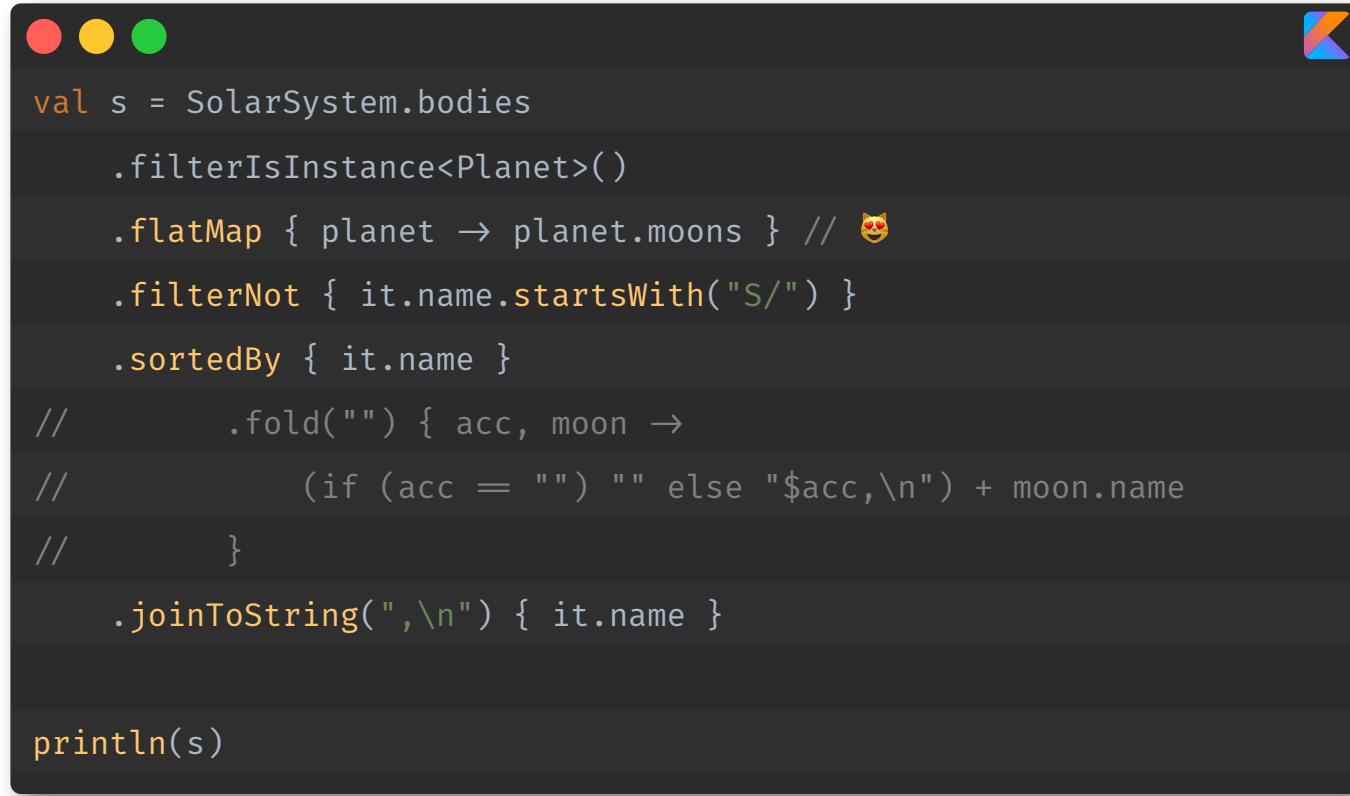
# Les collections

# Collections

#92







```
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)
```

```
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)
}
```

```
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

#97

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

ApiClassic  44535 ops/s

ApiSequence  23652 ops/s



```
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

#99

Benchmark	Mode	Cnt	Score	Error	Units
ApiClassicFirst	thrpt	200	241752.062	5022.663	ops/s
ApiSequenceFirst	thrpt	200	3615451.391	454502.198	ops/s
ApiClassicFirst				241752	ops/s
ApiSequenceFirst				3615451	ops/s

