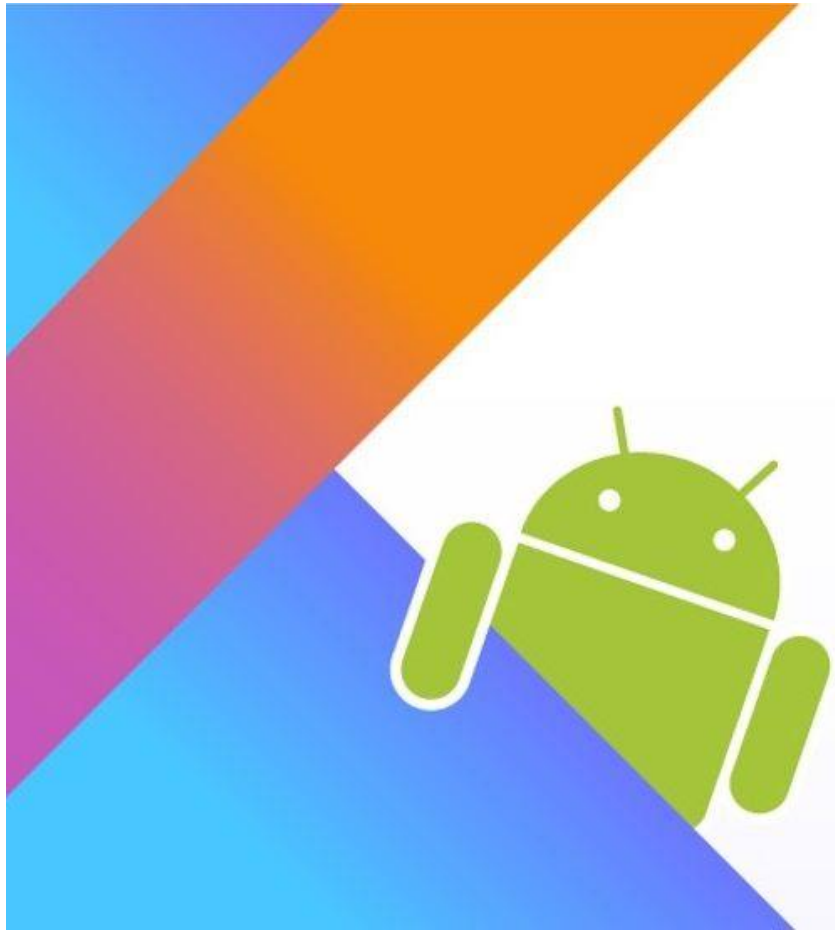
UNI GUAIRACÁ
CENTRO UNIVERSITÁRIO



Kotlin
for Android App Development

DESENVOLVIMENTO MOBILE

CONTATO

AlessandroStamatto @ gmail

AMBIENTE

Java: baixem e instalem a JDK 15

https://jdk.java.net/15/

IDE Android/Kotlin: baixem e instalem o Android Studio

https://developer.android.com/studio

PROGRAMAÇÃO DA DISCIPLINA

- Kotlin: o que vocês precisam saber da linguagem

(https://www.udacity.com/course/kotlin-bootcamp-for-programmers--ud9011)

- Grade Oficial da Google para Desenvolvimento Android Moderno

https://classroom.udacity.com/courses/ud9012

- Projeto de um aplicativo “completão”

GRADE ANDROID DA GOOGLE

1. Componentes básicos Android
2. Elementos gráficos Android & Constraints
3. Navegação de Telas & Fragmentos
4. Ciclo de Vida Android
5. Interfaces Gráficas Dinâmicas & Binding

GRADE ANDROID DA GOOGLE

6. Persistência de Dados Android & Room
7. Otimização Gráfica com Reciclagem
8. Comunicação com APIs e Recursos (Retrofit e Glide)
9. Serviços e Cacheamento Offline
10. Design avançado & Acessibilidade

