Criando Funções Anônimas de Forma Simples (Lambdas em Kotlin)

Com toda certeza Java é uma excelente linguagem de programação.

Porém não importa o quanto você ame a linguagem, você precisa admitir que boa parte do código que você escreve diariamente não está relacionado ao problema que queremos resolver, mas sim à sintaxe da linguagem.

Veja abaixo um exemplo de código:

```
button.setOnClickListener(new View.OnClickListener() {
    public void onClick(View view){
        Toast.make(context,
                "Boilerplate Alarm", Toast.LENGTH_LONG).show()
    }
}
```

Todo esse código para que uma simples mensagem seja exibida para o usuário é um tanto verboso, você não acha?

Lambdas

Kotlin permite o uso de lambdas, que são funções anônimas bem mais versáteis, limpas e fáceis de serem utilizadas. Veja como seria a chamada do código acima usando Kotlin:

```
//toast é uma Extension function

button.setOnClickListener { toast("Bye boilerplate") }
```

Apenas uma linha, problema resolvido! E o melhor, onde quer que seja necessário passar como parâmetro uma interface que contém apenas um método pode ser substituído por uma lambda. Você também pode de tudo definir as suas próprias lambdas, observe:

```
 val contemCaracteresSuficientes: (String) -> Boolean
= { nome -> nome.length >= 6 }contemCaracteresSuficientes("Fulano") //Retorna true
```

No exemplo acima definimos uma lambda que recebe uma *String* e retorna um *Boolean,* onde *o* objetivo é simplesmente validar se a quantidade de caracteres da String é maior ou igual a 6. Uma vez que a lambda foi associada à constante *contemCaracteresSuficientes*, é possível reutilizar a funcionalidade quantas vezes for necessário.

Esse código ainda poderia ser optimizado, uma vez que em Kotlin, quando uma lambda recebe apenas um parâmetro, podemos omitir seu nome e utilizar a palavra reservada ***it*** para referenciar o valor:

```
val contemCaracteresSuficientes: (String) -> Boolean = { it.length >= 6 }
```

Por Hoje é só pessoal, sucesso nos estudos.