Polimorfismo

Programação Orientada a Objetos

Prof. Jônata Tyska Carvalho

Polimorfismo: Agenda

1. Introdução
2. Amarração estática e dinâmica
3. Exemplos
4. Herança Múltipla
5. Resumo

Polimorfismo: Agenda

1. **Introdução**
2. Amarração estática e dinâmica
3. Exemplos
4. Herança Múltipla
5. Resumo

Polimorfismo: Agenda

1. **Introdução**
2. Amarração estática e dinâmica
3. Exemplos
4. Herança Múltipla
5. Resumo

Polimorfismo: Introdução

- Abstração
- Encapsulamento
- Herança
- **Polimorfismo**

Polimorfismo: Introdução

Definição

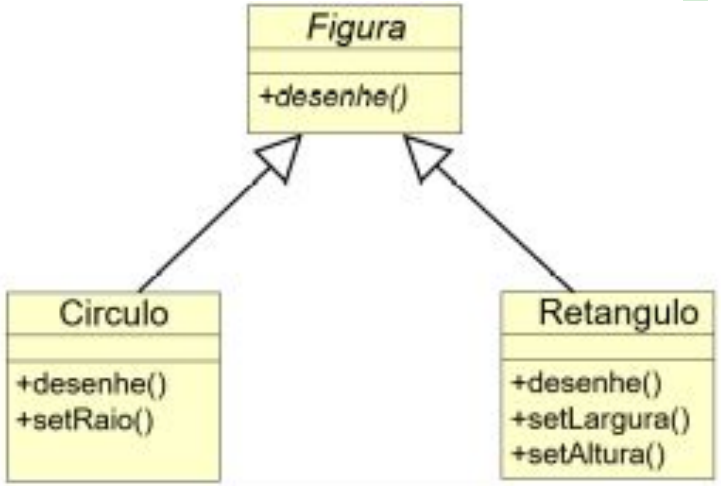
Polimorfismo = múltiplas formas

Objetos e operações podem assumir múltiplas formas

Polimorfismo: Introdução

Exemplo

```
figuras = []

retangulo = Retangulo(1, 2)

circulo = Circulo(2)

figuras.append(retangulo)
figuras.append(circulo)

for figura in figuras:
    print(figura.desenhe())
```



Polimorfismo: garantido pela herança de Figura

Polimorfismo: Introdução

Exemplo

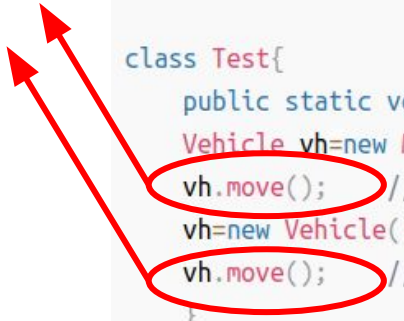
```
class Vehicle{
    public void move(){
    System.out.println("Vehicles can move!!");
    }
}

class MotorBike extends Vehicle{
    public void move(){
    System.out.println("MotorBike can move and accelerate too!!");
    }
}

class Test{
    public static void main(String[] args){
    Vehicle vh=new MotorBike();
    vh.move();    // prints MotorBike can move and accelerate too!!
    vh=new Vehicle();
    vh.move();    // prints Vehicles can move!!
    }
}
```

Polimorfismo: Introdução

Exemplo

mesmo método, objeto de mesmo tipo, comportamentos diferentes

```
class Vehicle{
    public void move(){
    System.out.println("Vehicles can move!!");
    }
}

class MotorBike extends Vehicle{
    public void move(){
    System.out.println("MotorBike can move and accelerate too!!");
    }
}

class Test{
    public static void main(String[] args){
    Vehicle vh=new MotorBike();
    vh.move();      // prints MotorBike can move and accelerate too!!
    vh=new Vehicle();
    vh.move();      // prints Vehicles can move!!
    }
}
```



Polimorfismo: Introdução

Por quê? Como sempre:

flexibilidade, extensibilidade e

manutenibilidade, evitar replicação...

