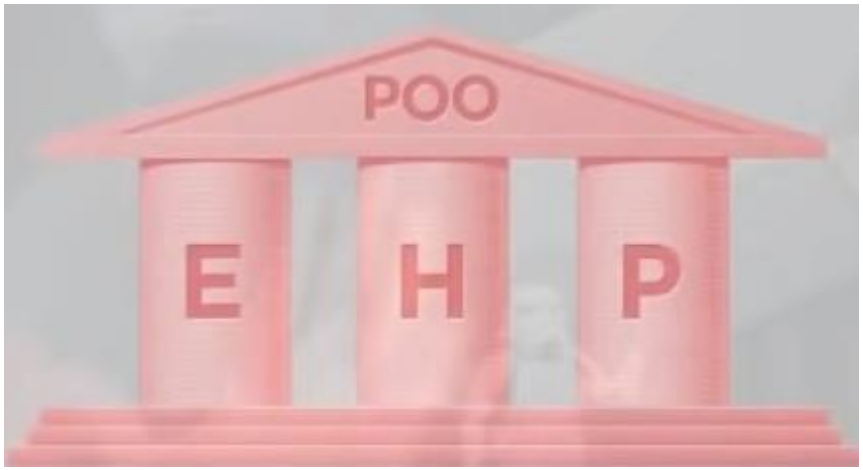
Programação Orientada a Objetos 2022

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.

Conceitos

Polimorfismo

- Habilidade de um objeto executar um método e obter o ***comportamento correto*** (ou desejado) em pontos apropriados do programa

Técnica de Prevenção de Erros

Padrões de Projeto

- Um padrão é uma descrição ou modelo a ser usado por uma equipe
- Padrões de Projeto facilitam a reutilização ao estabelecer um vocabulário comum no projeto

Refatoração

- Processo de alterar o código para melhorar sua estrutura sem alterar seu comportamento
- Significa tornar o código mais compreensível e bem estruturado
- Facilita o trabalho em equipe e a manutenção

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
```

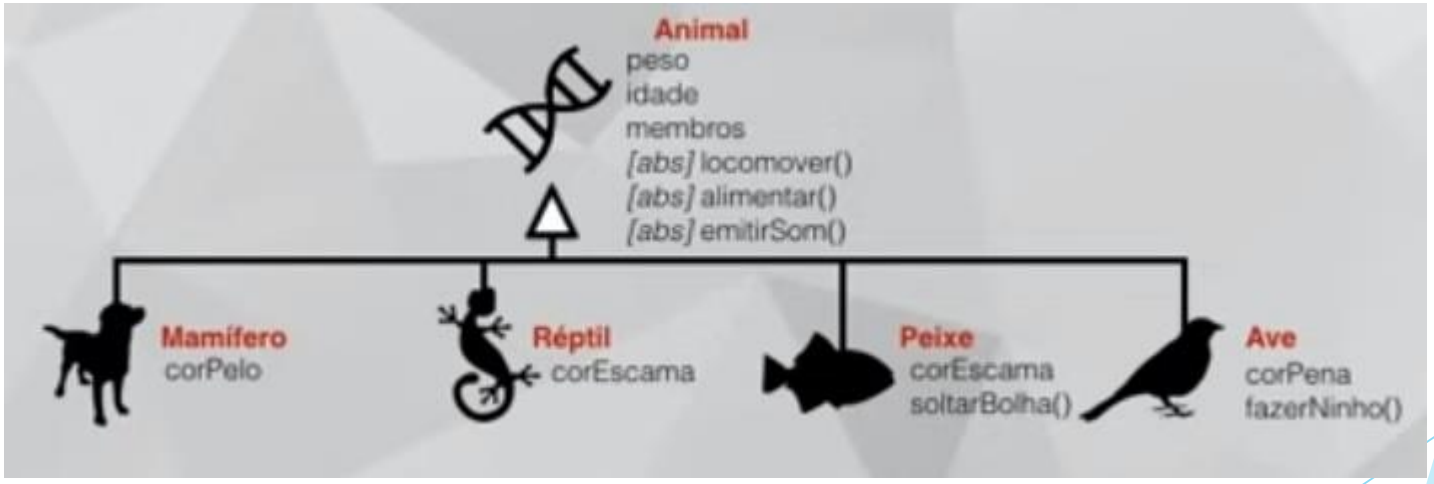
Tipos de Polimorfismo

- Sobreposição
- Sobrecarga

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

Exemplo

- Suponha a SuperClasse animal com suas SubClasses.
- Podemos executar o código da estutura abaixo.



- Inicialmente fazemos a SuperClasse Animal. Veja que ela é abstrata.

- Em seguida construímos a primeira subclasse Mamífero.

- É inserido a ideia de polimorfismo com a sobreposição. @sobrepor