KOTLIN

Desenvolvida pela

Necessidade de uma linguagem moderna

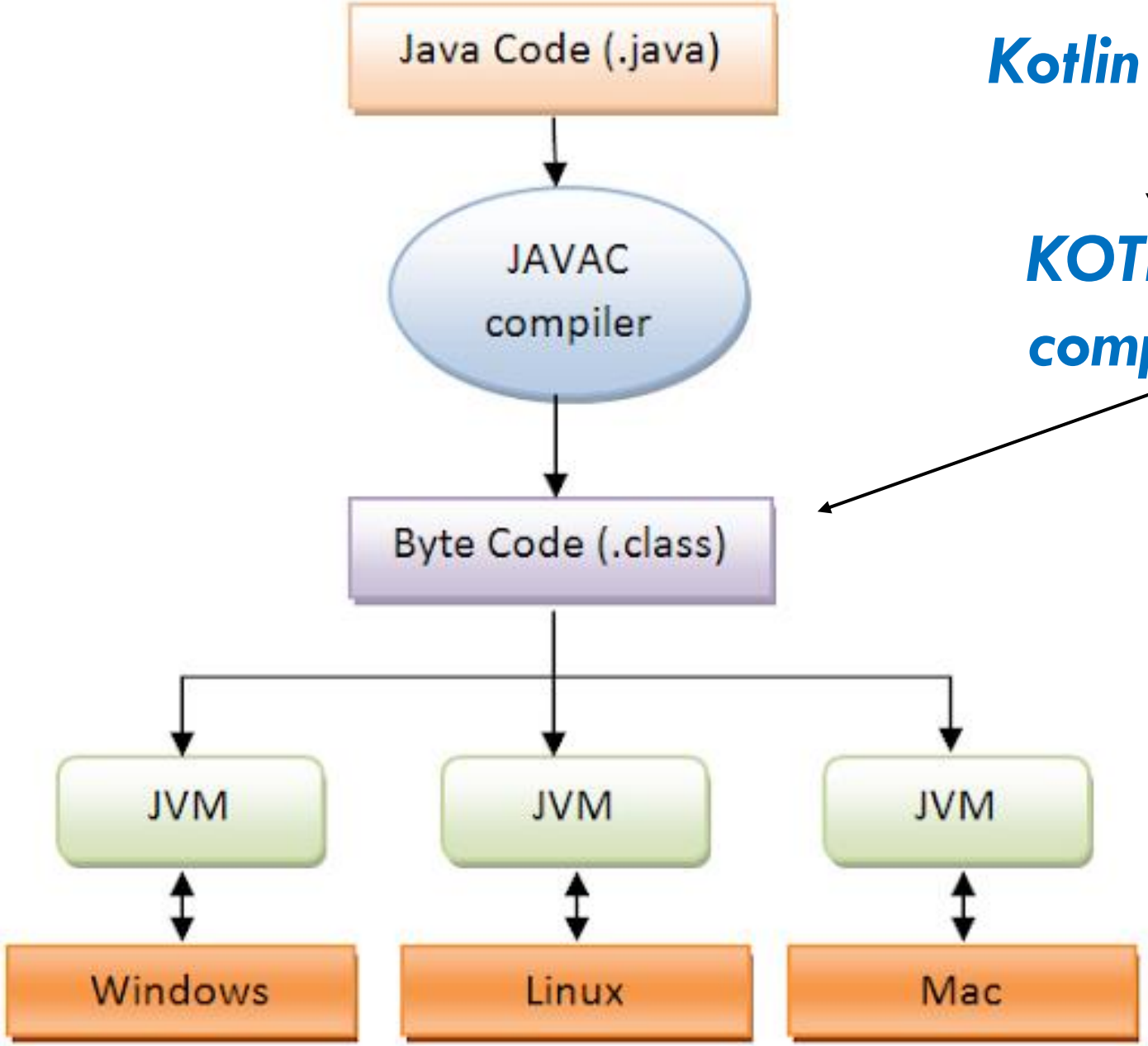
KOTLIN

2013 – IntelliJ IDEA como IDE de desenvolvimento Android: ***Android Studio***

2017 – Nova linguagem para desenvolvimento Android

2019 – Linguagem oficial e **principal** para desenvolvimento Android

KOTLIN


Java Code (.java)
JAVAC compiler
Byte Code (.class)
JVM
JVM
JVM
Windows
Linux
Mac
Kotlin Code
KOTLINC compiler

KOTLIN – VANTAGENS?