Polimorfismo: Introdução

Exemplo - sem polimorfismo

```python
import math
class Quadrado():
    def __init__(self, lado: float):
        self.__lado = lado

    def areaQuadrado(self):
        print(self.__lado * self.__lado)


class Retangulo():
    def __init__(self, base: float, altura: float):
        self.__base = base
        self.__altura = altura

    def areaRetangulo(self):
        print(self.__base * self.__altura)


class Circulo():
    def __init__(self, raio: float):
        self.__raio = raio

    def areaCirculo(self):
        print(math.pi * self.__raio**2)

lista=[Quadrado(2),Retangulo(2,4),Circulo(3)]
lista[0].areaQuadrado()
lista[1].areaRetangulo()
lista[2].areaCirculo()
```

Polimorfismo: Introdução

Exemplo - com polimorfismo

```python
class Quadrado(FormaGeometrica):
    def __init__(self, lado: float):
        super().__init__()
        self.__lado = lado

    def calcula_area(self):
        self.area = self.__lado * self.__lado


class Retangulo(FormaGeometrica):
    def __init__(self, base: float, altura: float):
        super().__init__()
        self.__base = base
        self.__altura = altura

    def calcula_area(self):
        self.area = self.__base * self.__altura


class Circulo(FormaGeometrica):
    def __init__(self, raio: float):
        super().__init__()
        self.__raio = raio

    def calcula_area(self):
        self.area = math.pi * self.__raio**2
```

Polimorfismo: Introdução

Exemplo - com polimorfismo

```python
class Quadrado(FormaGeometrica):
    def __init__(self, lado: float):
        super().__init__()
        self.__lado = lado

    def calcula_area(self):
        self.area = self.__lado * self.__lado


class Retangulo(FormaGeometrica):
    def __init__(self, base: float, altura: float):
        super().__init__()
        self.__base = base
        self.__altura = altura

    def calcula_area(self):
        self.area = self.__base * self.__altura


class Circulo(FormaGeometrica):
    def __init__(self, raio: float):
        super().__init__()
        self.__raio = raio
```

```python
figuras = [Quadrado(2), Retangulo(2,3), Circulo(2), Circulo(3)]

for fig in figuras:
    fig.calcula_area()
    print(fig.area)
```

Polimorfismo: Introdução

Exemplo - com polimorfismo

Duck Typing

```python
class Quadrado(FormaGeometrica):
    def __init__(self, lado: float):
        super().__init__()
        self.__lado = lado

    def calcula_area(self):
        self.area = self.__lado * self.__lado


class Retangulo(FormaGeometrica):
    def __init__(self, base: float, altura: float):
        super().__init__()
        self.__base = base
        self.__altura = altura

    def calcula_area(self):
        self.area = self.__base * self.__altura


class Circulo(FormaGeometrica):
    def __init__(self, raio: float):
        super().__init__()
        self.__raio = raio
```

```python
figuras = [Quadrado(2), Retangulo(2,3), Circulo(2), Circulo(3)]

for fig in figuras:
    fig.calcula_area()
    print(fig.area)
```

Polimorfismo: Introdução

Exemplo - com polimorfismo

```python
class Quadrado(FormaGeometrica):
    def __init__(self, lado: float):
        super().__init__()
        self.__lado = lado

    def calcula_area(self):
        self.area = self.__lado * self.__lado


class Retangulo(FormaGeometrica):
    def __init__(self, base: float, altura: float):
        super().__init__()
        self.__base = base
        self.__altura = altura

    def calcula_area(self):
        self.area = self.__base * self.__altura


class Circulo(FormaGeometrica):
    def __init__(self, raio: float):
        super().__init__()
        self.__raio = raio
```

```python
figuras = [Quadrado(2), Retangulo(2,3), Circulo(2), Circulo(3)]

for fig in figuras:
    if isinstance(fig,FormaGeometrica):
        fig.calcula_area()
        print(fig.area)
```

Polimorfismo: Introdução

Exemplo - com polimorfismo

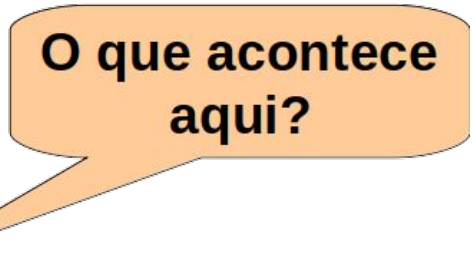
```
class Quadrado(FormaGeometrica):
    def __init__(self, lado: float):
        super().__init__()
        self.__lado = lado

    def calcula_area(self):
        self.area = self.__lado * self.__lado


class Retangulo(FormaGeometrica):
    def __init__(self, base: float, altura: float):
        super().__init__()
        self.__base = base
        self.__altura = altura

    def calcula_area(self):
        self.area = self.__base * self.__altura


class Circulo(FormaGeometrica):
    def __init__(self, raio: float):
        super().__init__()
        self.__raio = raio

    def calcula_area(self):
```

```
figuras = [Pessoa("João"),Quadrado(2), Retangulo(2,3), Circulo(2)]

for fig in figuras:
    if isinstance(fig,FormaGeometrica):
        fig.calcula_area()
        print(fig.area)
```