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

😅 Mais ça reste des collections Java

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

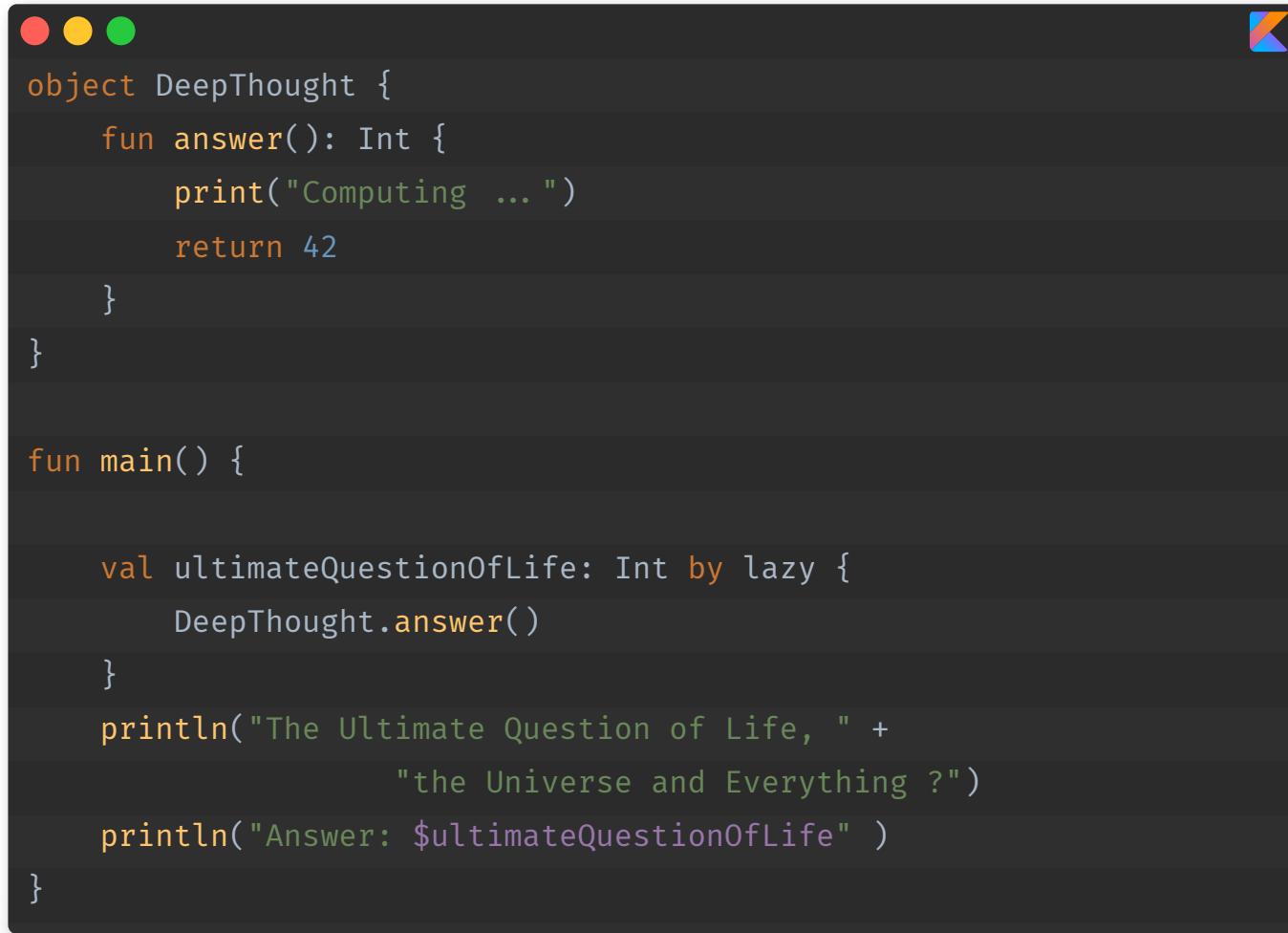
📏 Avant d'utiliser les `Sequence`, faites des mesures