```
classe abstrata Animal
   protegido peso: Real
   protegido idade: Inteiro
   protegido membros: Inteiro
   publico metodo abstrato locomover()
   publico metodo abstrato alimentar()
   publico metodo abstrato emitirSom()
FimClasse

classe Mamifero estende Animal
   privado corPelo: Caractere
   @Sobrepor
   publico metodo locomover()
      Escreva("Correndo")
   fimMetodo
   @Sobrepor
   publico metodo alimentar()
      Escreva("Mamando")
   fimMetodo
   @Sobrepor
   publico metodo emitirSom()
      Escreva("som de Mamífero")
   fimMetodo
FimClasse
```

- Foi mantida a SuperClasse Animal.
- A segunda subclasse é Reptil, que também estende a Classe Animal.
- É inserido a ideia de polimorfismo com a sobreposição. @sobrepor

```
classe abstrata Animal
  protegido peso: Real
  protegido idade: Inteiro
  protegido membros: Inteiro
  publico metodo abstrato locomover()
  publico metodo abstrato alimentar()
  publico metodo abstrato emitirSom()
FimClasse

classe Reptil estende Animal
  privado corEscama: Caractere
  @Sobrepor
  publico metodo locomover()
     Escreva("Rastejando")
  fimMetodo
  @Sobrepor
  publico metodo alimentar()
     Escreva("Comendo Vegetais")
  fimMetodo
  @Sobrepor
  publico metodo emitirSom()
     Escreva("som de Réptil")
  fimMetodo
FimClasse
```

- Foi mantida a SuperClasse Animal.
- A terceira subclasse é Peixe, que também estende a Classe Animal.
- Apresenta o método “soltarBolha”.
- É inserido a ideia de polimorfismo com a sobreposição. @sobrepor

```
classe abstrata Animal
   protegido peso: Real
   protegido idade: Inteiro
   protegido membros: Inteiro
   publico metodo abstrato locomover()
   publico metodo abstrato alimentar()
   publico metodo abstrato emitirSom()
FimClasse
classe Peixe estende Animal
   privado corEscama: Caractere
   @Sobrepor
   publico metodo locomover()
      Escreva(“Nadando”)
   fimMetodo
   @Sobrepor
   publico metodo alimentar()
      Escreva(“Comendo substâncias”)
   fimMetodo
   @Sobrepor
   publico metodo emitirSom()
      Escreva(“Peixe não faz som”)
   fimMetodo
   publico metodo soltarBolha()
      Escreva(“Soltou uma bolha”)
   fimMetodo
FimClasse
```

- Foi mantida a SuperClasse Animal.
- A quarta subclasse é Ave, que também estende a Classe Animal.
- Apresenta o método “fazerNinho”.
- É inserido a ideia de polimorfismo com a sobreposição. @sobrepor

```
classe abstrata Animal
  protegido peso: Real
  protegido idade: Inteiro
  protegido membros: Inteiro
  publico metodo abstrato locomover()
  publico metodo abstrato alimentar()
  publico metodo abstrato emitirSom()
FimClasse

classe Ave estende Animal
  privado corPena: Caractere
  @Sobrepor
  publico metodo locomover()
     Escreva(“Voando”)
  fimMetodo
  @Sobrepor
  publico metodo alimentar()
     Escreva(“Comendo frutas”)
  fimMetodo
  @Sobrepor
  publico metodo emitirSom()
     Escreva(“Som de ave”)
  fimMetodo
  publico metodo fazerNinho()
     Escreva(“Construiu um ninho”)
  fimMetodo
FimClasse
```

- Veja que na hora de instanciar os objetos, a primeira linha dar erro.
- Todas as demais classes podem ser instanciadas.
- Veja que apesar de termos o método locomover para todas as classes, a ação de cada um é diferente.
- A mesma ideia serve para os demais métodos.

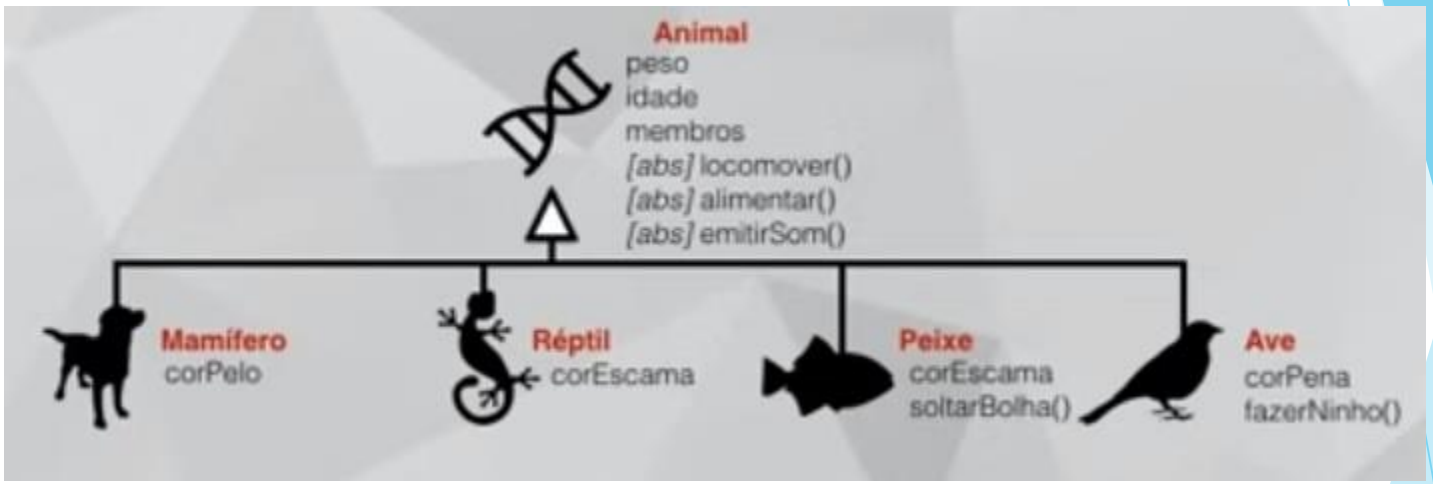
```
// Programa Principal
n = novo Animal()
m = novo Mamifero()
r = novo Reptil()
p = novo Peixe()
a = novo Ave()