Polimorfismo: Introdução

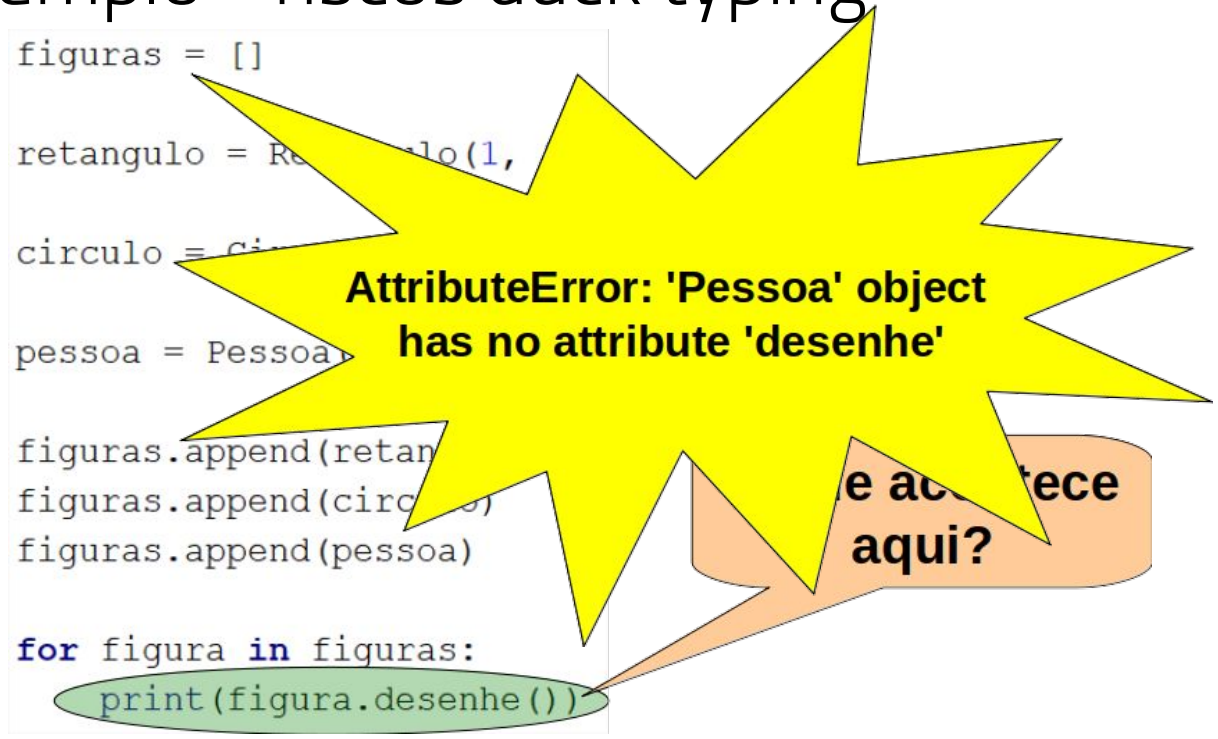
Exemplo - riscos duck typing

```
figuras = []

retangulo = Retangulo(1, 2)

circulo = Circulo(2)

pessoa = Pessoa("Jean")

figuras.append(retangulo)
figuras.append(circulo)
figuras.append(pessoa)

for figura in figuras:
    print(figura.desenhe())
```



Polimorfismo: Introdução

Exemplo - riscos duck typing



Polimorfismo: Introdução

Exemplo - riscos duck typing

```
figuras = []

retangulo = Retangulo(1, 2)

circulo = Circulo(2)

pessoa = Pessoa("Jean")

figuras.append(retangulo)
figuras.append(circulo)
figuras.append(pessoa)

for figura in figuras:
    print(figura.desenhe())
```

Não herda de Figura e não implementa "desenhe()"

Polimorfismo: Introdução

Definição - POO

Princípio pelo qual, objetos de duas ou mais classes derivadas de uma mesma superclasse podem invocar **operações que têm a mesma assinatura mas comportamentos distintos**, especializados para cada classe derivada, usando para tanto uma referência a um objeto do tipo da superclasse

Polimorfismo: Introdução

Definição - POO

"Polimorfismo é uma condição que existe quando a **tipagem dinâmica** e a **herança** interagem. Um único nome/identificador pode representar múltiplas classes relacionadas a uma mesma superclasse." [Booch, 2007]

Polimorfismo: Agenda

1. Introdução
2. **Amarração estática e dinâmica**
3. Exemplos
4. Herança Múltipla
5. Resumo

Polimorfismo - Amarração (binding)

Tipagem: forte x fraca

→ consistência de tipos. Ex:

```
"10"+5

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-7-11d4ded9e50f> in <module>
----> 1 "10"+5

TypeError: must be str, not int
```

