

<https://github.com/PacktPublishing/Android-Programming-with-Kotlin-for-Beginners>

# Android com Kotlin

Aula 05 - UI e Kotlin

## Android Programming with Kotlin for Beginners

Build Android apps starting from zero programming experience  
with the new Kotlin programming language



John Horton

# Aula 05 - UI e Kotlin

- Faremos a conexão entre o código kotlin e os layouts xml
- Exemplo com o Button e TextView
- Nullability em Kotlin

# Conexão entre Código (kt) e Layout (xml)

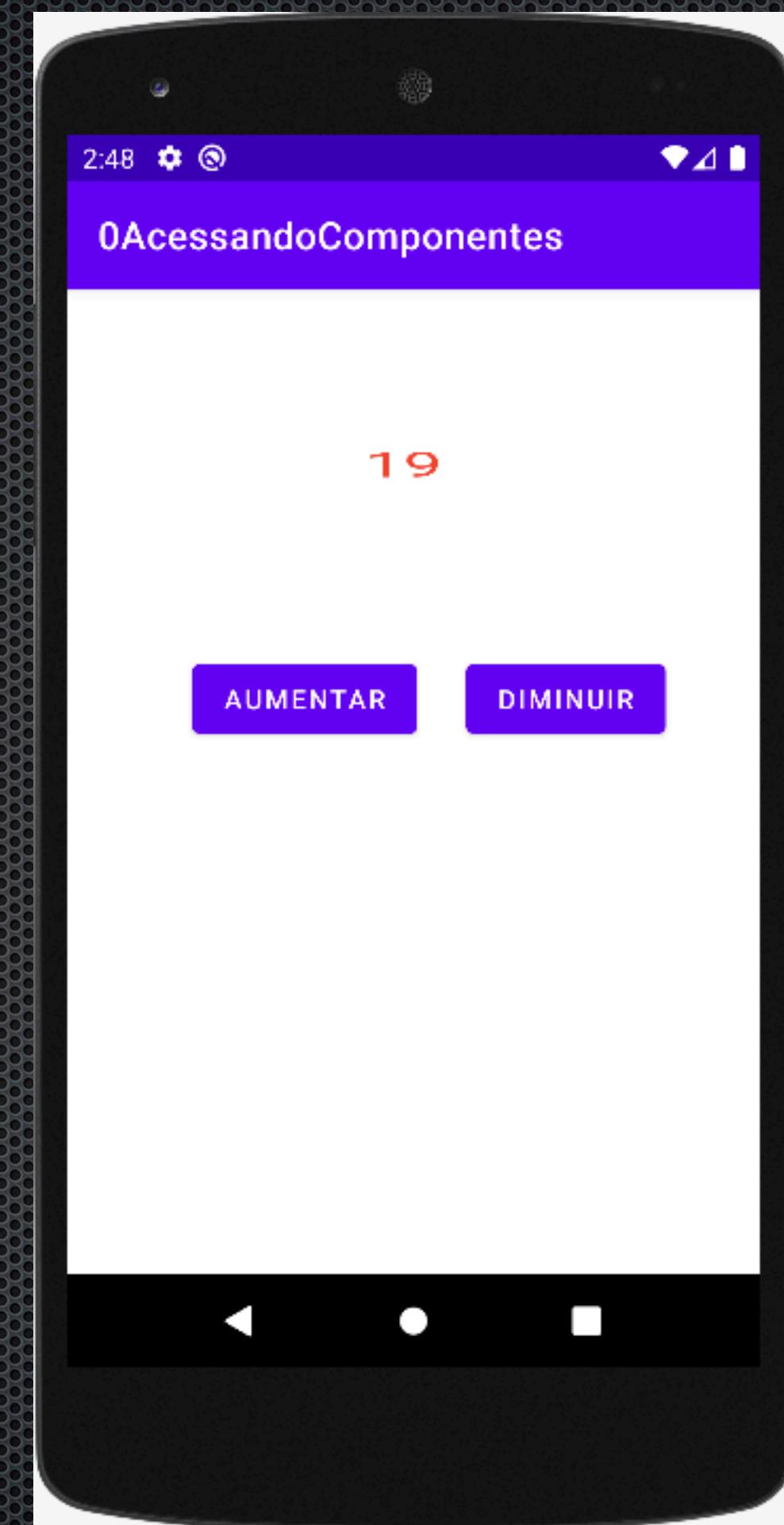
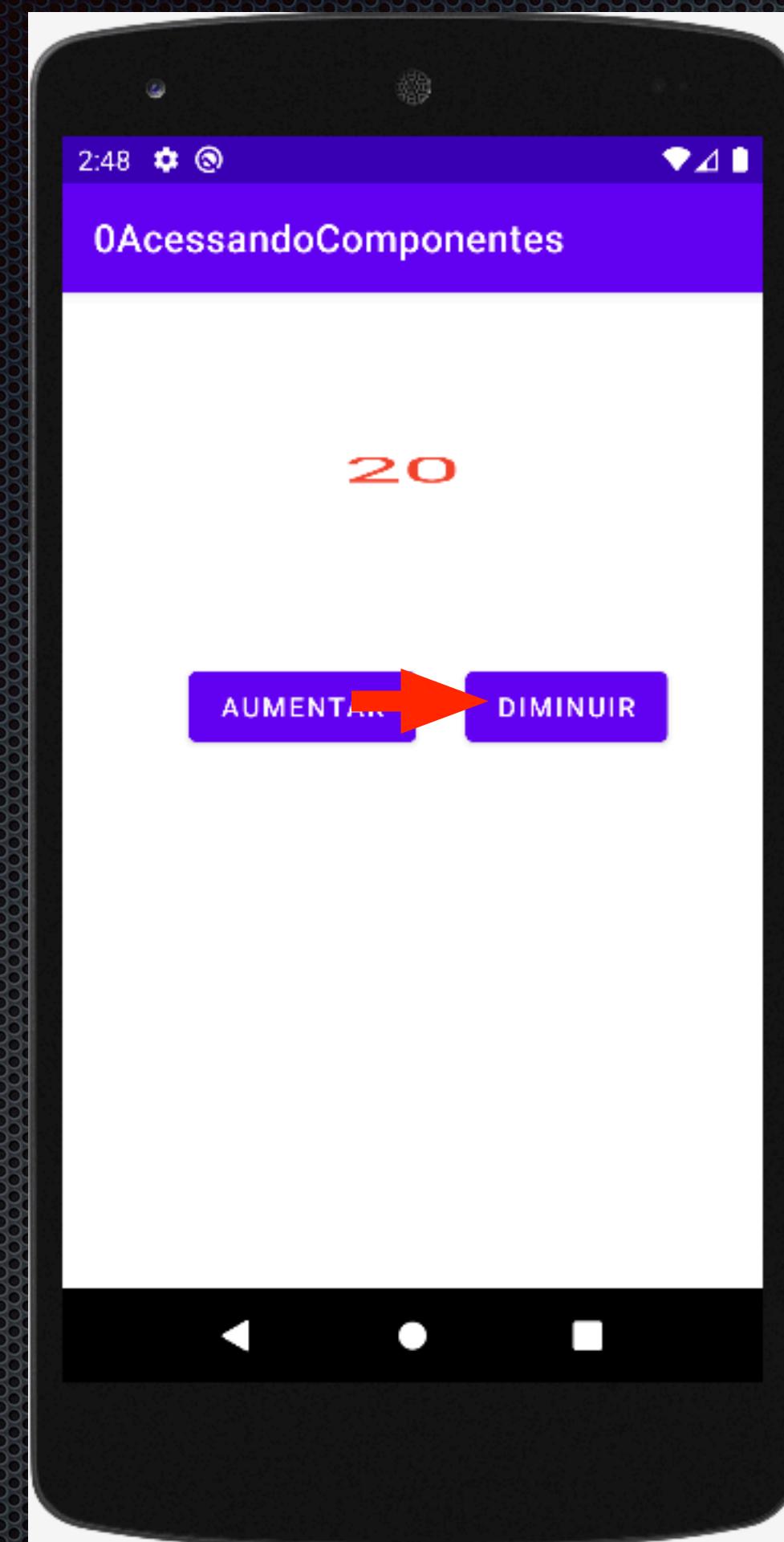
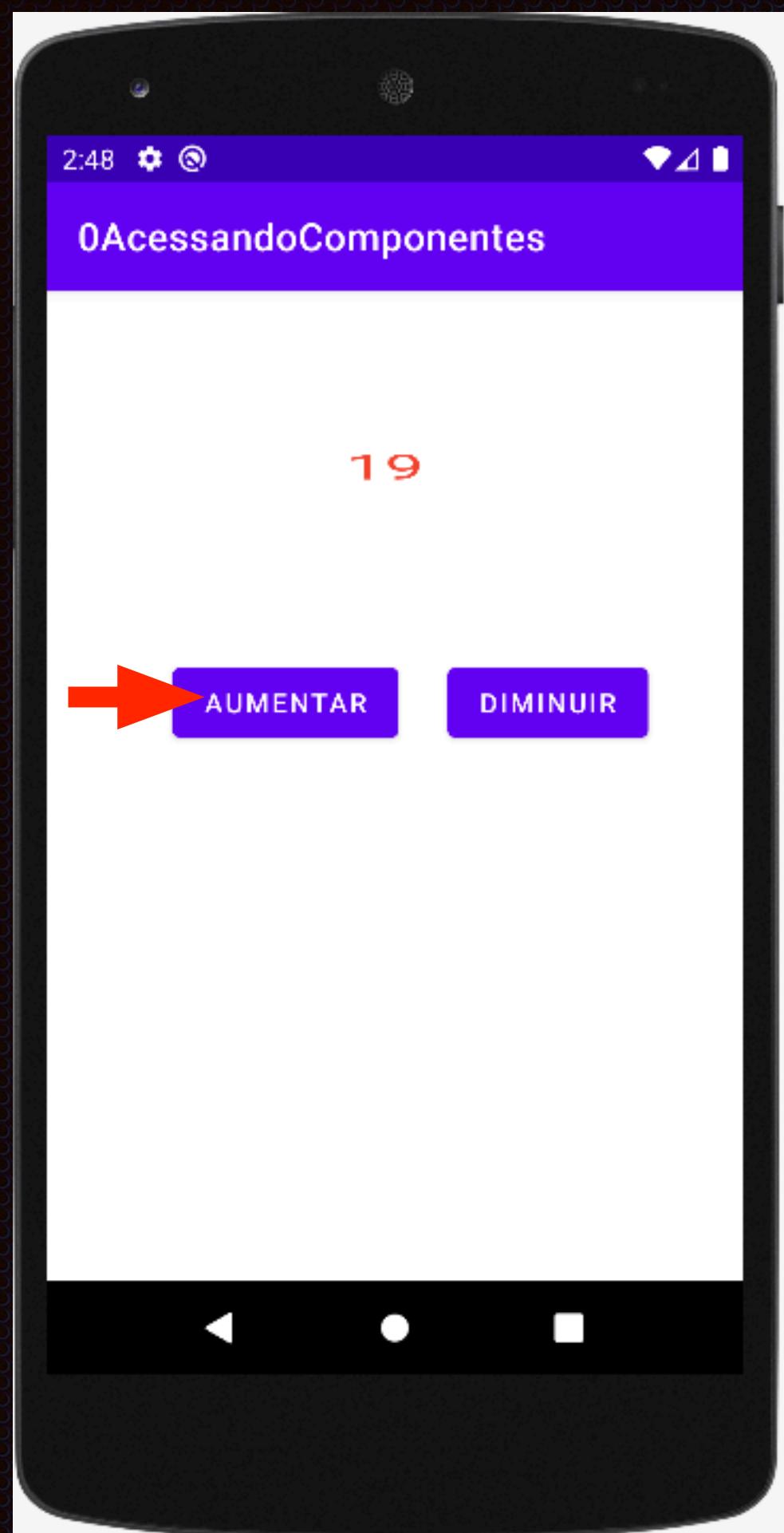
# Elementos UI (User Interface)