m.setPeso(85.3)
m.setIdade(2)
m.setMembros(4)
m.locomover()  // Correndo
m.alimentar()  // Mamando
m.emitirSom()  // Som de Mamífero

p.setPeso(0.35)
p.setIdade(1)
p.setMembros(0)
p.locomover()  // Nadando
p.alimentar()  // Comendo substâncias
p.emitirSom()  // Peixe não faz som
p.soltarBolha()

a.setPeso(0.89)
a.setIdade(2)
a.setMembros(2)
a.locomover()  // Voando
a.alimentar()  // Comendo frutas
a.emitirSom()  // Som de Ave
a.fazerNinho()
```

Polimorfismo

- Isso é polimorfismo: o mesmo método faz coisas diferentes.

- Polimorfismo de sobreposição: acontece quando substituímos um método de uma superClasse na sua subClasse, usando a mesma assinatura.

- Suponha agora criar novas subclasses.
- Essas novas subclasses também tem atributos e métodos.
- Como programar essas novas classes ?



- Veja que as classes “Canguru” e “Cachorro” estendem a classe “Mamífero”.

- O método “locomover()” em “Canguru” sobrepõe o mesmo método em “Mamífero”.

```
classe Canguru estende Mamifero
   publico metodo usarBolsa()
      Escreva(“Usando bolsa”)
   fimMetodo
   @Sobrepor
   publico metodo locomover()
      Escreva(“Saltando”)
   fimMetodo
FimClasse


classe Cachorro estende Mamifero
   publico metodo enterrarOsso()
      Escreva(“Enterrando Osso”)
   fimMetodo
   publico metodo abanarRabo()
      Escreva(“Abanando o Rabo”)
   fimMetodo
FimClasse
```

- As classes “Cobra”, “GoldFish” e “Arara” não tem nada a ser implementado.

- A classe “Tartaruga” tem o método “locomover” que sobrepõe o mesmo método em “Réptil”.

```
classe Cobra estende Reptil

FimClasse

classe Tartaruga estende Reptil
  @Sobrepor
  publico metodo locomover()
     Escreva(“Andando beeeeem devagar”)
  fimMetodo
FimClasse

classe GoldFish estende Peixe

FimClasse

classe Arara estende Ave

FimClasse
```

- Veja que no programa principal os objetos são instanciados.
- Alguns métodos são sobrepostos devido a ideia do polimorfismo.
- Classes abstratas não podem ser instanciadas.

```
// Programa Principal
m = novo Mamifero()
c = novo Canguru()
k = novo Cachorro()

m.setPeso(5.70)
m.setIdade(8)
m.setMembros(4)
m.locomover()  // Correndo
m.alimentar()  // Mamando
m.emitirSom()  // Som de Mamífero

c.setPeso(55.30)
c.setIdade(3)
c.setMembros(4)
c.locomover()  // Saltando
c.alimentar()  // Mamando
c.emitirSom()  // Som de Mamífero
c.usarBolsa()

k.setPeso(3.94)
k.setIdade(5)
k.setMembros(4)
k.locomover()  // Correndo
k.alimentar()  // Mamando
k.emitirSom()  // Som de Mamífero
k.abanarRabo()
```

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!!