Tipagem: estática x dinâmica

→ momento no qual tipos são amarrados a variáveis

Estática → compilação → early binding

Dinâmica → execução → late binding

Polimorfismo - Amarração (binding)

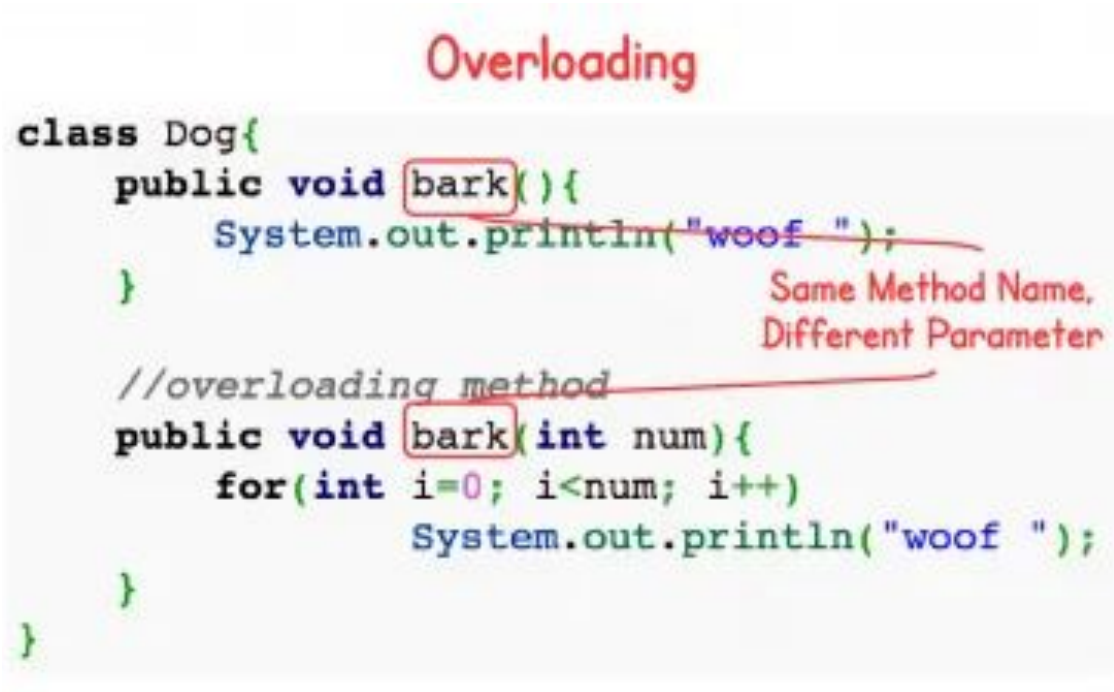
Tipagem: estática x dinâmica

Sobrecarga→ Estática → compilação → early binding

Sobreposição → Dinâmica → execução → late binding

Polimorfismo – Amarração (binding)

Sobrecarga - tempo de compilação – early binding



Polimorfismo – Amarração

Sobreposição

Tempo de execução

late binding

```python
class Veiculo:
    # ...

    def lavar(self):
        total = self.dono.dinheiro
        if total >= 30:
            self.dono.dinheiro = total - 30
            self.__sujo = False

    def frear(self):
        pass

    # ...


class Carro(Veiculo):
    # ...

    def frear(self):
        self.velocidade -= 10
        if self.velocidade < 0:
            self.velocidade = 0


class Bicicleta(Veiculo):
    # ...

    def lavar(self):
        total = self.dono.dinheiro
        if total >= 10:
            self.dono.dinheiro = total - 10
            self.sujo = False

    def frear(self):
        self.velocidade -= 1
        if self.velocidade < 0:
            self.velocidade = 0
```

Polimorfismo – Amarração (binding)

Sobreposição
Tempo de execução
late binding

```
class Vehicle{
    public void move(){
    System.out.println("Vehicles can move!!");
    }
}

class MotorBike extends Vehicle{
    public void move(){
    System.out.println("MotorBike can move and accelerate too!!");
    }
}

class Test{
    public static void main(String[] args){
    Vehicle vh=new MotorBike();
    vh.move();    // prints MotorBike can move and accelerate too!!
    vh=new Vehicle();
    vh.move();    // prints Vehicles can move!!
    }
}
```

Polimorfismo – Amarração (binding)

Amarração Tardia

- Permite com que as aplicações invoquem métodos especializados a partir de uma classe base comum

- Maior flexibilidade para a escrita de código reutilizável

- Desvantagem: menor eficiência computacional em C++, há cerca de 15% de custo extra na chamada do método

