



# KOTLIN

Disciplina: Estruturas de Linguagens  
Professor: Francisco Figueiredo G. Sant'anna  
Dupla: Matheus Rocha (201810037911)  
Matheus Ignácio (201810035211)



# Origens e influências

Com seu desenvolvimento iniciado em 2010 pela JetBrains, Kotlin é uma linguagem de programação *open-source* influenciada por:

- Java
- ML
- C#
- Groovy
- JavaScript
- Scala





# Classificação

Kotlin é classificada como uma linguagem:

- Imperativa
- Orientada a objetos
- Funcional
- Compilada
- Tipagem Estática



# Áreas de uso

Kotlin é utilizada em várias áreas, as principais sendo:

- Desenvolvimento de aplicações Android;
- Desenvolvimento de aplicações multiplataforma;
- Desenvolvimento Web (*client* e *server-side*);
- Kotlin for Native;
- Kotlin for Data Science.




# Descrição da funcionalidade escolhida

- Extensions:

- Permite que você estenda uma classe com novas funções e/ou propriedades;
- Não é necessário se trabalhar com herança;
- Melhora a legibilidade do código.

```
fun MutableList<Int>.swap(index1:
Int, index2: Int) {
    val tmp = this[index1]
    this[index1] = this[index2]
    this[index2] = tmp
}
```



```
class Pessoa{
    var nome : String? = null

    fun mostre(){
        println(nome)
    }
}
```

```
fun main() {
    var p1 = Pessoa()
    p1.nome = "Lia"

    var p2 = Pessoa()
    p2.nome= "Carla"

    var p3=p1.composto(p2)
}
```

```
fun Pessoa.Composto(p : Pessoa) : Pessoa {
    var ptemp = Pessoa()
    ptemp.nome = this.nome + " " + p.nome
    return ptemp
}
```




# Exemplo de uso real

Sem o uso de *Extensions*:

```
Picasso.with(imageView.context).load(url).into(imageView)
```

Com o uso de *Extensions*:

```
fun ImageView.loadUrl(url: String) {  
    Picasso.with(context).load(url).into(this)  
}  
imageView.loadUrl(url)
```



```
...
internal fun String.toEditable(): Editable =
    Editable.Factory.getInstance().newEditable(this)

internal fun Activity.attachFragment(manager: FragmentManager, containerId: Int,
view: Fragment, tag: String) {
    manager.beginTransaction()
        .replace(containerId, view, tag)
        .commitNowAllowingStateLoss()
}

internal fun Fragment.getCalendarTime(): String {
    val cal = Calendar.getInstance(TimeZone.getDefault())
    val format = SimpleDateFormat("d MMM yyyy HH:mm:ss Z")
    format.timeZone = cal.timeZone
    return format.format(cal.time)
}

internal fun Fragment.makeToast(value: String) {
    Toast.makeText(activity, value, Toast.LENGTH_SHORT).show()
}
...
```





# Kotlin vs Java

```
fun MutableList<String>.swap(index1: Int, index2: Int) {  
    val tmp = this[index1]  
    this[index1] = this[index2]  
    this[index2] = tmp  
}  
  
fun main() {  
    val list =  
mutableListOf("Estruturas", "De",  
"Linguagem", "EDL")  
    list.swap(0, 2)  
}
```

```
import java.util.*;  
  
public static void main(String[] args) {  
    ArrayList<String> mylist =  
        new  
ArrayList<String>();  
    mylist.add("Estruturas");  
    mylist.add("De");  
    mylist.add("Linguagens");  
    mylist.add("EDL");  
  
    Collections.swap(mylist, 1, 2);  
    Collections.swap(mylist, 3, 1);  
}
```



## O uso de *Extensions*...

- Permitiu que o código ficasse mais fácil de ler e mais compacto, facilitando seu entendimento;
- Chamada de função mais limpa, com menos argumentos;
- Evitou a importação de uma biblioteca;



# Fontes

- <https://antonioleiva.com/extension-functions-kotlin/>
- <https://github.com/BracketCove/SpaceNotes/blob/master/app/src/main/java/com/wiseassblog/spacenotes/common/AndroidExt.kt>
- <https://www.geeksforgeeks.org/swapping-items-list-java-collections-swap/>
- <https://www.geeksforgeeks.org/kotlin-mutablelistof/>
- <https://kotlinlang.org/docs/reference/extensions.html>