- Quando rodamos nosso App e na função `onCreate()`, chamamos o método `setContentView()` carregamos os elementos UI do XML para memória como **objetos**.
- Estes objetos são armazenados em uma parte da memória chamado **Heap**. Não temos acesso direto, mas podemos referenciar algo que está armazenado lá. É aqui que ocorre o garbage collection.
- Já variáveis declaradas dentro de funções são armazenadas em um local da memória chamado **Stack**.

# Acesso à Componentes XML

- Podemos acessar os componentes através de seus ids
- Alterar atributos de um objeto (ex: button.text = “nome”)
- Implementar uma interface a partir de um objeto (ex: OnClickListener de um Button)

# Exemplo 1



```
9      <TextView  
10         android:id="@+id/texto"  
11         android:layout_width="139dp"  
12         android:layout_height="77dp"  
13         android:text="18"  
14         android:textSize="18dp"  
15         android:textAlignment="center"  
16         android:textAllCaps="true"  
17         android:textColor="#F44336"  
18         app:layout_constraintBottom_toBottomOf="parent"  
19         app:layout_constraintEnd_toEndOf="parent"  
20         app:layout_constraintHorizontal_bias="0.476"  
21         app:layout_constraintStart_toStartOf="parent"  
22         app:layout_constraintTop_toTopOf="parent"  
23         app:layout_constraintVertical_bias="0.18" />  
24  
25      <Button  
26          android:id="@+id/diminuir"  
27          android:layout_width="wrap_content"  
28          android:layout_height="wrap_content"  
29          android:text="Diminuir"  
30          app:layout_constraintBottom_toBottomOf="parent"  
31          app:layout_constraintEnd_toEndOf="parent"  
32          app:layout_constraintHorizontal_bias="0.809"  
33          app:layout_constraintStart_toStartOf="parent"  
34          app:layout_constraintTop_toTopOf="parent"  
35          app:layout_constraintVertical_bias="0.407" />  
36  
37      <Button  
38          android:id="@+id/aumentar"  
39          android:layout_width="wrap_content"  
40          android:layout_height="wrap_content"
```

Precisamos colocar os ids no XML

```
3 import androidx.appcompat.app.AppCompatActivity  
4 import android.os.Bundle  
5 import android.view.View  
6 import android.view.View.OnClickListener  
7 import android.widget.Button  
8 import android.widget.TextView  
9  
10 class MainActivity : AppCompatActivity(), OnClickListener  
11  
12     private var tamanho = 18  
13     private lateinit var aumentar:Button  
14     private lateinit var diminuir:Button  
15     private lateinit var texto: TextView  
16  
17     override fun onCreate(savedInstanceState: Bundle?) {  
18         super.onCreate(savedInstanceState)  
19         setContentView(R.layout.activity_main)  
20         aumentar = findViewById(R.id.aumentar)  
21         diminuir = findViewById(R.id.diminuir)  
22         aumentar.setOnClickListener(this)  
23         diminuir.setOnClickListener(this)  
24         texto = findViewById(R.id.texto)  
25     }  
26  
27     override fun onClick(botao: View?) {  
28         if (botao == aumentar) {  
29             tamanho += 1  
30             texto.textScaleX += 1  
31         } else if (botao == diminuir) {  
32             tamanho -= 1  
33             texto.textScaleX -= 1  
34         }  
35         texto.text = "$tamanho"  
36     }  
37 }
```

