Kotlin
vs
Modern Java

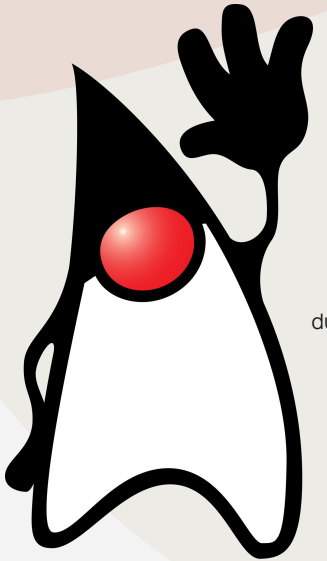
A biased presentation by Alexander Wu

TABLE OF CONTENTS

01

Java

public static void main (String[] args)



duke

Why Java?

Write once, run everywhere

No need to compile to platform native machine code with the JVM

Automatic garbage collection

No need to manage pointers. All managed by the JVM. Segfault-less!

Ease to learn

Good documentation with a friendlier approach to OOP

Where?

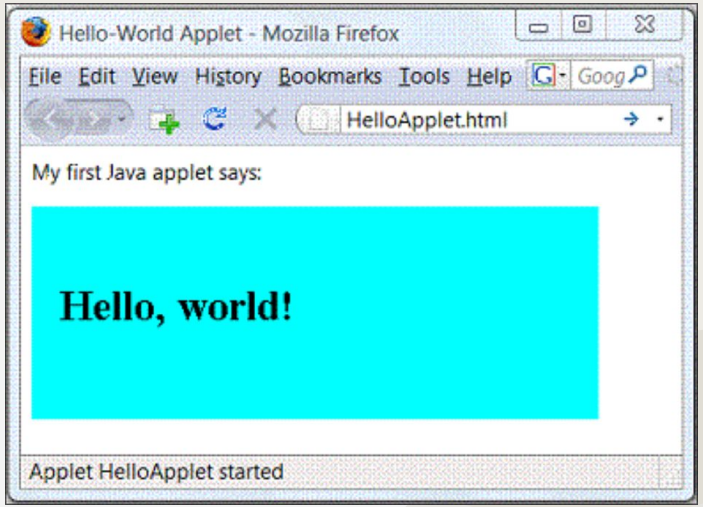
Java in the wild

Applets

Java can get embedded into the browser

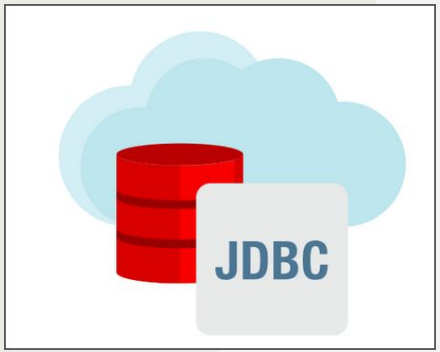
- Deprecated in Java 9
- Performance and security issues

(use JS instead)



