Evolutionary Architecture: Evolving even the language

Luram Archanjo

Who am I?

- Software Engineer at Sensedia
- MBA in Java projects
- Java and Microservice enthusiastic

Agenda

- Use case
- Microservices
- Evolutionary Architecture
- Challenges to adopt Kotlin
- Kotlin benefits
- Questions

Use case

Use case

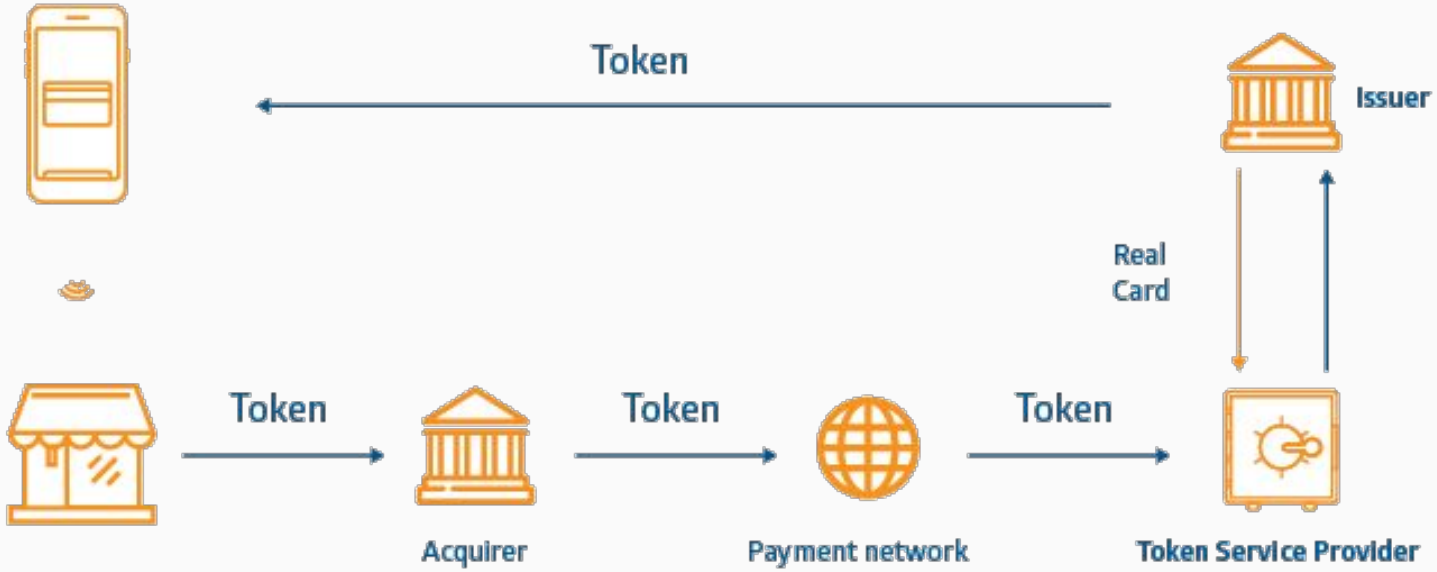
Token Service Provider

Tokenization reduces the value of stored payment credentials by removing consumers' **primary account numbers (PANs)** from the transaction process. It does this by **replacing them** with a unique identifier called a **payment token**.

This reduces the appeal of stealing the credentials as they would be largely useless to hackers.

Source: https://www.rambus.com/blogs/what-is-a-token-service-provider/

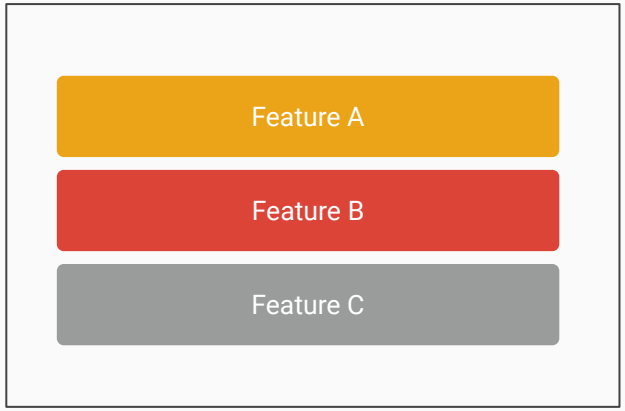
Use case


Token
Issuer
Real
Card
Token
Token
Token
Acquirer
Payment network
Token Service Provider