# Les delegates

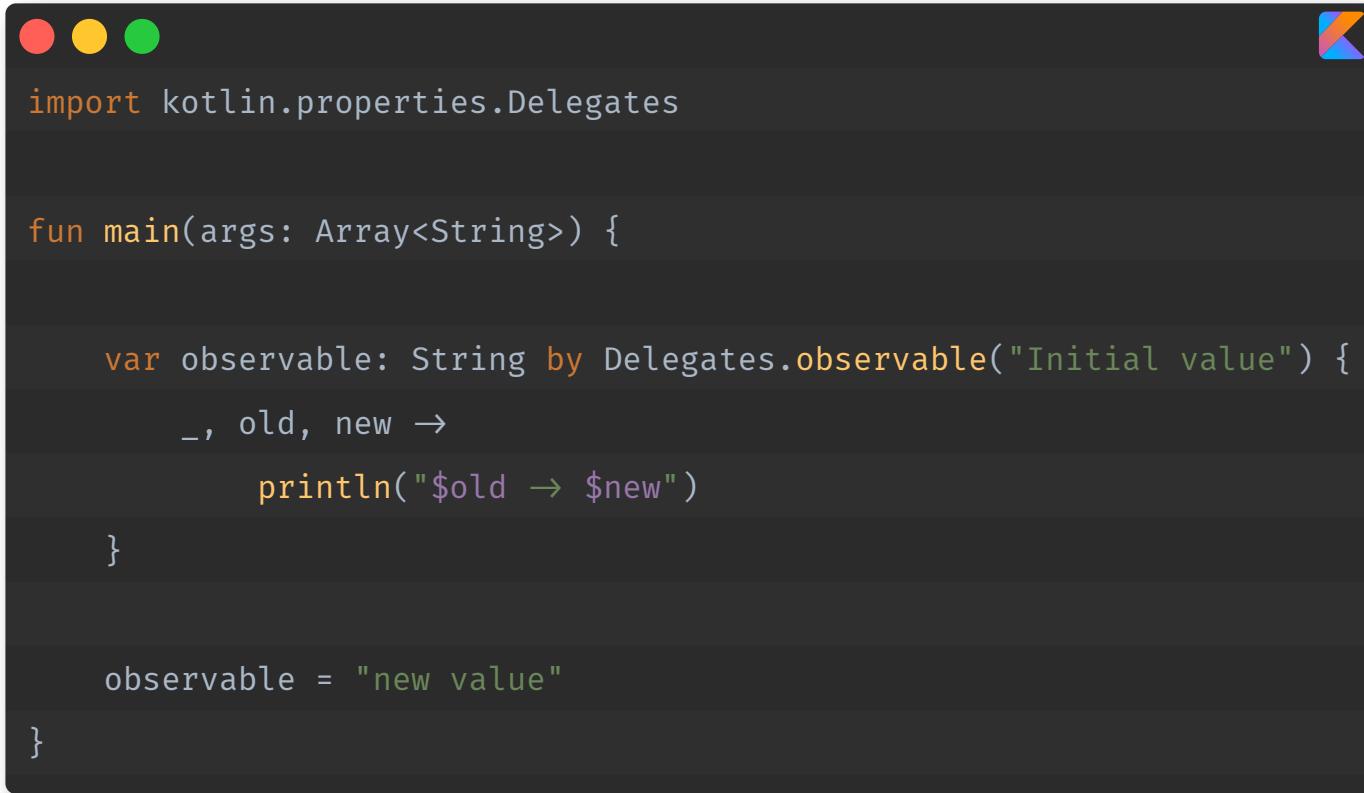
```
import kotlin.properties.ReadOnlyProperty
import kotlin.reflect.KProperty

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 RivieraDev"
}
```



```
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" )  
}
```