Enterprise Servers

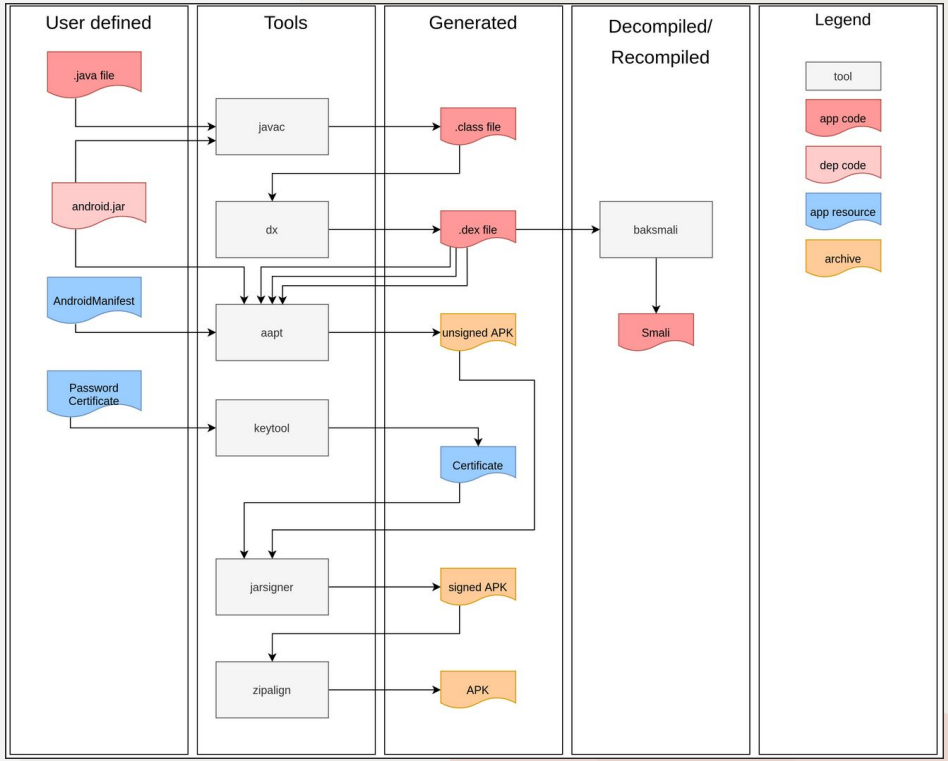
- Stability and reliability
- Massive ecosystem
- Integration



Android

Adopted by google to be the official language for Android

Oracle sued Google for this.



Live Coding!

Create a Java class Person that has an age, name, and a list of children.
The constructor should take age, name and number of children. With the number of children, every child should be initialized to a random age from 1 to age - 1 with a name of parent name_childNumber
Create getters for name and age.
Also create a function called printChildren that takes in an interface PersonFilter that also has to be created with a boolean method filter. printChildren should call every child manually with a for loop

02

Modernization

void main()

01

Java 8

Functional Programming?

Lambda Expressions
Method reference

Optional
Supplier Consumer Interface
Streams
...

02

Java 9 to 11

Performance Bumps

Local-variable type inference (var)

Modules
New Garbage Collector
Private interface methods
New string methods
...

02 Java 11 to 17

QOL

Records
Switch statements
Multiline strings “””
instanceOf matching
Improved nullPointerException
Sealed classes
...

03 Java 18 to 21

public static void main(String[] args) is dead

Record Classes
New main entry point
Unnamed classes (preview)

Unnamed variables
Switch pattern matching with instanceOf
...

04 Java 22 to 25

Nothing Notable

Flexible Constructor Bodies
String templates

Primitive types in patterns
Better module imports

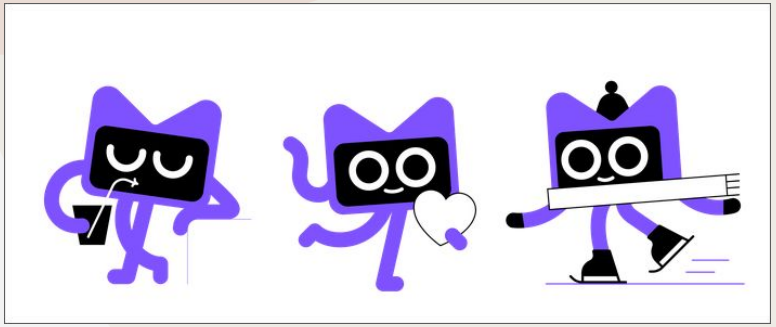
...

03

Kotlin

fun main()

Wow so fun!!



Live Coding! (again)

Rewrite everything in kotlin using data classes and higher order functions

Immutability

Everything is side effect-less

› Dafny?? (contracts)