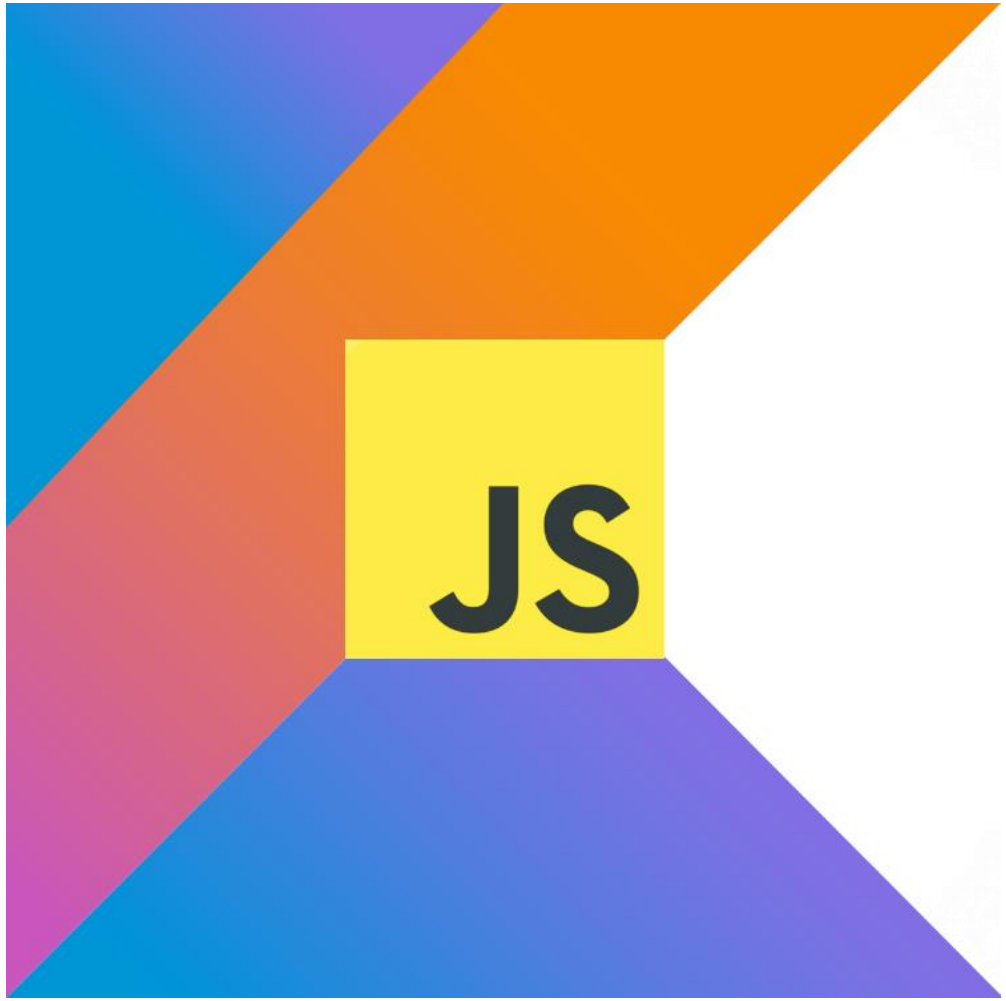
40% mais conciso

Evita erros de *null*

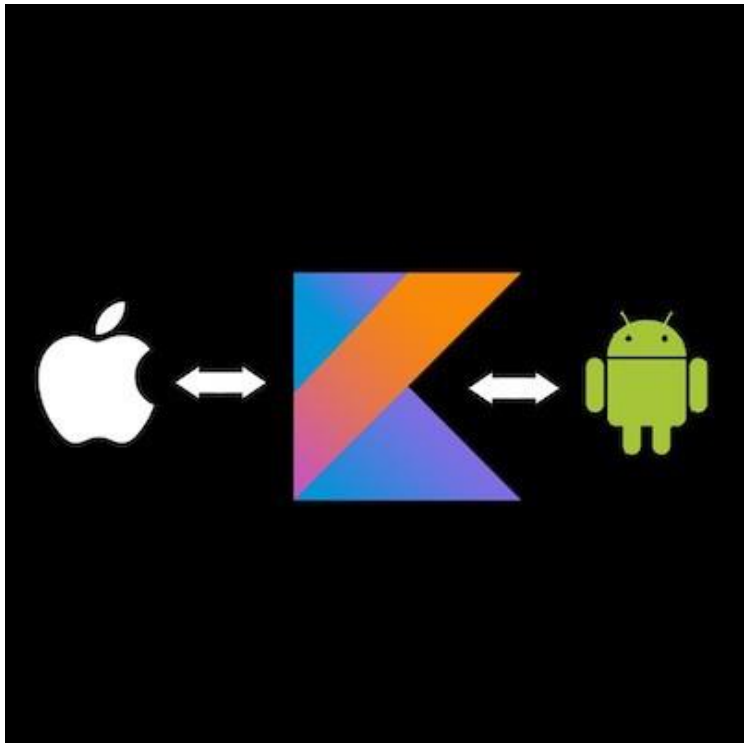
Evita erros de conversão de tipos

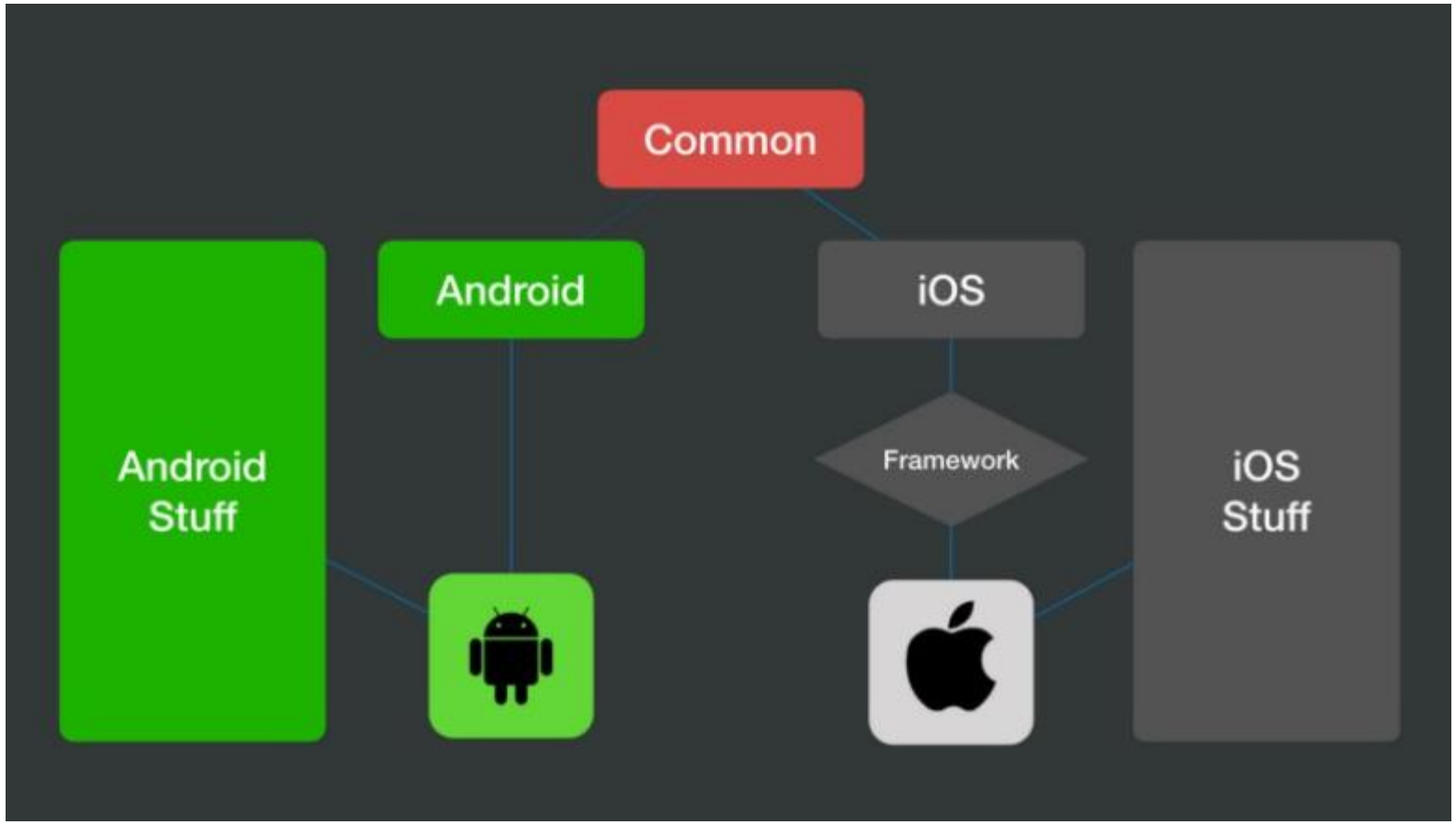
Várias facilidades modernas para programadores