Polimorfismo – Amarração (binding)

Amarração Tardia

```
class Vehicle{
    public void move(){
    System.out.println("Vehicles can move!!");
    }
}

class MotorBike extends Vehicle{
    public void move(){
    System.out.println("MotorBike can move and accelerate too!!");
    }
}

class Test{
    public static void main(String[] args){
    Vehicle vh=new MotorBike();
    vh.move();    // prints MotorBike can move and accelerate too!!
    vh=new Vehicle();
    vh.move();    // prints Vehicles can move!!
    }
}
```

late binding

Polimorfismo – Amarração (binding)

Amarração estática (early binding)

- Em C++ os métodos são estaticamente amarrados por default (independente do tipo do objeto)

```
class Pessoa {
  public:
    void mostrar() { println ("pessoa"); }
}
class Aluno: public Pessoa {
   public:
    void mostrar() { println ("Aluno"); }
}
  :              :
Pessoa *p = new Aluno;
p->mostrar(); // chama o "mostrar" de Pessoa
```

Polimorfismo - Amarração (binding)

Algumas linguagens deixam o programador decidir:

- Em C++ amarração estática por padrão, mudando isso através da palavra-chave **virtual**

```
class Pessoa {
 public:
    void ler() { }                  // estática
    virtual void imprimir() { } // tardia
}
```

Polimorfismo - Amarração (binding)

Algumas linguagens deixam o programador decidir:

- Em Java amarração tardia por padrão, mudando isso através da palavra-chave **final**

```
3  public class Base {
4
5      final void foo(int i) {}
6
7      //overloading is allowed
8      void foo(int i, String s){}
9  }
10
11 class Child extends Base{
12
13     @Override
14     void foo(int i, String s) {}
15
16     @Override
17     void foo(int i) {}
18 }
```

Cannot override the final method from Base

1 quick fix available:

Remove 'final' modifier of 'Base.foo'(..)

Polimorfismo: Agenda

1. Introdução
2. Amarração estática e dinâmica
3. **Exemplos**
4. Herança Múltipla
5. Resumo

Polimorfismo – exemplos

- herança e polimorfismo

- Maior controle para evitar erros - tipos inconsistentes

- Alguns erros podem ser difíceis de achar em função do *duck typing*

```python
#superclasse
class Humano:
    def quem_sou(self):
        print("Eu sou um humano")

#subclasse
class Aluno(Humano):
    def quem_sou(self):
        print("Eu sou um aluno")


#subclasse
class Professor(Humano):
    def quem_sou(self):
        print("Eu sou um professor")

lista = [Humano(),Aluno(),Professor(),4]
for obj in lista:
    if isinstance(obj,Humano):
        obj.quem_sou()
    else:
        print("Objeto não compatível!")
```

```
Eu sou um humano
Eu sou um aluno
Eu sou um professor
Objeto não compatível!
```

Polimorfismo – exemplos

- herança e polimorfismo

```python
from abc import ABC, abstractmethod
import math

class FormaGeometrica(ABC):
    def __init__(self):
        self.__area = 0

    @property
    def area(self):
        return self.__area

    @area.setter
    def area(self, area: float):
        self.__area = area

    @abstractmethod
    def calcula_area(self):
        pass
```

```python
class Quadrado(FormaGeometrica):
    def __init__(self, lado: float):
        super().__init__()
        self.__lado = lado

    def calcula_area(self):
        self.area = self.__lado * self.__lado


class Retangulo(FormaGeometrica):
    def __init__(self, base: float, altura: float):
        super().__init__()
        self.__base = base
        self.__altura = altura

    def calcula_area(self):
        self.area = self.__base * self.__altura


class Circulo(FormaGeometrica):
    def __init__(self, raio: float):
        super().__init__()
        self.__raio = raio

    def calcula_area(self):
        self.area = math.pi * self.__raio**2
```

```python
figuras = [Quadrado(2), Retangulo(2,3), Circulo(2),

for fig in figuras:
    if isinstance(fig,FormaGeometrica):
        fig.calcula_area()
        print(fig.area)
```