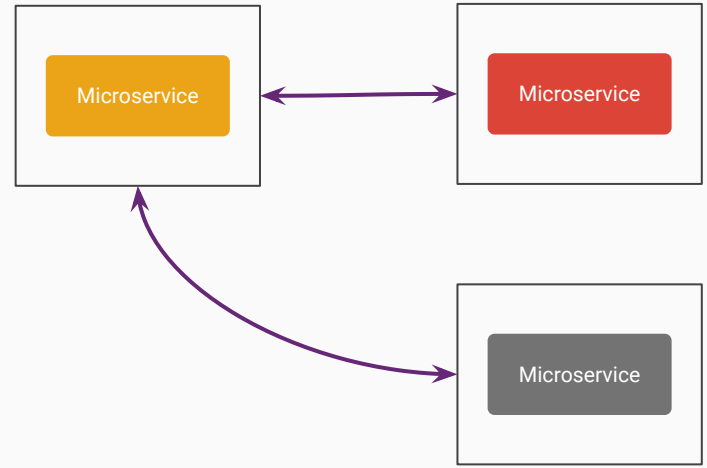
Microservices

Moving to Microservices

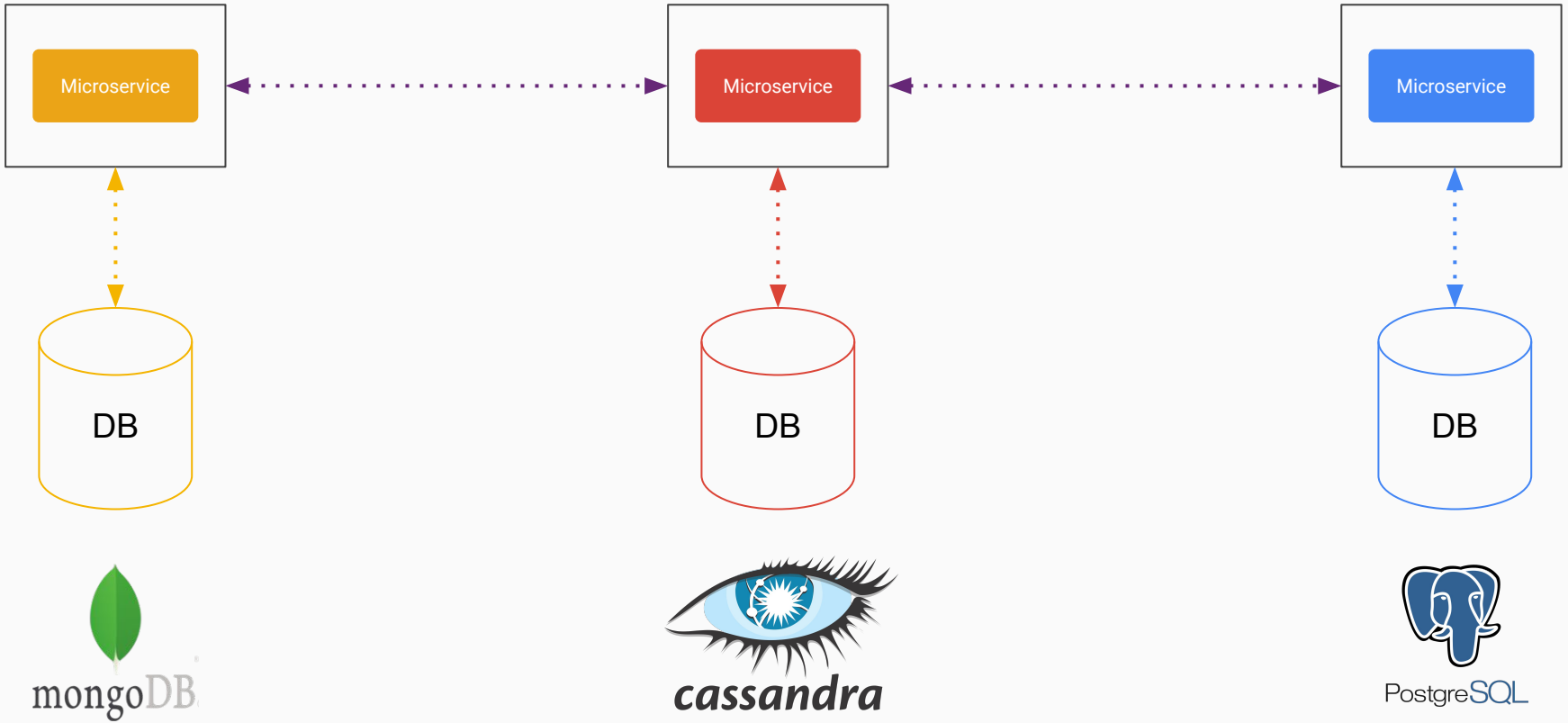
Monolith



Microservices



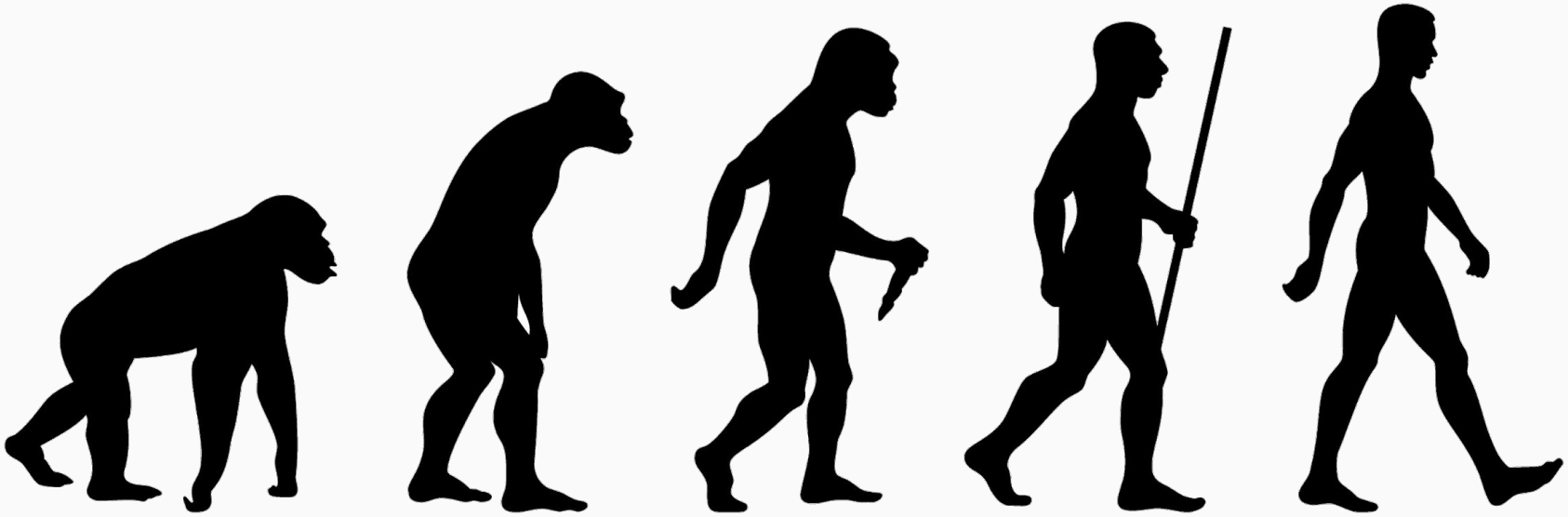
Technological Heterogeneity



Evolutionary Architecture

Evolutionary Architecture

An evolutionary architecture supports guided, incremental change as a first principle across multiple dimensions.