KOTLIN – JAVASCRIPT



KOTLIN – MULTIPLATAFORMA MOBILE ("BETA")





KOTLIN – EXPERIMENTE

play.kotlinlang.org

Kotlin Playground is an online sandbox to explore Kotlin programming language. Browse code samples directly in the browser

```
/**
 * You can edit, run, and share this code.
 * play.kotlinlang.org
 */

fun main() {
    println("Hello, world!!!")
}
```

```
Hello, world!!!
```

KOTLIN – MULTIPARADIGMA

Tipagem Estática com inferência

Orientada a Objetos

Funcional

Concorrente

Extensiva

KOTLIN

Tipagem Estática

```kotlin
fun main() {
    val nome: String = "Alessandro Stamatto"
    val inicial: Char = 'A'
    val idade: Int = 34
    val altura: Float = 1.65

    println("Nome: $nome, Idade: $idade")
}
```

```
Nome: Alessandro Stamatto, Idade: 34
```

KOTLIN

Tipagem Estática **com inferência**

```
fun main() {
    val nome = "Alessandro Stamatto"
    val inicial = 'A'
    val idade = 34
    val altura = 1.65

    println("Nome: $nome, Idade: $idade")
}
```

```
Nome: Alessandro Stamatto, Idade: 34
```

KOTLIN

Orientada a Objetos

```kotlin
class Pessoa(val nome: String, val idade: Int) {
    fun apresentar() {
        println("Sou $nome, $idade anos.")
    }
}
fun main() {
    val prof = Pessoa("Alessandro Stamatto", 34)
    prof.apresentar()
}
```

```
Sou Alessandro Stamatto, 34 anos.
```

KOTLIN

Funcional

```
fun main() {
    val numeros = listOf(1, 2, 3, 4, 5, 6, 7, 8)
    val pares = numeros.filter { it % 2 == 0 }
    println(pares.joinToString(","))
}
```

```
2,4,6,8
```

KOTLIN

Concorrente

```
suspend fun main() = coroutineScope {
    for (i in 0 until 10) {
        launch {
            delay(1000L - i * 10) //quanto maior espere menos
            print("$i ")
        }
    }
}
```

```
9 8 7 6 5 4 3 2 1 0
```

KOTLIN

Concorrente

```
suspend fun processarImagens(): Imagem = coroutineScope {
    val imagem1 = async { baixarImagem1() }
    val imagem2 = async { baixarImagem2() }
    val imagem_junta = juntarImagens(imagem1.await(), imagem2.await())

    return imagem_junta
}
```

KOTLIN

Extensiva

```kotlin
fun String.primeiroNome(): String {
    return split(" ")[0]
}
fun main() {
    println("Alessandro Stamatto".primeiroNome())
}
```

```
Alessandro
```

KOTLIN - VARIÁVEIS

```kotlin
fun main() {
    val imutável = 5
    imutável = imutável + 3

    println(imutável)
}
```

```
Val cannot be reassigned
```

```kotlin
fun main() {
    var mutável = 5
    mutável = mutável + 3

    println(mutável)
}
```

```
8
```

KOTLIN

Você pode escrever o tipo explicitamente

```kotlin
fun main() {
    val nome: String = "Alessandro Stamatto"
    val inicial: Char = 'A'
    val idade: Int = 34
    val altura: Float = 1.65

    println("Nome: $nome, Idade: $idade")
}
```

```
Nome: Alessandro Stamatto, Idade: 34
```

KOTLIN – NULO

```
var JAMAIS_é_nulo: String = "aha"
var PODE_ser_nulo_me_Salve: String? = null
//var Java_PODE_ser_nulo_mas_me_Deixa: String! = null


println(PODE_ser_nulo_me_Salve.length) //Erro: Se pá é nulo...
```

KOTLIN – NULO

```
var PODE_ser_nulo_me_Salve: String? = null

if (PODE_ser_nulo_me_Salve != null)
    println(PODE_ser_nulo_me_Salve.length)
```

