



# Kotlin

Rafael Vieira Coelho  
[rafaelvc2@gmail.com](mailto:rafaelvc2@gmail.com)

<https://kotlinlang.org>

# Exceções em Kotlin

---

- Não existe diferenciação entre checked e unchecked exceptions.
- Kotlin não tem checked exceptions para reduzir a quantidade de código (try-catch).
- Para indicar em Kotlin que o código pode gerar uma exceção, usamos a notação `@Throws`.

# Lançando uma Exceção (throw)

- Aqui lançamos uma exceção caso o número não esteja no intervalo entre 0 e 100.

```
val percentage =  
    if (number in 0..100)  
        number  
    else  
        throw IllegalArgumentException(  
            "A percentage value must be"  
            +  
            "between 0 and 100: $number")
```

# Capturando uma Exceção Lançada (catch)

---

- O try aqui está sendo usado como uma expressão, o que não era permitido em Java.

```
val number = try {  
    Integer.parseInt(string)  
} catch (e: NumberFormatException) {  
    return  
}
```

# Anotação @Throws

- A notação @Throws indica que quem for chamar esta função, deve tratar a exceção IOException.

```
@Throws(IOException::class)
fun foo() {
    throw IOException()
}
```

# Vejamos Esta Comparaçāo entre Java e Kotlin

```
fun foo() { throw IOException() }
```

```
// Java
try {
    DemoKt.foo();
}
catch (IOException e) {
    // ...
}
```

```
@Throws(IOException::class)
fun bar() { throw IOException() }
```

```
// Java
try {
    DemoKt.bar();
}
catch (IOException e) {
    // ...
}
```

# Pergunta: Existe alguma diferença entre chamar foo e bar em Java?

- Não tem diferença
- O código que chama foo compila, mas o que chama bar não.
- O código que chama foo não compila, mas o que chama bar sim.

```
fun foo() { throw IOException() }
```

```
1 // Java
2 try {
3     DemoKt.foo();
4 }
5 catch (IOException e) {
6     // ...
7 }
```

```
1 // Java
2 try {
3     DemoKt.bar();
4 }
5 catch (IOException e) {
6     // ...
7 }
```

```
@Throws(IOException::class)
fun bar() { throw IOException() }
```

# Pergunta: Existe alguma diferença entre chamar foo e bar em Java?

- Não tem diferença
- O código que chama foo compila, mas o que chama bar não.
- O código que chama foo não compila, mas o que chama bar sim.

```
fun foo() { throw IOException() }
```

```
1 // Java
2 try {
3     DemoKt.foo();
4 }
5 catch (IOException e) {
6     // ...
7 }
```

```
1 // Java
2 try {
3     DemoKt.bar();
4 }
5 catch (IOException e) {
6     // ...
7 }
```

Java

```
@Throws(IOException::class)
fun bar() { throw IOException() }
```



# O código que chama foo não compila, mas o que chama bar sim!

The screenshot shows an IDE interface with two files open:

- Ex1NotacaoK.kt**:

```
import java.io.IOException

fun foo() {
    throw IOException()
}

@Throws(IOException::class)
fun bar() {
    throw IOException()
}
```
- Ex1NotacaoJ.java**:

```
import java.io.IOException;

public class Ex1NotacaoJ {
    public static void main(String[] args) {
        try {
            Ex1NotacaoKKt.bar();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    try {
        Ex1NotacaoKKt.foo();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

A tooltip at the bottom left of the IDE window states: "Exception 'java.io.IOException' is never thrown in the corresponding try block".

The Java code contains a redundant catch block for `IOException` in the first `try` block, as the exception is never thrown from the `bar()` method.

# O Tipo nothing

- O tipo de uma expressão throw em koltin é um tipo especial chamado Nothing

The screenshot shows an IDE interface with two main panes. The left pane displays the code file `Ex2Nothing.kt`. The right pane shows the run output.

**Code (Ex2Nothing.kt):**

```
1 data class Person(val name: String = "")  
2  
3     fun fail(message: String): Nothing {  
4         throw IllegalArgumentException(message)  
5     }  
6  
7     fun main() {  
8         val p = Person()  
9  
10        if (p.name != "")  
11            throw IllegalArgumentException("Name required 1")  
12        else  
13            fail("Name required 2")  
14    }  
15}
```

**Run Output:**

```
Exception in thread "main" java.lang.IllegalArgumentException: Name required 2  
at Ex2NothingKt.fail(Ex2Nothing.kt:5)  
at Ex2NothingKt.main(Ex2Nothing.kt:14)  
at Ex2NothingKt.main(Ex2Nothing.kt)  
Process finished with exit code 1
```

# O Tipo nothing

The screenshot shows an IDE interface with a code editor and a run output window.

**Code Editor (Ex2Nothing.kt):**

```
1  data class Person(val name: String = "")  
2  
3  fun fail(message: String): Nothing {  
4      throw IllegalArgumentException(message)  
5  }  
6  
7  fun main() {  
8      val p = Person()  
9  
10     if (p.name == "")  
11         throw IllegalArgumentException("Name required 1")  
12     else  
13         fail("Name required 2")  
14 }  
15
```

**Run Output:**

Run: Ex2NothingKt

```
/Applications/IntelliJ IDEA CE.app/Contents/jbr/Contents/Home/bin/java" -javaagent  
Exception in thread "main" java.lang.IllegalArgumentException: Name required 1  
    at Ex2NothingKt.main(Ex2Nothing.kt:12)  
    at Ex2NothingKt.main(Ex2Nothing.kt)  
Process finished with exit code 1
```



[\*\*https://github.com/rafael-vieira-coelho/kotlin/tree/Aula-08---  
Exceções\*\*](https://github.com/rafael-vieira-coelho/kotlin/tree/Aula-08---Exceções)

---

[rafaelvc2@gmail.com](mailto:rafaelvc2@gmail.com)