Source: https://www.thoughtworks.com/books/building-evolutionary-architectures

Our stack

Evolutionary Architecture

Languages	
Technologies	  
Databases	 

All(15) microservices were written in

We wanted to change, but there were many challenges!

- Expertise in Java Frameworks e.g: Spring Stack, Hibernate etc.
- Expertise in Object-Oriented Programming
- Learning curve, sprint in progress

is going to overcome all of those challenges?

Yes!!!

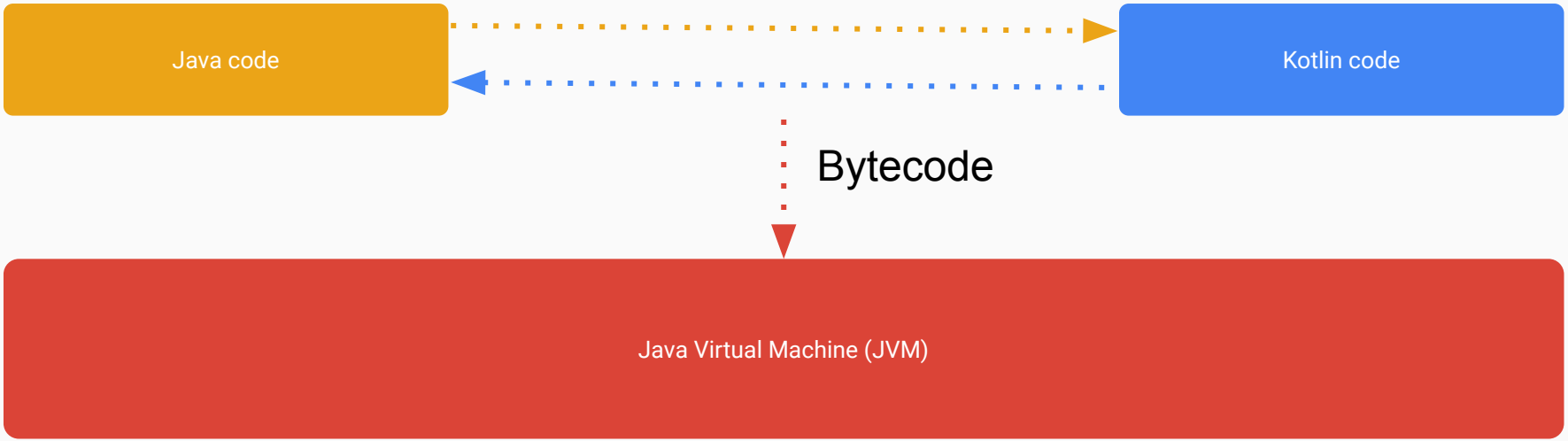
Expertise in Java Frameworks

Expertise in Object-Oriented Programming

Learning curve, sprint in progress

Expertise in Java Frameworks

Interoperability: Kotlin is fully compatible with all **Java-based frameworks**, which lets you stay on your familiar technology stack while reaping the benefits of a more modern language.



Source: https://kotlinlang.org/docs/reference/server-overview.html

Expertise in Java Frameworks

Expertise in Java Frameworks

Expertise in Object-Oriented Programming

Learning curve, sprint in progress

Expertise in Object-Oriented Programming

Object-oriented programming (OOP) is a programming paradigm based on the concept of **"objects"**, which can contain data, in the form of fields (often known as attributes), and code, in the form of procedures (often known as methods).

Code

```
data class Person(val firstName: String, val surname: String, val age: Int) {

   fun fullName() = "$firstName $surname"

}
```

Source: https://en.wikipedia.org/wiki/Object-oriented_programming

Expertise in Object-Oriented Programming

Functional programming is a programming paradigm based structures and elements of computer programs, that treats computation as the evaluation of mathematical functions and avoids **changing-state** and **mutable data**.

