Convenções de código

Motivação

Convenções de código são um conjunto de recomendações para definir o estilo de codificação usado em métodos, variáveis, nome de classes, organização e nomenclatura de arquivos, comentários e outros recursos da linguagem. Convenciona-se para que a leitura seja mais fácil ao se deparar com um código desconhecido e também para que não haja diferentes estilos de programação dentro de um projeto, uma vez que frequentemente projetos são executados por diversas pessoas.

Idioma

O idioma será inglês, uma vez que os métodos por padrão compartilhados em repositórios, palavras reservadas e documentações oficiais da linguagem são inglês. Dessa maneira, não existirão métodos e documentações em diferentes línguas. No entanto, a linguagem dos métodos não interferem no entendimento dos conceitos. A qualquer momento, é possível trocar o nome dos métodos caso necessário.

Os comentários serão português para compreensão mais clara já que se trata de um curso destinado para falantes de português.

Para nome de variáveis, funções, atributos, classes e qualquer código que não seja comentário, nunca usa-se acentuação.

Ponto e vírgula

A linguagem deixa como opcional o uso do ponto e vírgula e dessa maneira, como Kotlin preza pela limpeza do código, o uso do ponto e vírgula é desencorajado. Caso seja declarado na IDE, não haverá problema de compilação, mas haverá o aviso constante de que não é necessário.

Linguagem Kotlin

Nomenclaturas

Uso de **camelCase** começando com letra minúscula para:

- Variáveis
- Métodos
- Atributos de classe
- Parâmetros

Para constantes usando a palavra reservada **const** ou variáveis definidas com **val** fora de classes, devem usar todas as letras maiúsculas separadas com underline.

```
const val ITEM_KEY = "TYX"
val MAX_ITEMS: Int = 8

class Course (val rating: Float) {
   private val courseName = "Kotlin"

   fun isThisCourseAwesome(isAwesome: Boolean) {
       if (isAwesome) {
           println("Com certeza! Estamos falando de $courseName.")
       }
   }
}
```

Classes

Se um arquivo Kotlin possuir uma classe somente, o nome do arquivo deve ser o mesmo nome da classe com a extensão .kt. Se um arquivo contiver múltiplas classes, o ideal é que o nome do arquivo seja algo que descreva a responsabilidade do arquivo da melhor maneira e mais sucinta possível.

Para a nomenclatura, usar o padrão **PascalCase** com a primeira letra sempre maiúscula.

Os parâmetros construtores das classes, caso sejam poucos, podem ser escritas em uma única linha. Classes com construtores mais longos devem ser formatadas para que cada propriedade fique em uma linha. Além disso, os parênteses para fechar o construtor deve ficar em uma nova linha.

```
class Person(val name: String)

class Course(
   val title: String,
   val students: Int,
   val rating: Float
)
```

Dentro de uma classe, ordenar o conteúdo preferencialmente:

- Variáveis de escopo da classe e inicializações
- Construtores secundários
- Objects e Companion objects
- Override
- Métodos com sobrecarga

Chaves

Em if, else, while, for, do..while, try..catch, funções e classes sempre utilizar as chaves para abertura e fechamento do corpo das instruções.

```
if (condicao) {
} else {
}
```

```
while (condicao) {
}
```

```
try {
} catch (e: Exception) {
} finally {
}
```

```
for (l in listOf(1, 2)) {
}
```

Unit

O tipo Unit corresponde a nulo. Em funções que não há retorno, não há necessidade de declarar o retorno como Unit, basta deixar em branco. Caso a função possua retorno, então é necessário declarar o tipo.

```
fun noReturn() {
    println("Sem retorno")
}

fun sayMyName(name: String): String {
    return "Hello, $name"
}
```

Activities e Layout

Nomes de arquivos de layout devem ser prefixados com os elementos que representam:

Componente	*Classe*	*Nome do layout*
Activity	UserProfileActivity	activity_user_profile
Fragment	SignUpFragment	fragment_sign_up
Dialog	ChagePasswordDialog	dialog_change_password

Arquivos XML

Quando um elemento não tiver nenhum conteúdo dentro das tags, deve-se usar **self closing** tags.

```
<TextView android:textSize="32sp" />
```

Identificadores de elementos

Identificadores de elementos são escritos em minuscula_com_underline.

Elemento	*Prefixo*
TextView	text_
EditText	edit_
Button	button_
ImageView	image_
Menu	menu_

```
<TextView android:id="@+id/text_name" />

<EditText android:id="@+id/edit_cupom_price" />

<ImageView android:id="@+id/image_pets" />
```