Polimorfismo - exemplos

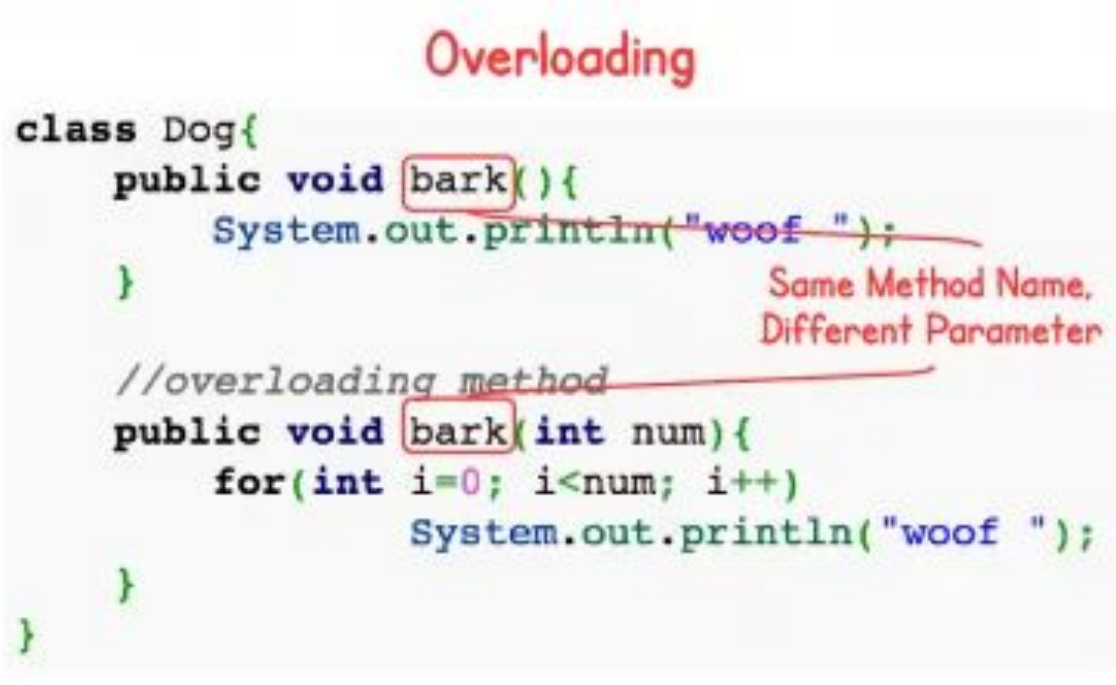
- herança e polimorfismo

- Em um determinado nível
  será importante apenas saber
  que é uma Ferramenta
  → ex: inventário do jogador
  → ex: ferramenta ativa
  → cada especialização é responsável pelos
  seus comportamentos

```
from abc import ABC, abstractmethod
class Ferramenta(ABC):
    @abstactmethod
    def __init__(self):
        pass

    @abstactmethod
    def descrever(self):
        pass

    @abstactmethod
    def usar(self):
        pass

    @abstactmethod
    def descartar(self):
        pass

class Martelo(Ferramenta):
        ...

class Pa(Ferramenta):
        ...

class Machado(Ferramenta):
```

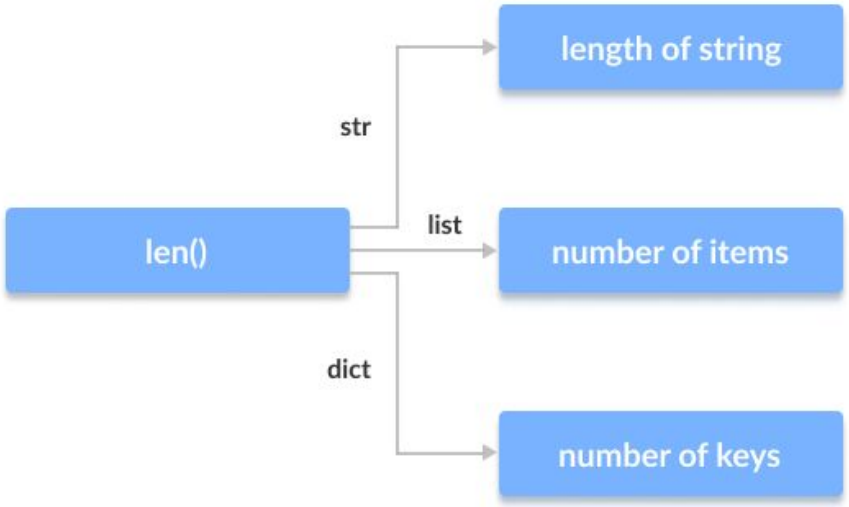
Polimorfismo – exemplos

- sentido amplo
  - **sobrecarga**
  - funções python
  - operadores python
  - duck typing