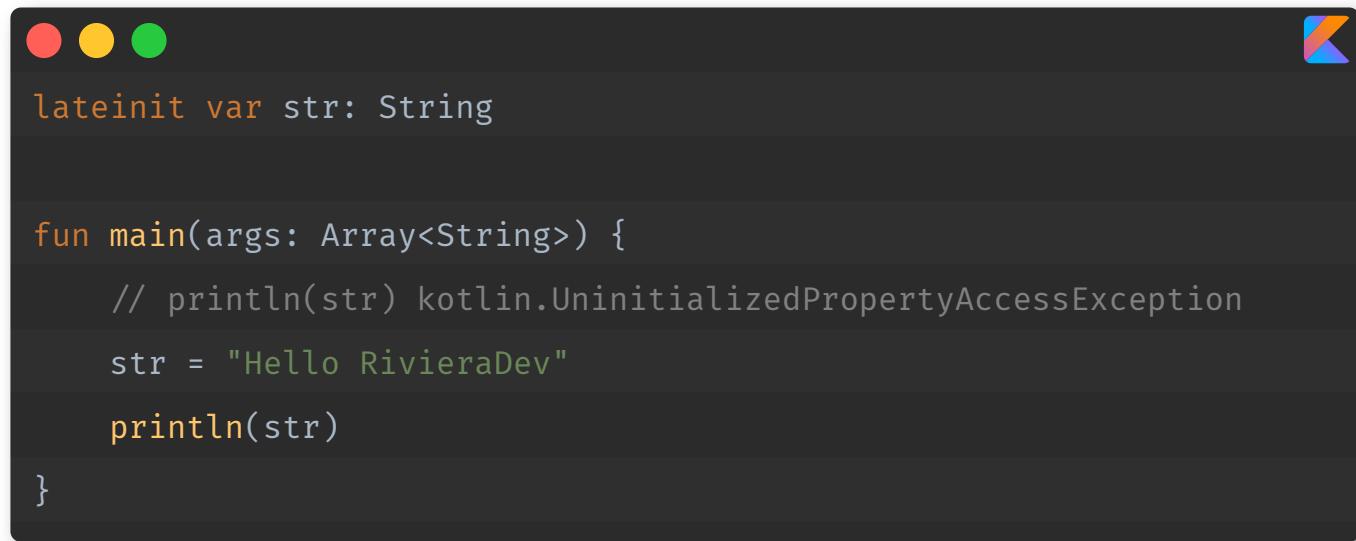


```
import kotlin.properties.Delegates

fun main(args: Array<String>) {

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

    observable = "new value"
}
```



```
lateinit var str: String

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

`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

# Un peu plus sur les fonctions

```
import java.time.Instant

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

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

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

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

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

```
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\u0000\n\u0002\u0010\u000e\n\u0002\b\u0000",
    d2 = {"L_13_advanced_function/Logger;", "", "name", "", "(Ljava/lang/String;)V", "level", "L_13_a
)
public final class Logger {
    private final Logger.Level level;
    private final String name;

    public final void info(@NotNull Function0 message) {
        Intrinsics.checkNotNull(message, "message");
        Logger.Level lvl$iv = Logger.Level.INFO;
        if (access$getLevel$p(this).compareTo((Enum)lvl$iv) >= 0) {
            String var4 = '[' + access$getLevel$p(this).name() + "] " + access$getName$p(this) + " - "
            System.out.println(var4);
        }
    }
}
```

```
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>()
```

```
package _13_advanced_function;

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\u0002\u0010\u0011\n\u0002\u0010\u0000", "main", "", "args", "", "", "([Ljava/lang/String;)V"},
    d2 = {}
)
public final class ReifiedKt {
    public static final void main(@NotNull String[] args) {
        Intrinsics.checkNotNull(args, "args");
        Pojo p1 = new Pojo();
        p1.setName("Plop1");
        System.out.println(p1);
        JavaBeanBuilder this$_iv = JavaBeanBuilder.INSTANCE;
        Pojo p2 = (Pojo)this$_iv.createBean(Pojo.class);
        p2.setName("Plop2");
        System.out.println(p2);
    }
}
```

## Cas d'utilisation du **reified**

Pour créer des extensions Kotlin des fonctions Java qui utilisent des  
**Class<T>**

## Cas d'utilisation du **inline**, **noinline**, **crossinline**

Quand on utilise **reified**

Quand on sait ce qu'on fait, ➔

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

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

# 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>>

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/>

...



