08

Para saber mais - Herança com construtor secundário

Ao utilizar a herança, comentei que existem outras maneiras de implementar além do construtor primário.

Considerando o exemplo da classe `Gerente`:

```
class Gerente(
    nome: String,
    cpf: String,
    salario: Double,
    val senha: Int
) : Funcionario(
    nome = nome,
    cpf = cpf,
    salario = salario
) {

    //métodos

}
```

Temos o seguinte resultado ao usar o construtor secundário:

```
class Gerente : Funcionario {

    val senha: Int

    constructor(
        nome: String,
        cpf: String,
        salario: Double,
        senha: Int
    ) : super(
        nome = nome,
        cpf = cpf,
        salario = salario
    ) {
        this.senha = senha
    }

    //métodos

}
```

Note que, após a assinatura do construtor, temos a chamada do `super()`, que indica o uso do construtor da classe mãe, assim como vimos na reutilização do método `bonificacao()`.

Também existe o caso que a classe base não tem construtor personalizado:

```
open class Animal {

}

class Gato : Animal() {

}
```

Ou via construtor secundário:

```
class Gato : Animal {

    constructor() : super()

}
```

Mesmo não tendo, a chamada do construtor padrão é necessária. Portanto, em toda herança é obrigatório o uso do construtor da classe mãe.