Polimorfismo - exemplos

- sentido amplo
  - sobrecarga
  - **funções python**
  - operadores python
  - duck typing



exemplo: https://www.programiz.com/python-programming/polymorphism

Polimorfismo - exemplos

- sentido amplo
  - sobrecarga
  - funções python
  - **operadores python**
  - duck typing

```
num1 = 1
num2 = 2
print(num1+num2)
```

```
str1 = "Python"
str2 = "Programming"
print(str1+" "+str2)
```


exemplo: https://www.programiz.com/python-programming/polymorphism

Polimorfismo - exemplos

- sentido amplo
  - sobrecarga
  - funções python
  - operadores python
  - **duck typing**

```python
class Cat:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def info(self):
        print(f"I am a cat. My name is {self.name}. I am {self.age} years old.")

    def make_sound(self):
        print("Meow")


class Dog:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def info(self):
        print(f"I am a dog. My name is {self.name}. I am {self.age} years old.")

    def make_sound(self):
        print("Bark")
```


exemplo: https://www.programiz.com/python-programming/polymorphism

Polimorfismo – exemplos

- sentido amplo
  - sobrecarga
  - funções python
  - operadores python
  - **duck typing**

```
cat1 = Cat("Kitty", 2.5)
dog1 = Dog("Fluffy", 4)

for animal in (cat1, dog1):
    animal.make_sound()
    animal.info()
    animal.make_sound()
```

Output

```
Meow
I am a cat. My name is Kitty. I am 2.5 years old.
Meow
Bark
I am a dog. My name is Fluffy. I am 4 years old.
Bark
```

