

# Kotlin

Checagem de Tipos, Casting e Null Safety



---

---

---


---

---

---

---

## Tópicos Abordados



- Kotlin e o *null*
- Nullable Types
- Acesso em Nullable Types
- Smart Casts
- O operador *as*
- O operador *as?*

---

---

---


---

---

---

---

## Kotlin e o *null*



- A linguagem Kotlin foi criada pensando em null safety
  - Evitar problemas com *NullPointerException*
- Nenhum tipo de dado em Kotlin aceita o valor null
  - A não ser que você permita isso de forma explícita

```
var s: String  
s = null;
```

→ Erro de compilação!

---

---

---


---

---

---

---

## Nullable Types



- Para que a variável aceite a atribuição *null*, o tipo deve ter um “?” no final

```
var s: String?
s = null;
```

Non-Null Type	Nullable Type
Int	Int?
Double	Double?
String	String?
Boolean	Boolean?
Character	Character?
MyClass	MyClass?
...	...

---

---

---

---


---

---

---

---

## Acesso em Nullable Types



- Você só pode acessar properties ou métodos em nullable types se usar uma das técnicas abaixo

Testar a variável

```
var s: String? = null
if (s != null) s = s.toUpperCase()
```

Usar a invocação segura (?.)

```
var s: String? = null
s = s?.toUpperCase()
```

→ s = null

Usar o operador Elvis (?:)

```
var s: String? = null
s = s?.toUpperCase() ?: ""
```

→ s = ""

---

---

---

---


---

---

---

---

## Acesso em Nullable Types



Usar o operador !!

```
var s: String? = null
s = s!!.toUpperCase()
```

→ NullPointerException

---

---

---

---

---

---

---

---

## Smart Casts



- O Kotlin usa o operador *is* para dar suporte ao recurso de *smart cast*

```
fun check(p: Any) {  
    if (p is String) {  
        print(p.toUpperCase())  
    } else if (p is Int) {  
        print(p * 2)  
    } else {  
        print("Outro tipo")  
    }  
}
```

Na linha seguinte, o compilador sabe o tipo do dado

O operador *is* checa se o tipo é de determinada classe

## Smart Casts



```
fun check(p: Any) {  
    if (p is String && p.length == 0) {  
        print("String vazia")  
    }  
}
```

O compilador sabe que é uma String

## O operador *as*



- Usado para forçar o casting
- Pode lançar uma exceção do tipo *ClassCastException* se a conversão de tipo não puder ser realizada

```
var obj: Any = "abc"  
var s = obj as String
```

Converte o tipo de *Any* para *String*

```
var obj: Any = "abc"  
var i = obj as Int
```

*ClassCastException!*

## O operador *as?*



- Funciona da mesma forma que o *as*, mas não lança exceção
  - Ele retorna *null*

```
var obj: Any = "abc"  
var i = obj as? Int
```

→ Converte o tipo *Int?* e atribui *null*

---

---

---

---

---

---

---



---

---

---

---

---

---

---