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/>

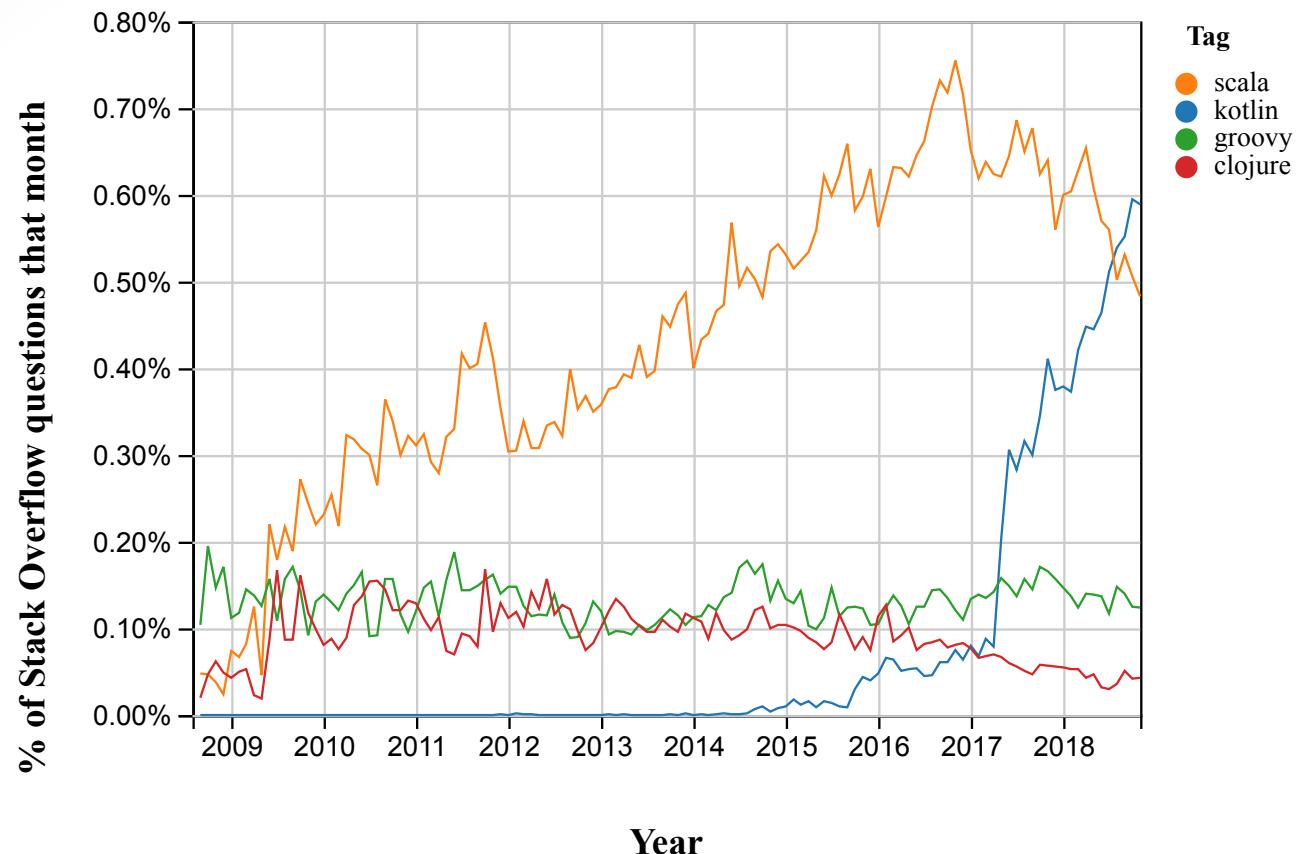
## Natif

Faire des applications sans JVM

Partager du code avec iOS

WebAssembly

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



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

C'est déjà mature

- 👉 Code plus expressif, plus sûr, plus simple
- 🤝 Interopérable avec Java
- 👍 Outilage (éditeur, gradle, maven)
- 👍 Ecosystème et communauté
- 🚀 Évolution rapide
- 孵 Code multiplatform

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

## Questions ?