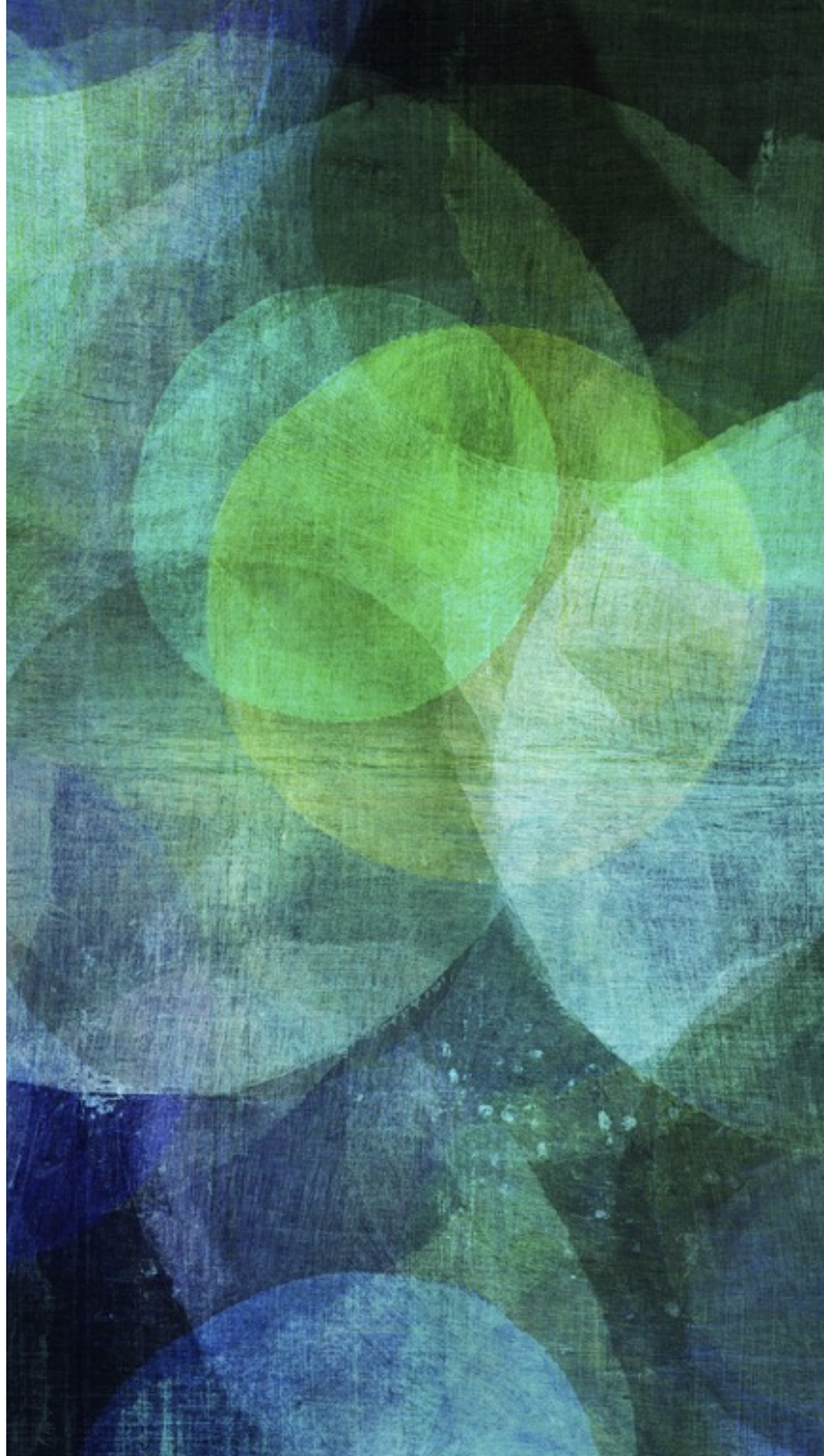
KOTLIN

Introdução ao desenvolvimento no Android



DENIS OLIVEIRA

- Interessado em diversas linguagens e linhas de desenvolvimento.
- Atualmente desenvolvedor iOS
- Gosta de jogar xbox

BEBLUE

- Fintech
- Uma carteira de pagamentos com recompensas e benefícios.

AGENDA

- Apresentação
- Recursos
- Código

KOTLIN

- Foi desenvolvida pela JetBrains
- Começou em 2011
- A versão 1.0 foi 2016
- Em 2017 a Google oficializou como uma das principais linguagens para desenvolvimento no Android

QUEM JÁ USA KOTLIN

- Nubank
- Pinterest:
- Evernote
- Uber
- Pivotal
- Atlassian: Aplicativo do Trello

KOTLIN

- Linguagem de programação tipada estáticamente
- Pode ser compilada para JVM, Javascript, nativo
- Operabilidade com JAVA

JAVA

- Java 8 está disponível somente na SDK minima 24
- Alternativas usar Retrolambda
- Compilador JACK

VANTAGENS

- Com Kotlin é possível suprir todo o gap que linguagens modernas fornecem.

