



Composição de objetos

Um **objeto X** pode ser atributo de **outro objeto Y**, esse conceito se chama **composição**. Em **Kotlin** não foge muito de como é em **Java**, apenas declarar a **classe X** como atributo da **classe Y** faz essa **composição**, sendo possível **acessar os atributos de X dentro de Y**:

```
abstract class Conta {
    val titular : Cliente,
    val numeroConta: String
}

var saldo = BigDecimal(0.0)
private set

fun deposita(valor: BigDecimal){
    this.saldo += valor
}

abstract fun saca(valor: BigDecimal)

class Cliente {
    val nome: String,
    val cpf : String,
    var endereco : Endereco = Endereco()
    private val senha: String
} : Autenticavel {

    override fun autentica(senha: String): Boolean {
        return when {
            this.senha != senha -> false
            else -> true
        }
    }
}

class Endereco {
    var logradouro: String = "",
    var numero: Int = 0,
    var bairro: String = "",
    var cidade: String = "",
    var estado: String = "",
    var cep: String = "",
    var complemento: String = ""
}
```

Rola uma **discussão** entre **usar Herança ou composição**. A composição não tem a capacidade do polimorfismo, porém ela evita o problema principal da herança que é herdar todo o legado da superclasse que talvez não fosse necessário.



Variável global e init

Não considero boa prática usar **variável global** no **Kotlin**, ela é bem parecida com **JavaScript**, onde a variável fica no arquivo **“separada” de funções ou classes**, é uma variável de arquivo mesmo.

Os objetos de **Kotlin** tem uma sintaxe para **executar algum código imediatamente após a criação do objeto**, se chama **“init”** (é uma prática pra evitar o construtor secundário):

```
var totalContas = 0
private set

abstract class Conta {
    val titular : Cliente,
    val numeroConta: String
}

{
    var saldo = BigDecimal(0.0)
    private set

    init {
        print("Criando conta")
        totalContas++
    }

    fun deposita(valor: BigDecimal){
        this.saldo += valor
    }

    abstract fun saca(valor: BigDecimal)
```