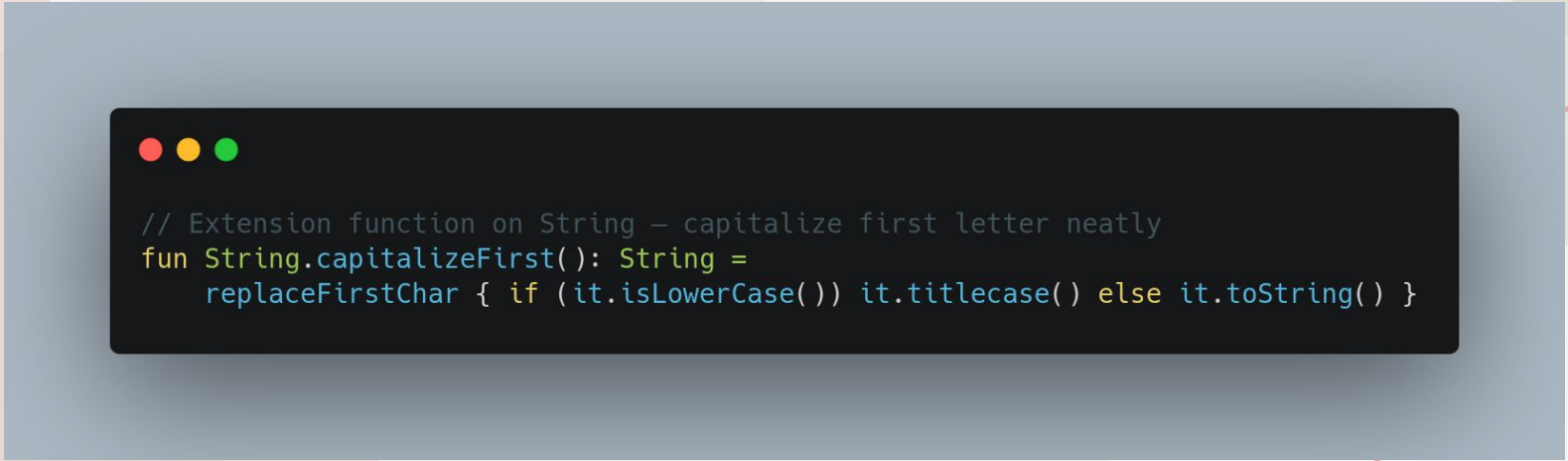
```
// Define a function that tells the compiler:
// "If this function returns true, the input 'value' is guaranteed to be non-null."
fun <T> isNotNull(value: T?): Boolean {
    contract {
        returns(true) implies (value != null)
    }
    return value != null
}

// Another contract example: ensures a lambda is called exactly once
inline fun callOnce(action: () -> Unit) {
    contract {
        callsInPlace(action, InvocationKind.EXACTLY_ONCE)
    }
    action()
}
```

Extension functions

Easier method chaining

Also allows for scope functions (let, run, with, apply, also)



```
// Extension function on String — capitalize first letter neatly
fun String.capitalizeFirst(): String =
    replaceFirstChar { if (it.isLowerCase()) it.titlecase() else it.toString() }
```

Null safety

Elvis.

```
val result = input
     ?.trim()
     ?.takeIf { it.isNotEmpty() }
     ?.uppercase()
     ?: "default"
         .takeIf { it.length > 3 }
         ?.reversed()
     ?: "EMPTY"
```

Everything is an expression

No more ternaries

```
// Everything below is *one big expression chain*
val result = run {
    println("🔍 Processing input: $input")

    // if is an expression
    if (input != null) {
        try {
            // try is also an expression
            val number = input.toInt()

            when { // when is an expression
                number < 0 -> "Negative number: $number"
                number in 0..50 -> run { // run is an expression
                    println("Number in lower range")
                    "Small positive: $number"
                }
                else -> run {
                    println("Number is large!")
                    "Large number: $number"
                }
            }
        } catch (e: NumberFormatException) {
            // The catch block itself *returns* a value
            "❌ Invalid number: ${e.message}"
        }
    } else {
        "⚠️ Input was null"
    }
}
```

Operator Overloading

list[index]
no more Arraylist.get(index)

1 Liner Leetcode

https://leetcode.com/problems/merge-intervals/

04

The conversion process

Gradle? gradle.kts!

Interoperability

Calling Java from Kotlin

- getters/setters gets converted
- @NotNull
- @Nullable

Same is available backwards!

```
plugins {
    id "org.jetbrains.kotlin.jvm" version "2.2.21"
}
```

—Some Kotlin Dev.

Questions?