



Programação Orientada a Objetos 2022

Prof. Dr. José Erinaldo da Fonsêca

São Paulo - SP, Maio de 2022.

Pilares de POO



- ▶ Encapsulamento
- ▶ Herança
- ▶ Polimorfismo

Polimorfismo

Permite que
um mesmo nome
represente vários
comportamentos diferentes.

Polimorfismo

- ▶ A principal aplicação do **Polimorfismo** está na definição de regras e **padrões de projeto**
- ▶ Os dois principais modelos deste objetivo incluem **classes abstratas** que podem ser implementadas explicitamente (e utilizadas via **Herança**) ou implicitamente (através de **interfaces**)

Classes Abstratas

- ▶ São **modelos para outras classes**
- ▶ Não podem ser instanciadas
- ▶ Classes mais especializadas herdam sua implementação
- ▶ **<extends>**

Interfaces

- ▶ Interfaces são **padrões** definidos **através de especificações**
- ▶ Seus métodos são definidos, mas não implementados
- ▶ **<implements>**

Assinatura do método

- ▶ Todo método tem uma assinatura.
- ▶ É importante identificar quais métodos possuem a mesma assinatura.
- ▶ Ter a mesma assinatura, significa ter mesmas quantidades e os tipos de parâmetros.
- ▶ No exemplo ao lado, observe que os dois primeiros métodos possuem a mesma assinatura.

```
publico metodo calcMedia(n1: Real,  
                          n2: Real): Real  
  
publico metodo calcMedia(v1: Real,  
                          v2: Real):Inteiro  
  
publico metodo calcMedia(bim: Inteiro,  
                          n1: Real,  
                          n2: Real): Real  
  
publico metodo calcMedia(n1: Real,  
                          n2: Real,  
                          n3: Real,  
                          n4: Real): Real  
  
publico metodo calcMedia(medMin: Real,  
                          medMax: Real,  
                          sit: Caractere,  
                          bim: Inteiro)  
                          :Caractere
```

Tipos de Polimorfismo

- ▶ Sobreposição
 - ▶ Sobrecarga
- ▶ O tipo Sobreposição é mais utilizado que o tipo Sobrecarga.

Exemplo

- ▶ Relembrando “Polimorfismo de sobreposição”.
- ▶ Os métodos “emitirSom” possui assinaturas semelhantes.
- ▶ Os métodos “emitirSom” estão em classes diferentes.



Código...

- ▶ O método “emitirSom” na superClasse é cadastrado como abstrato. Logo, não pode ser instanciado.
- ▶ Veja que todas as classes possui o método “emitirSom”.
- ▶ Todos os métodos “emitirSom” necessita @Sobrepor o mesmo método na SuperClasse “Animal”.
- ▶ Todos os métodos possui a mesma assinatura e estão em classes diferentes.

```
classe abstrata Animal
    protegido peso: Real
    protegido idade: Inteiro
    protegido membros: Inteiro
    publico metodo abstrato emitirSom()
FimClasse
classe Mamifero estende Animal
    protegido corPelo: Caractere
    @Sobrepor
    publico metodo emitirSom()
        Escreva("som de Mamífero")
    fimMetodo
FimClasse
classe Lobo estende Mamifero
    @Sobrepor
    publico metodo emitirSom()
        Escreva("Auuuuuuuuuuuu!")
    fimMetodo
FimClasse
classe Cachorro estende Lobo
    @Sobrepor
    publico metodo emitirSom()
        Escreva("Au!Au!Au!")
    fimMetodo
FimClasse
```


Outro método....

- ▶ Suponha que o método “reagir()”, da classe cachorro, tenha comportamentos diferentes.
- ▶ Para tipo de ação o “reagir()” pode ser de forma diferente.
- ▶ Isso é também polimorfismo.



reagir()

falar frase	agradável: abanar e latir agressiva: rosnar
horário do dia	manhã: abanar tarde: abanar e latir noite: ignorar
dono	é dono: abanar não é: rosnar e latir
idade e peso	novo e leve: abanar novo e pesado: latir velho e leve: rosnar velho e pesado: ignorar

Outro método....

