Kotlin – Para que serve o Data Class?

A ideia do Data Class, além de manter o mesmo comportamento de uma classe comum, é nos oferecer funções pré implementadas baseando-se nas properties do construtor primário.

o Data Class nos entrega as seguintes funções:

•equals()/hashCode(): para comparação de objetos;
•toString(): impressão padrão no formato “Pessoa(nome=Fulano de Tal, idade=21)”;
•componentN(): para permitir o uso do Destructuring Declaration;
•copy(): para copiar o objeto com flexibilidade.

Aproveitando a menção dessas funções, vamos explorar como cada uma delas funciona.

**Comparando os objetos**

Para testarmos o comportamento de comparação, vamos criar um clone do Fulano de Tal e realizar um print:

```
val fulano = Pessoa("Fulano de Tal", 21)
val cloneDoFulano = Pessoa("Fulano de Tal", 21)
print(fulano == cloneDoFulano)
```

O resultado desse código é true, portanto, a comparação entre os objetos está sendo feita a partir das suas properties!

**Imprimindo os objetos do Data Class**

Agora vamos ver o toString() da Ciclana:

```
val ciclana = Pessoa()
ciclana.nome = "Ciclana da Silva Sauro"
ciclana.idade = 28
print(ciclana)
```

Ao testar o código temos o seguinte resultado:

Pessoa(nome=Ciclana da Silva Sauro, idade=28)

Pegando as properties em variáveis separadas

Com a implementação do componentN() somos capazes de declarar variáveis a partir das properties do objeto conforme a proposta do Destructuring Declaration:

```
val (nome, idade) = ciclana
```

Isso mesmo, as variáveis, nome e idade são uma cópia das properties do objeto ciclana.

**Copiando objetos**

Se a Ciclana tiver uma irmã gêmea podemos copiar a idade da Ciclana e modificar apenas apenas o nome para o da irmã gêmea dela:

```
val ciclana = Pessoa()
ciclana.nome = "Ciclana da Silva Sauro"
ciclana.idade = 28
val beltrana = ciclana.copy("Beltrana da Silva Sauro")
print(beltrana)
```

Ao testar o código temos o seguinte resultado:

```
Pessoa(nome=Beltrana da Silva Sauro, idade=28)
```

A conclusão que chegamos é que o Data Class é uma alternativa para criarmos uma classe, mas, como vimos, ele só é necessário caso uma das suas implementações padrões seja necessária para você. Portanto, não o utilize caso você não tenha a intenção de usar alguma de suas implementações padrões.

É válido lembrar que ambas funções, funcionam apenas para properties do construtor primário, ou seja, as demais properties da classe, como por exemplo no corpo da mesma, não serão consideradas.

Por hoje é só pessoal, sucesso nos estudos!