KOTLIN – NULO

```
    //!! significa: "Confia que eu sei o que to fazendo"
    println(PODE_ser_nulo_me_Salve!!.length)
}
```

```
Exception in thread "main" java.lang.NullPointerException
  at FileKt.main (File.kt:6)
  at FileKt.main (File.kt:-1)
  at sun.reflect.NativeMethodAccessorImpl.invoke0 (NativeMethodAccessorImpl.java:-2)
```

KOTLIN – NULO

```
    //?! segue em frente se não for nulo,
    //senão desiste e a expressão inteira vira nulo
    println(PODE_ser_nulo_me_Salve?.length)
}
```

```
null
```

KOTLIN – NULO

```
var PODE_ser_nulo_me_Salve: String? = null

//?: não faz nada se NÃO for nulo
//mas se for nulo troca a expressão inteira por um valor
println(PODE_ser_nulo_me_Salve?.length ?: 0)
```

```
0
```

KOTLIN - COLEÇÕES

```
fun main() {
    //Lista (Vetor dinâmico)
    val chamada = listOf("Amanda", "João", "Alex", "Maria")
    val ordenada = chamada.sorted()
    println(chamada.joinToString(","))
}
```

```
Amanda,João,Alex,Maria
```

KOTLIN - COLEÇÕES

```kotlin
fun main() {
    //Conjuntos
    val frutas = setOf("Melancia", "Morango", "Açaí")

    println(if (frutas.contains("Morango")) "tem morango!" else "não tem...")
}
```

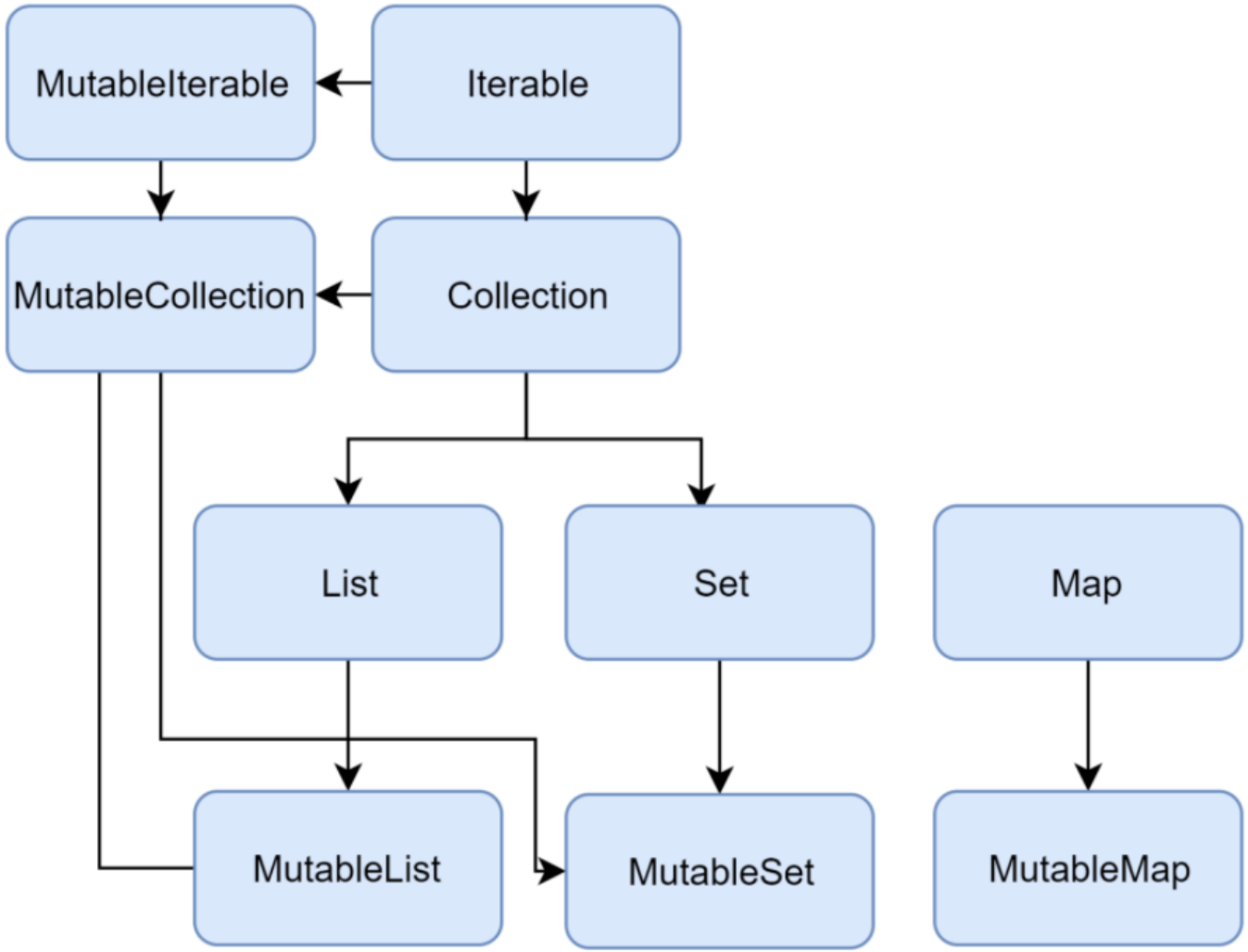
```kotlin
val resposta =
    if ("Morango" in frutas)
        "tem morango!"
    else
        "não tem..."
```

