08

# Python OO vs Java OO

**Transcrição**

Avançamos bastante no conteúdo do curso, mas vale ressaltar que o paradigma OO não é uma exclusividade da linguagem Python. Orientação a Objetos é um dos paradigmas mais utilizados entre as linguagens de programação.

Existem linguagens que continuam sendo procedurais, como linguagem C, assim como outros paradigmas funcionais. Inclusive, em alguns casos, os dois começam a misturar.

É possível afirmar que o paradigma OO domina o mercado de desenvolvimento.

Isto significa que se você tem uma boa base de OO vista no curso de Python, também já terá aprendido sobre Java, PHP, C++, além de outras linguagens que seguem o mesmo paradigma.

Podemos perceber isso, comparando o arquivo `conta.py` e o `Conta.java`. A diferença entre os dois são os detalhes da sintaxe, mas o paradigma é o mesmo. Por exemplo, os dois terão uma classe `Conta`, que em Java está assim:

```
class Conta {

    //atributos
    private int numero;
    private String titular;
    private double saldo;
    private double limite;

    //construtor
    Conta(int numero, String titular, double saldo, double limite) {
        this.numero = numero;
        this.titular = titular;
        this.saldo = saldo;
        this.limite = limite;
    }
}
```

Enquanto a classe no Python está:

```
class Conta:

    def __init__(self, numero, titular, saldo, limite):
        print("Construindo objeto ... {}".format(self))
        self.__numero = numero
        self.__titular = titular
        self.__saldo = saldo
        self.__limite = limite
```

A diferença na sintaxe é que o primeiro usa chaves ( `{}` ), enquanto o segundo usa dois pontos ( `:` ). No Python, nós começamos com a função `__init__`, que é a função construtora. No entanto, no Java, não basta apenas adicionarmos o construtor, temos que definir os atributos antes.

Mas nas duas sintaxes temos uma função ocupando o papel de construtor. No Python, ela vai receber o nome de `__init__`, no Java, ela terá o mesmo nome da classe.

Quem tem conhecimentos sobre Java, sabe que o seu construtor também se chama `<init>`.

No entanto, a principal diferença é a parte superior no arquivo `Conta.java`, em que precisamos definir os atributos explicitamente e especificamos que eles são privados. No `conta.python`, usamos a convenção `__` para fazer o mesmo. Tanto no Python, quanto no Java, existem formas de acessar um atributo, mesmo que definindo como privado.

Se seguirmos comparando os dois arquivos, veremos que ambos têm o método `extrato()`, mas o Java não receberá `self`.

```
//metodos
void extrato() {
    System.out.println("Saldo de " + this.saldo)
}
```

Mas também existe o `self` no mundo Java: `this`, que é disponibilizado implicitamente, mesmo não estando declarado. Veremos que o uso do `.` também é correspondente, assim como os métodos privados.

Vemos em `conta.py`, que definimos as propriedades para acessar o saldo. No mundo Java, escrevemos um método para cada ação, como foi feito no `getSaldo()` de `Conta.java` — que precisa do `getTitular()`, `getNumero()` e `getLimite()`.

```
public double getLimite() {
    return limite;
}

public void setLimite(double limite) {
    this.limite = limite;
}

public double getNumero() {
    return numero;
}

public void getTitular() {
    return titular;
}

public double getSaldo() {
    return saldo;
}
```

E da mesma forma como existem métodos estáticos no mundo Python, existe no mundo Java:

```
public static String codigo() {
    return "001"
}
```

Observe que usamos a palavra-chave `static`. Mesmo conhecendo apenas Python, conseguimos entender a lógica do código Java. Como criar um objeto baseado em uma conta Java?

```
Conta contaDoNico = new Conta(123, "Nico", 55.5, 1000.0);
contaDoNico.deposita(100.0);
```

Nós também usamos o construtor `Conta()`, passamos os parâmetros, mas adotamos a palavra `new`. O endereço fica guardado na referência `contaDoNico`, porém, declaramos também o tipo. Mais acima, definimos cada tipo dos atributos: `int`, `String`, `double`. Em Java, ou utilizamos a tipagem estática, ou definimos o tipo da variável.

Fizemos também a chamada usando a referência `contaDoNico` para passar a função com o valor.

Perceba que quando aprendemos sobre o paradigma Orientado a Objetos, podemos aplicar o conceito em diversas linguagens, porque ele é seguido da mesma forma, mudando apenas as sintaxes específicas de cada linguagem.