

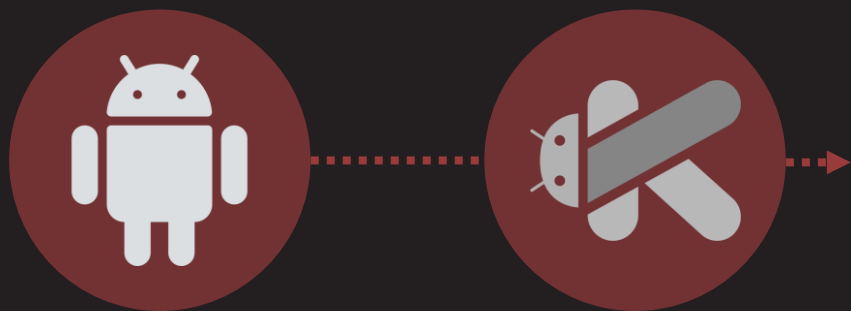
FIA/P GRADUAÇÃO

# Hybrid Mobile App Development

Prof. Andrey Masiero

#03 – Sobrecarga, Enum e NamedParameter

# Começando o caminho Jedi



# Enum

Classes de domínio

# Classificando as transações por tipo

- Criamos nossa classe de transação, mas até o momento, não sabemos qual o tipo de transação que temos.

```
class Transacao(val valor : BigDecimal,  
                val descricao : String,  
                val data : Calendar)
```

- Como podemos resolver essa situação?

# Classificando as transações por tipo

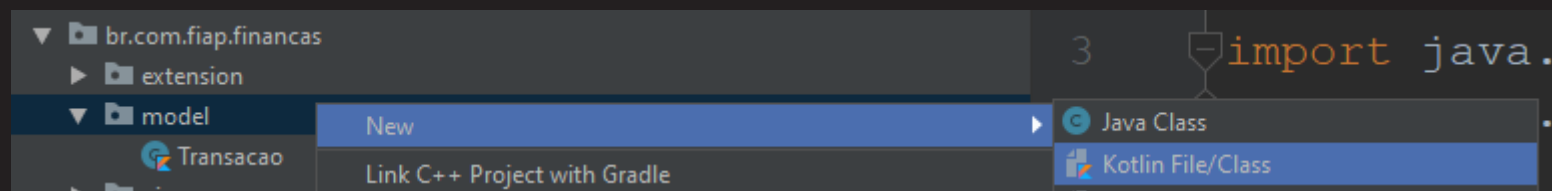
- Podemos criar uma propriedade booleana para representar receita em true e despesa em false.

```
class Transacao(val valor : BigDecimal,  
                val descricao : String,  
                val receita : Boolean,  
                val data : Calendar)
```

- Isso funciona bem, mas o problema é que não conseguimos uma semântica muito boa em nosso código.

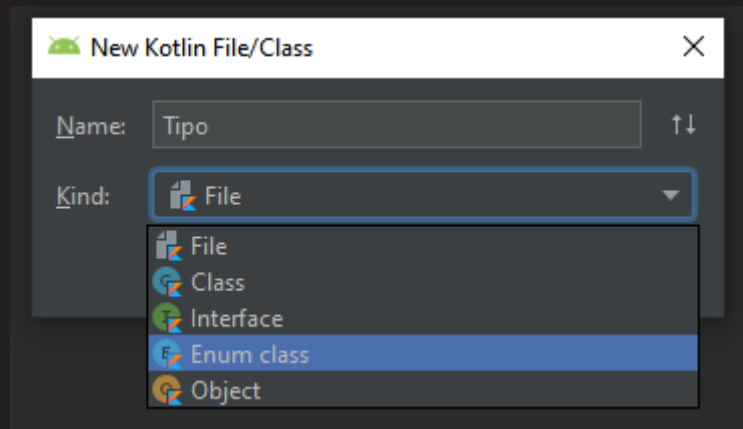
# Classificando as transações por tipo

- A forma mais apropriada para resolvermos isso é utilizando uma classe especial chamada de Enum.
- Enum ou enumerador é uma classe que define constantes para facilitar a categorização de algo e nosso código.
- Clique com botão direito no pacote model e escolha New → Kotlin File/Class:



# Classificando as transações por tipo

- Digite o nome da classe enum como Tipo e selecione Enum class, no combo Kind:



- Em seguida clique em OK.