KOTLIN - COLEÇÕES

```kotlin
// Mapa (Vetor Associativo)
val mapa_associação = mapOf(
    "Chave1" to "Valor1",
    "Chave2" to "Valor2",
    "Chave3" to "Valor3"
)
for((k, v) in mapa_associação) {
    println("$k: $v")
}
```

```
Chave1: Valor1
Chave2: Valor2
Chave3: Valor3
```

KOTLIN - COLEÇÕES



KOTLIN – CONDICIONAIS

```
val usuario = "James" + " " + "Bond"

println(if(usuario == "James Bond") "autorizado" else "rejeitado")
```

KOTLIN – CONDICIONAIS

```
val nome = "Arturia"

val arma =
    when (nome) {
        "Arturia" -> "Espada Excalibur"
        "Chulaind" -> "Lança Gáe-Bolg"
        else -> "sem arma"
    }
```

KOTLIN – CONDICIONAIS

```kotlin
val nota  = 7.2
val avaliação =
    when {
        nota > 8 -> "Excelente"
        nota > 5 -> "Assistível"
        else -> "Lixo"
    }

println(avaliação)
```

KOTLIN – LAÇOS

```
for (i in 1..10)
    println(i)
for (i in 6 downTo 0 step 2)
    println(i)
for (i in listOf(7, 15, 23))
    println(i)
for ((idx, i) in listOf(7, 15, 23).withIndex())
    println("$idx $i")
```

KOTLIN – CLASSES

```
//parâmetros var e val formam parte do "construtor primário"
class Classe(var texto: String = "", qtd: Int = 1) {
    //var e val 'top-level' também fazem parte da construção primária
    val vezes = if (qtd < 1) 1 else qtd

    //código de construção primária deve ser colocado em bloco init
    init {
        texto = "|" + texto + "|"
    }
```

KOTLIN – CLASSES

```
class Classe(var texto: String = "", qtd: Int = 1) {
    val vezes = if (qtd < 1) 1 else qtd

    fun imprimir() {
        for (i in 1..this.vezes)
            println(this.texto)
    }
}
```

KOTLIN – CLASSES

```
class Classe(var texto: String = "", qtd: Int = 1) {
    val vezes = if (qtd < 1) 1 else qtd

    fun imprimir() {
        for (i in 1..vezes)
            println(texto)
    }
}
```

KOTLIN – CLASSES

```
class Classe(var texto: String = "", qtd: Int = 1) {
    val vezes = if (qtd < 1) 1 else qtd

    //construtores secundários precisam chamar o primário
    constructor(qtd: Int) : this("Padrão", qtd) {
        texto = "!" + texto + "!"
    }
```

KOTLIN – CLASSES

```kotlin
open class Pessoa(var nome: String) {
    open fun apresentar() {
        println("Pode me chamar $nome")
    }
}
class Professor(nome: String): Pessoa(nome) {
    override fun apresentar() {
        println("Pode me chamar de Prof. $nome")
    }
}
```

KOTLIN – FUNÇÕES

```kotlin
fun lower(s: String) = s.toLowerCase()

//extender String para ter o método lowers
//o this é a string onde chamamos o método
fun String.lowers() = this.toLowerCase()

fun main() {
    //"Woa" vira o this
    println("Woa".lowers())
}
```

KOTLIN – FUNÇÕES

```
//this pode ser omitido!
fun String.lowers() = toLowerCase()
```

KOTLIN – FUNÇÕES

```kotlin
val numeros = listOf(1,2,3,4,5,6)
var pares: List<Int> = listOf()

pares = numeros.filter( fun (it: Int) = it % 2 == 0)

pares = numeros.filter( {it: Int -> it % 2 == 0})
pares = numeros.filter( {it % 2 == 0} )
pares = numeros.filter {
    it % 2 == 0
}
```

KOTLIN – EXERCÍCIOS

Palíndromo: verifique que uma dada String é igual lida da direita para esquerda. Ex: *Arara*

Anagrama: verifique se duas Strings possivelmente diferentes usam as mesmas letras. Ex: *Lobo* e *Bolo*

KOTLIN – EXERCÍCIOS

Palíndromo: verifique que uma dada String é igual lida da direita para esquerda. Ex: *Arara*

```
fun éPalindromo(s: String) =
    s.toLowerCase() == s.toLowerCase().reversed()
```

KOTLIN – EXERCÍCIOS

Anagrama: verifique se duas Strings possivelmente diferentes usam as mesmas letras. Ex: *Lobo* e *Bolo*

```
fun String.alfabetizada() = toLowerCase().toCharArray().sorted()

fun éAnagrama(a: String, b: String) =
    a.alfabetizada() == b.alfabetizada()
```