# No código em Kotlin

# Outra possibilidade...

```
17 ⚡ override fun onCreate(savedInstanceState: Bundle?) {
18     super.onCreate(savedInstanceState)
19     setContentView(R.layout.activity_main)
20     aumentar = findViewById(R.id.aumentar)
21     diminuir = findViewById(R.id.diminuir)
22     aumentar.setOnClickListener(this)
23     texto = findViewById(R.id.texto)
24
25     diminuir.setOnClickListener { it: View!
26         tamanho -= 1
27         texto.textScaleX -= 1
28         texto.text = "$tamanho"
29     }
30 }
31
32 ⓘ override fun onClick(botao: View?) {
33     if (botao == aumentar) {
34         tamanho += 1
35         texto.textScaleX += 1
36         texto.text = "$tamanho"
37     }
38 }
```

# Nullability

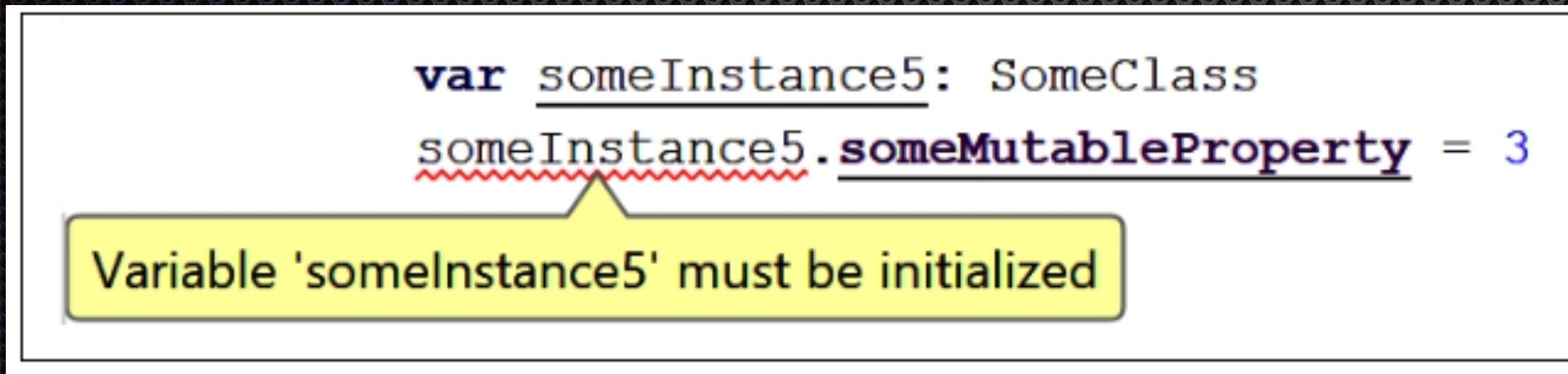
# val x var

- Temos duas opções quando declaramos uma variável em Kotlin
- Quando declaramos com **val**, não podemos atribuir um novo valor à variável.
- Mas podemos modificar atributos de um objeto que tenha sido declarado como **var**, mesmo que a variável que referencie o objeto seja **val**.

# Objetos Null

```
Aluno x;  
x.setNome("João");
```

- Um dos erros mais comuns em java é o acesso a objetos não inicializados (resulta em NullPointerException).
- Em kotlin, para não ter que ficar fazendo testes para verificar se um objeto é nulo, podemos usar os operadores ? e !!



- Para solucionar o problema acima, podemos garantir que o objeto seja inicializado com nulo com o operador ?: `var someInstance5: SomeClass? = null`
- Podemos usar diretamente antes de acessar um atributo para inicializá-lo com null, evitando o erro de objeto nulo: `val someInt = someInstance5?.someImmutableProperty`

# Objetos Null

- E também podemos usar a assertiva !! para garantir que o objeto não é nulo.

```
val someBoolean = true
if(someBoolean) {
    someInstance5 = someInstance
}

someInstance5!!.someMutableProperty = 3
```

- Dê preferencia à usar o operador ? ao invés do !! pois ele não termina abruptamente a aplicação em caso de erro.

<https://medium.com/luzalabs/kotlin-vari%C3%A1veis-e-nullable-8dd977bc376d>