# Classificando as transações por tipo

- Basta agora definir os valores com palavras em caixa alta e separadas por vírgula.

```
enum class Tipo {  
    RECEITA, DESPESA  
}
```

- Agora, basta vincularmos ao modelo de transação.

```
class Transacao(val valor : BigDecimal,  
                val descricao : String,  
                val tipo : Tipo,  
                val data : Calendar)
```

# Classificando as transações por tipo

- Por fim, basta colocarmos na instância da classe ao declarar os objetos.

```
val transacoes = listOf(  
    Transacao(BigDecimal( val: 20.5),  
        descricao: "Comida",  
        Tipo.DESPESA,  
        Calendar.getInstance()),  
    Transacao(BigDecimal( val: 100.0),  
        descricao: "Rendimento",  
        Tipo.RECEITA,  
        Calendar.getInstance())  
)
```

# Parâmetros com Valor Padrão

Melhorando os valores repetidos

# Parâmetros com Valor Padrão

- Perceba que nos dois objetos utilizamos sempre o `Calendar.getInstance()`.

```
val transacoes = listOf(  
    Transacao(BigDecimal( val: 20.5 ),  
        descricao: "Comida",  
        Tipo.DESPESA,  
        Calendar.getInstance()),  
    Transacao(BigDecimal( val: 100.0 ),  
        descricao: "Rendimento",  
        Tipo.RECEITA,  
        Calendar.getInstance())  
)
```

- Será que existe uma maneira melhor de trabalhar com esse tipo de situação?

# Parâmetros com Valor Padrão

- O Kotlin permite a nós definir um valor padrão para parâmetros de construtores e métodos. Veja como fica a declaração do construtor da transação.

```
class Transacao(val valor : BigDecimal,  
                val descricao : String,  
                val tipo : Tipo,  
                val data : Calendar = Calendar.getInstance())
```

# Sobrecarga de Construtores

Diversificando a instância do objeto

# Sobrecarga de Construtores

- O Kotlin, também nos permite criar modificações de construtores.

```
class Transacao(val valor : BigDecimal,  
               val descricao : String,  
               val tipo : Tipo,  
               val data : Calendar = Calendar.getInstance()) {  
  
    constructor(valor: BigDecimal, tipo: Tipo) : this(valor, descricao: "Indefinida.", tipo)  
}
```

- Contudo, isso dificulta um pouco a criação do objeto, pois nossa classe fica um pouco verbosa, com múltiplos construtores.

# Named Parameters

Dando nome aos bois



# Named Parameters

- Para evitar a criação de construtores, podemos utilizar os Named Parameters, que nada mais é que informar para qual parâmetro o valor informado deve ser endereçado.

```
val transacoes = listOf(  
    Transacao(valor = BigDecimal( val: 20.5),  
        descricao = "Comida",  
        tipo = Tipo.DESPESA),  
    Transacao(valor = BigDecimal( val: 100.0),  
        tipo = Tipo.RECEITA,  
        descricao = "Rendimento")  
)
```

- Assim, não importa mais a ordem dos parâmetros dentro do mecanismo de instância do construtor.

# Exercícios

Hora do descanso.

# Exercícios

- Crie uma App de uma única activity que seja capaz de criar uma lista de tarefas. Utilize para esse exercício a linguagem Kotlin.
- Para auxiliar no desenvolvimento do layout, utilize o seguinte livro (grátis):

<https://leanpub.com/google-android>




# Próximos Passos

O que veremos na próxima aula

# Na próxima aula...

- Formatações de Números
- String templates
- Refatorando o código (Boas Práticas)



**Copyright © 2020**  
**Prof. Andrey Masiero**

Todos direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem o consentimento formal, por escrito, do Professor (autor).

*Do or do not. There is no try – Mestre Yoda*