ANDROID – COMPONENTES BÁSICOS

Vamos criar um Projeto ***"Empty Activity"***, Kotlin, API 22

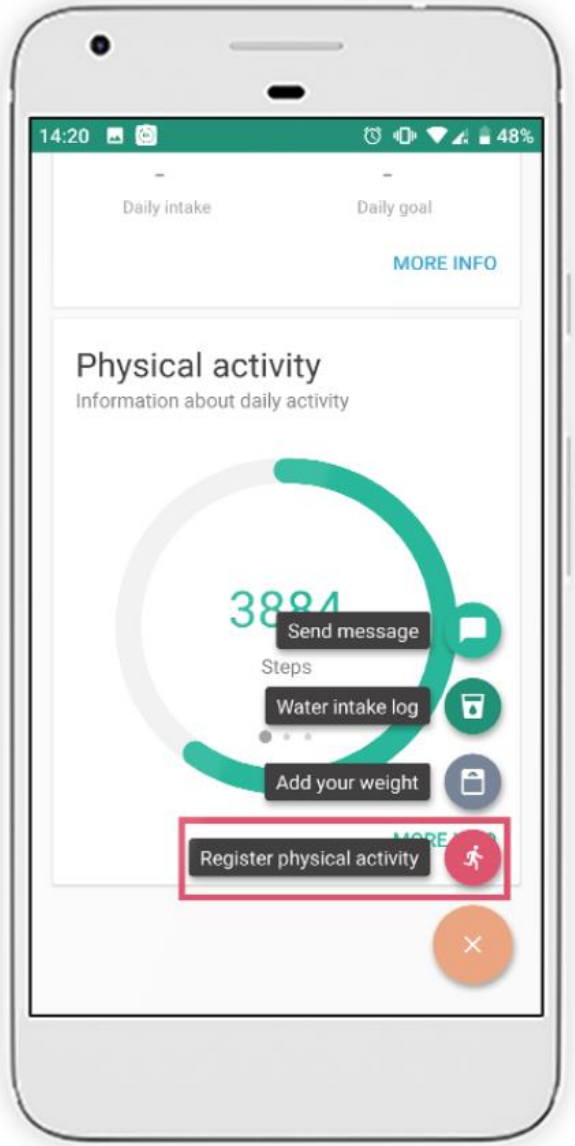
ANDROID – COMPONENTES BÁSICOS

Estrutura de Arquivos:

- *manifests/AndroidManifest.xml* : *configurações gerais, define coisas como Nome, Permissões, Ícone...*
- *java* : códigos Java/Kotlin, implementação
- res : recursos como imagens, definições de interface, strings…

ANDROID – COMPONENTES

Activity : Uma Janela/Tela interativa.

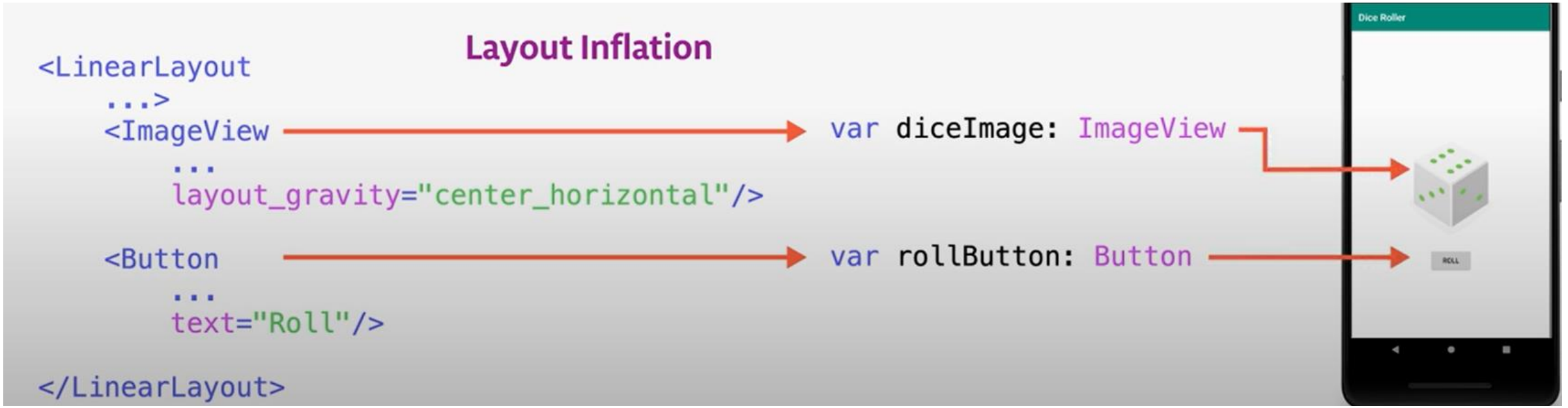


ANDROID – COMPONENTES BÁSICOS

Inflation : Seguir uma “receita” construindo objetos gráficos

ANDROID – COMPONENTES BÁSICOS

Inflation : Seguir uma “receita” construindo objetos gráficos



ANDROID – COMPONENTES BÁSICOS

View : Um objeto gráfico visível, exemplos como TextView, Button, ImageView, CheckBox, RadioButton, EditText, ProgressBar...