POR ONDE COMEÇAR

- KOAN
- DOC na pagina do Kotlin
- Tutoriais
- ...

TIPOS BASICS

- Números
  - Constantes Literárias
- Caracteres
- Operadores
- Booleanos
- Arrays
- Strings
  - Strings Literais
  - String Template

VISIBILIDADE E MODIFICADORES

- Pacotes
- Classes e Herança
  - private, protected, internal, public
- Modulos

INTERFACES

```
interface MyInterface {
    val prop: Int // abstract

    val propertyWithImplementation: String
        get() = "foo"

    fun foo() {
        print(prop)
    }
}

class Child : MyInterface {
    override val prop: Int = 29
}
```

RESOLVENDO CONFLITO DE INTERFACE

```
interface A {
    fun foo() { print("A") }
    fun bar()
}

interface B {
    fun foo() { print("B") }
    fun bar() { print("bar") }
}

class C : A {
    override fun bar() { print("bar") }
}

class D : A, B {
    override fun foo() {
        super<A>.foo()
        super<B>.foo()
    }

    override fun bar() {
        super<B>.bar()
    }
}
```

CLASSES

- Constructors
  - primario e secundário
- Initializer Block
- Functions
- Properties
- Nested e Inner Class
- Object Declarations
- Por default todas as classe são final sem nenhum marcador

PROPERTIES

- Propriedades mutáveis
  - var
- Propriedades imutáveis
  - val
  - lateinit
  - const
  - get e setter customizado

```
var <propertyName>[: <PropertyType>] [= <property_initializer>]
    [<getter>]
    [<setter>]
```

DATA CLASS

- Quando identificado com a marcação o compilador adiciona
  - equals, hashCode, toString, copy, componentsN

SEALED CLASS

- Em paralelo ao java são classes abstratas que não podem ser instanciadas

NESTED E INNER CLASS

```
class Outer {
    private val bar: Int = 1
    class Nested {
        fun foo() = 2
    }
}
```

```
class Outer {
    private val bar: Int = 1
    inner class Inner {
        fun foo() = bar
    }
}
```

EXTENSIONS

- Uma forma de criar composição de um objeto as extensions são resolvidas estaticamente
  - Functions
  - Properties
  - Companion Objects
  - Scope
  - Members

EXTENSIONS

```
fun <T> MutableList<T>.swap(index1: Int, index2: Int) {
    val tmp = this[index1] // 'this' corresponds to the list
    this[index1] = this[index2]
    this[index2] = tmp
}
```

```
val l = mutableListOf(1, 2, 3)
l.swap(0, 2) // 'this' inside 'swap()' will hold the value of 'l'
```

OUTROS

- Enum Class
- Objects
  - Objects Declarations
  - Objects Expression
    - Companion Objects
- Delegation
- Delegation Properties

RECURSOS

- Null Safe
- Composição
- Sobrecarga de operadores

OPERADORES

- Safe call ?.
- Unsafe call !!
- Elvis operator ?:
- Sobrecarga

SOBRECARGA DE OPERADORES

Expression	Translated to
`+a`	`a.unaryPlus()`
`-a`	`a.unaryMinus()`
`!a`	`a.not()`

Expression	Translated to
`a++`	`a.inc()` + see below
`a--`	`a.dec()` + see below

Expression	Translated to
`a in b`	`b.contains(a)`
`a !in b`	`!b.contains(a)`

Expression	Translated to
`a + b`	`a.plus(b)`
`a - b`	`a.minus(b)`
`a * b`	`a.times(b)`
`a / b`	`a.div(b)`
`a % b`	`a.rem(b)`, `a.mod(b)` (deprecated)
`a..b`	`a.rangeTo(b)`

OUTRAS FUNÇÕES EMBUTIDAS NA LINGUAGEM

- Coroutines
  - Uma forma imperativa de escrever códigos assíncronos

TESTE COM KOTLIN

- Spek

KOTLIN ANDROID EXTENSIONS

- Atualmente é plugin para o Android Studio
- Elimina a necessidade de usar o findViewById
- Todos os ids declarados no xml torna-se acessíveis no código

ANKO

- Anko -> (An)droid (Ko)tlin
- É um conjunto de quatro bibliotecas que facilitam o desenvolvimento de aplicações Android reduzindo muito boylerplate
- Os modulos são:
  - Commons
  - Layouts
  - Sqlite
  - Coroutines

BIBLIOTECAS CONHECIDAS PARA ANDROID

- Retrofit
- RxJava
- Realm
- ...

LET'S DO IT

O QUE VIMOS ANTES NA PRÁTICA

- Iniciando com Kotlin
- Configurando o ambiente
- Configurando os plugins no projeto
- Usando Kotlin Android Extension Plugin
- Usando Anko
- Usando Retrofit

REFERENCIA

- https://kotlinlang.org
- https://github.com/Kotlin/anko
- http://spekframework.org
- https://www.kotlindevelopment.com