- ▶ O mesmo método “reagir()” possui assinaturas diferentes.
- ▶ Todos os métodos “reagir()” se encontra dentro da mesma classe “Cachorro”.
- ▶ Este é um exemplo de Polimorfismo de Sobrecarga.

```
classe Cachorro estende Lobo
publico metodo reagir(frase: Caractere)

fimMetodo
publico metodo reagir(hora, min: Inteiro)

fimMetodo
publico metodo reagir(dono: Logico)

fimMetodo
publico metodo reagir(idade: Inteiro,
                      peso: Real)

fimMetodo
FimClasse
```

- ▶ Todos os métodos “reagir()” possui a sua assinatura característica.
- ▶ Para cada ação, a classe “cachorro” tem uma reação com o método “reagir()”.
- ▶ Todos os métodos “reagir()” estão dentro da mesma classe.

```
classe Cachorro estende Lobo
    publico metodo reagir(frase: Caractere)
        se(frase="toma comida" ou frase="Olá")
            escreva("Abanar e Latir")
        senao
            escreva("Rosnar")
        fimSe
    fimMetodo
    publico metodo reagir(hora, min: Inteiro)
        se(hora<12)
            escreva("Abanar")
        senaoSe (hora>=18)
            escreva("Ignorar")
        senao
            escreva("Abanar e Latir")
        fimSe
    fimMetodo
    publico metodo reagir(dono: Logico)
        se(dono = verdadeiro)
            escreva("Abanar")
        sendo
            escreva("Rosnar e Latir")
        fimSe
    fimMetodo
FimClasse
```

- ▶ A classe “cachorro” estende a classe “Lobo”.
- ▶ O método “reagir()” está presente várias vezes dentro da mesma classe e com assinaturas diferentes.
- ▶ Isso é polimorfismo de Sobrecarga.

```
classe Cachorro estende Lobo
publico metodo reagir(idade: Inteiro,
                      peso: Real)
    se(idade<5)
        se(peso<10)
            escreva("Abanar")
        senao
            escreva("Latir")
        fimSe
    senao
        se(peso<10)
            escreva("Rosnar")
        senao
            escreva("Ignorar")
        fimSe
    fimSe
fimMetodo
FimClasse
```

Instanciando...

- ▶ Ao criar o objeto “Cachorro”, observe as diversas reações que podem acontecer.
- ▶ Polimorfismo de Sobrecarga, pois o mesmo método se encontra dentro da mesma classe e com diversas assinaturas.

```
// Programa Principal
c = novo Cachorro()

c.reagir("Olá")           // Abanar e Latir
c.reagir("Vai apanhar")  // Rosnar
c.reagir(11, 45)         // Abanar
c.reagir(21, 00)         // Ignorar
c.reagir(verdadeiro)     // Abanar
c.reagir(falso)          // Rosnar e Latir
c.reagir(2, 12.5)        // Latir
c.reagir(17, 4.5)       // Rosnar
```

Conclusão

- ▶ **Polimorfismo de Sobreposição**

Assinaturas iguais

Classes diferentes

- ▶ **Polimorfismo de Sobrecarga**

Assinaturas diferentes

Mesma classe.

Referências

- ▶ DEITEL, P.; DEITEL, H.; **Java: como programar**. 10ª ed. São Paulo: Pearson Education do Brasil, 2017. E-book
- ▶ SCHILDT, H.; **Java para iniciantes: crie, compile e execute programas Java rapidamente** [recurso eletrônico]. 6. ed. Porto Alegre: Bookman, 2015. E-book
- ▶ SEBESTA, R.W; **Conceitos de linguagem de programação** [recurso eletrônico]. 11ª ed. Porto Alegre: Bookman, 2018. E-book.
- ▶ Acessado em 10/02/2022. <https://www.cursoemvideo.com/curso/java-poo/>



FIM!!