ANDROID – LAYOUT

- Constraint Layout é mais avançado, vamos trocar por Linear Layout
- Height wrap content
- Orientation vertical

ANDROID – TEXT

- Aumentar o texto para 50 Scale-independent Pixel (sp) com o atributo textSize
- Mudar o texto para um número do dado

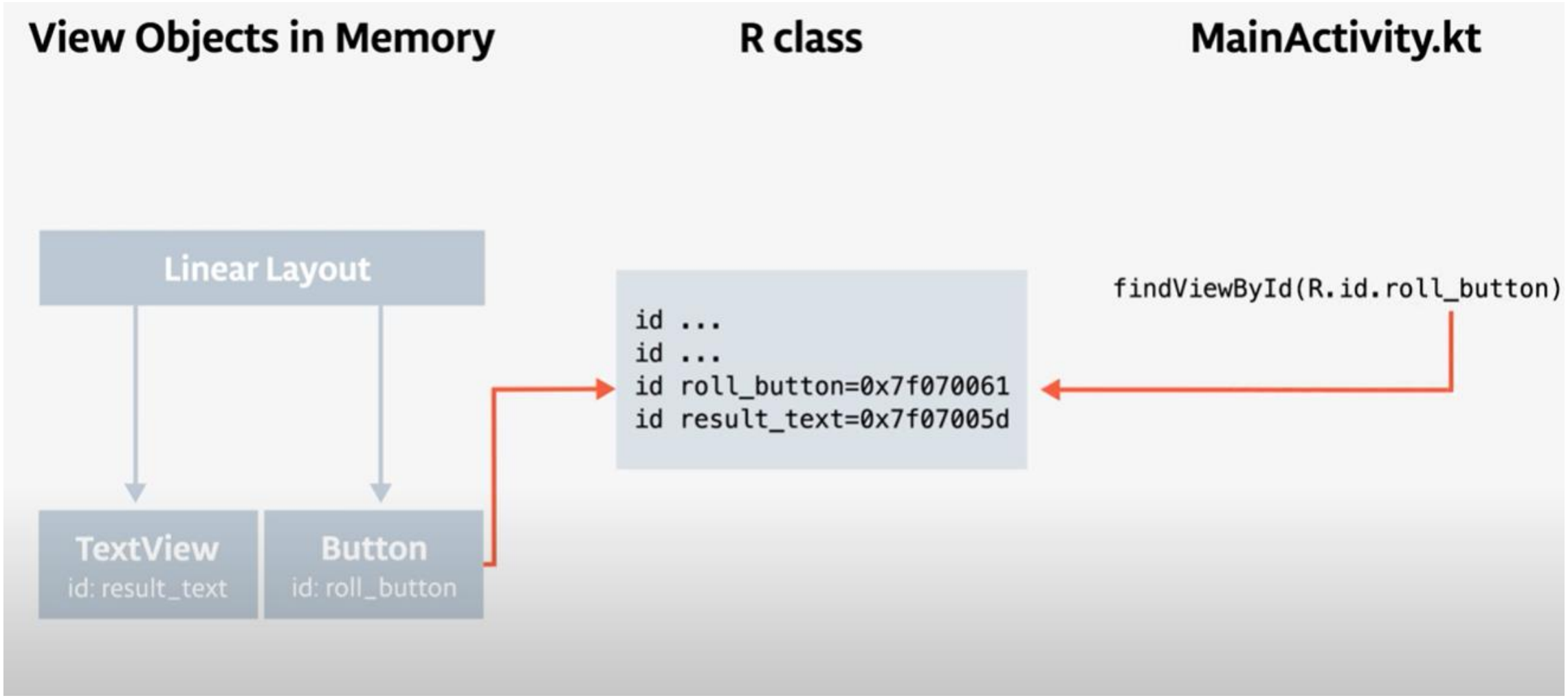
ANDROID – BUTTON

- Botão “Rolar dado”
- Extrair string resource

ANDROID – GRAVIDADE

- Puxar o Layout inteiro para o centro vertical com layout_gravity de center_vertical
- Puxar o texto e o botão para o centro horizontal com layout_gravity de center_horizontal

ANDROID – IDENTIFICADORES



ANDROID – PROGRAMATICAMENTE

```
val botão: Button = findViewById(R.id.roll_button)

botão.text = "Tá na hora do Show"
```

ANDROID – PROGRAMATICAMENTE

```
botão.setOnClickListener { it: View!
    Toast.makeText(
        context: this,
        text: "botão clicado!", Toast.LENGTH_SHORT
    ).show()
}
```

ANDROID – PROGRAMATICAMENTE

```
    botão.setOnClickListener {rolarDado()}
}

fun rolarDado() {
    val resultText: TextView = findViewById(R.id.result_text)
    val de1a6 = Random().nextInt( bound: 6) + 1
    resultText.text = de1a6.toString()
}
```

ANDROID – RESOURCES

Baixar imagens de dados:

https://github.com/udacity/andfun-kotlin-dice-roller/raw/master/DiceImages.zip

- Arrastar para res/drawable SEM SER -v24

- Mudar TextView para ImageView com src dice1

ANDROID – RESOURCES

```
val dado_imagem: ImageView = findViewById(R.id.dado_imagem)
val de1a6 = Random().nextInt( bound: 6) + 1
val faces = listOf(
    R.drawable.empty_dice,
    R.drawable.dice_1,
    R.drawable.dice_2,
    R.drawable.dice_3,
    R.drawable.dice_4,
    R.drawable.dice_5,
    R.drawable.dice_6,
)
dado_imagem.setImageResource(faces[de1a6])
```

ANDROID – OTIMIZANDO FINDVIEW

```
//lateinit = "Confia que vou inicializar antes de usar"
lateinit var dado_imagem: ImageView

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    var botão: Button = findViewById(R.id.roll_button)

    dado_imagem = findViewById(R.id.dado_imagem)
```