Code

```
data class Person(val name: String, String, val age: Int)

val persons = listOf(Person("Person 1", 15), Person("Person 2", 22))

persons.filter { person -> person.age >= 18 }
```

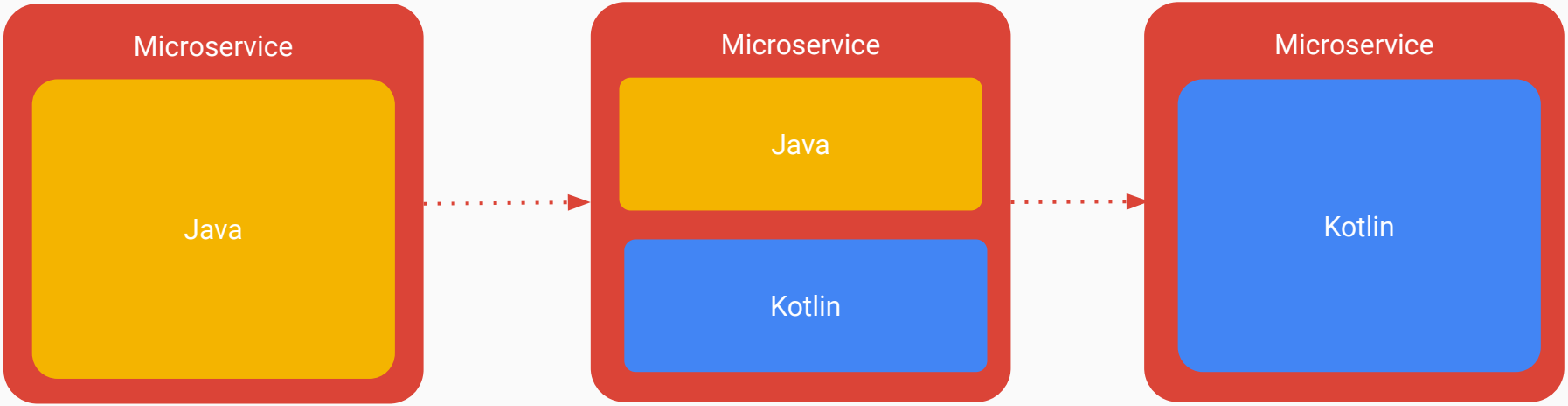
Source: https://en.wikipedia.org/wiki/Functional_programming

Expertise in Java Frameworks
Expertise in Object-Oriented Programming

Learning curve, sprint in progress

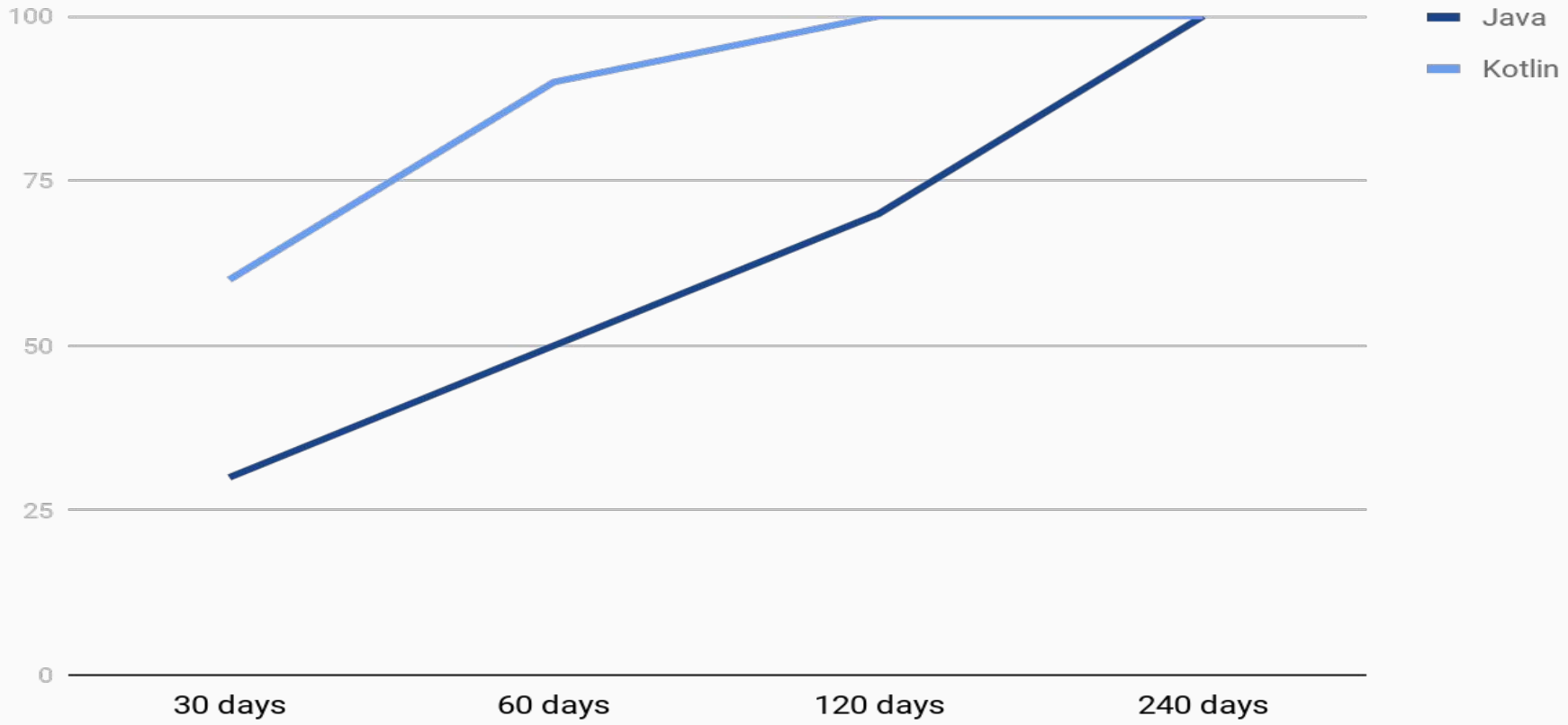
Learning curve, sprint in progress

Migration: Kotlin supports gradual, step by step migration of large codebases from Java to Kotlin. You can start writing **new** code in Kotlin while keeping **older** parts of your system in Java.



Source: https://kotlinlang.org/docs/reference/server-overview.html

Learning curve, sprint in progress



Kotlin benefits

40% less code

40% less code

Java

```
public class Person {

   private final String name;
   private final Integer age;

   public Person(String name, Integer age) {
      this.name = name;
      this.age = age;
   }

   public String getName() {
      return name;
   }

   public Integer getAge() {
      return age;
   }

}
```

Kotlin

```
data class Person(val name: String, val age: Int)
```

Default Values

```
data class Person(val name: String = "Unknown", val age: Int = 0)
```

Initialization

```
val person = Person("Luram Archanjo", 25)

val person = Person() // Name = Unknown & age = 0
```

Null Safety

Null Safety

Java

```
final Person person = null; // accept null

person.getName().length();
```

Exception java.lang.NullPointerException

Kotlin

```
val person: Person = null // compilation error

val person: Person? = null // accept null

person.name.length // compilation error
```

Safe Calls

```
person?.name?.length // return null
```

Elvis Operator

```
person?.name?.length ?: 0 // return 0 if null
```

Performs lots of checks, reducing runtime errors and the number of bugs in the code

Who is using Kotlin?
Why?

Who is using Kotlin? Why?

Source: https://github.com/ygaller/kotlin-companies/wiki

Fail-fast principle

=

Time to market

Summary

2º Place

Concise

- Easier to maintain
- Easier to read
- Easier to apply changes when necessary

1º Place

Interoperable with Java

- Known frameworks
- Low learn curve

3º Place

Modern

- Safe compiler
- Type interface
- Collection
- Lambdas

Thanks a million!

Questions?

/larchanjo

/luram-archanjo