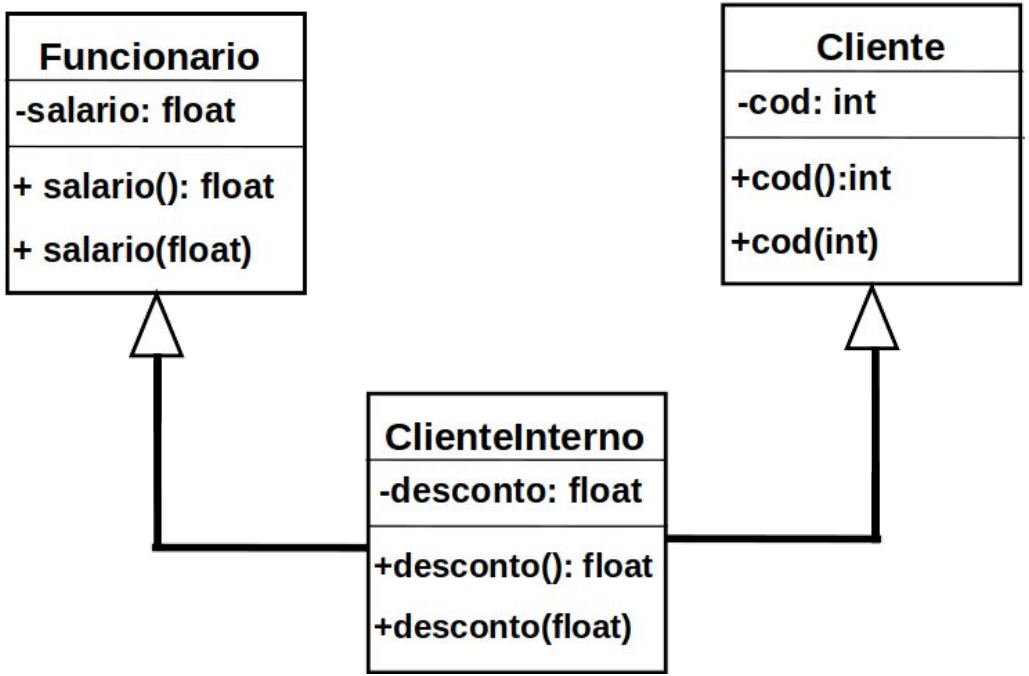

exemplo: https://www.programiz.com/python-programming/polymorphism

Polimorfismo: Agenda

1. Introdução
2. Amarração estática e dinâmica
3. Exemplos
4. **Herança Múltipla**
5. Resumo

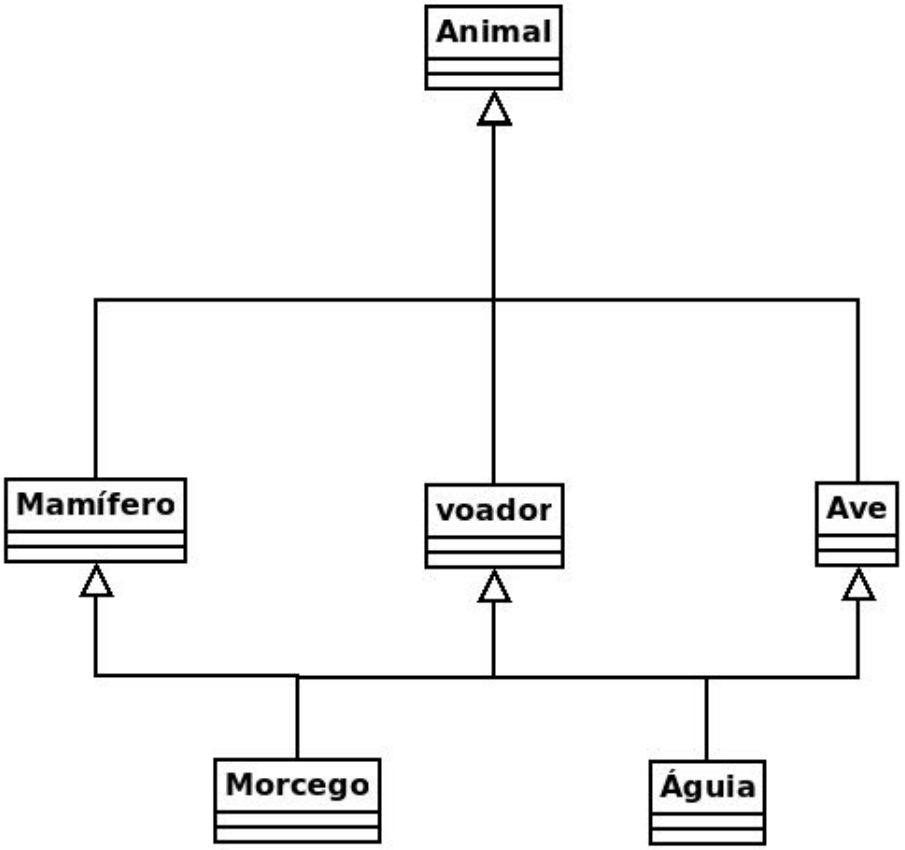
Herança Múltipla

- Classe que herda mais de uma superclasse



Herança Múltipla – exemplos

- Animais

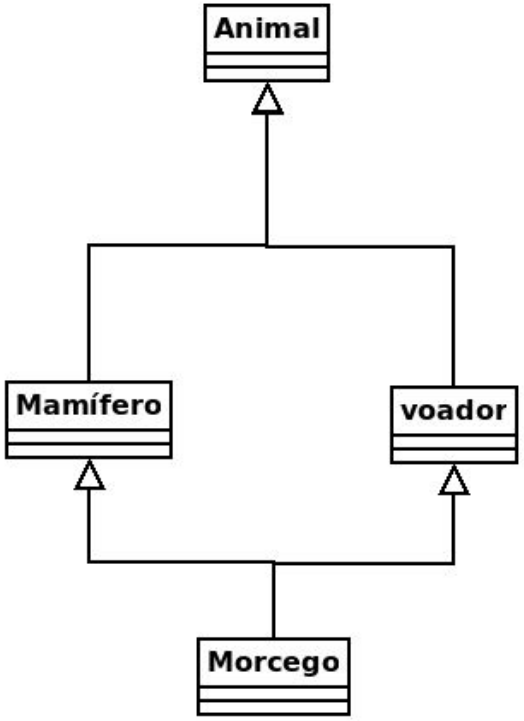


Herança Múltipla – exemplos

- Animal
  - Morcego = mamífero voador

Questão:

- Se um método da superclasse Animal é redefinido nas subclasses (mamífero e voador), qual método o objeto Morcego irá usar?

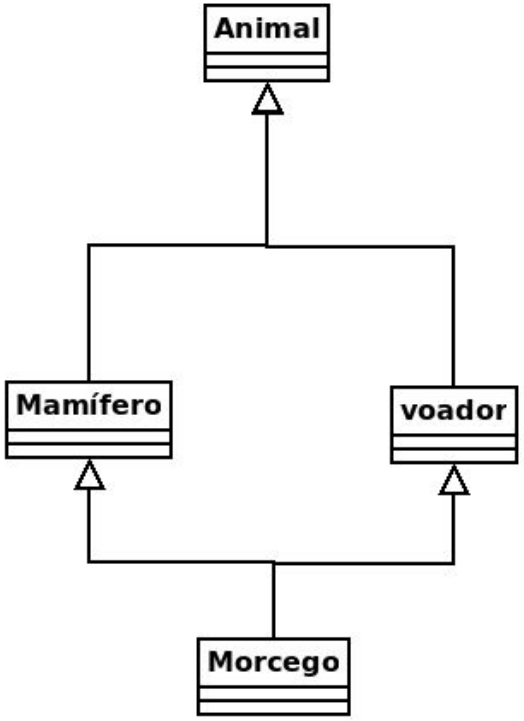


Herança Múltipla – diamond problem

- Animal
  - Morcego = mamífero voador

Questão:

- Se um método da superclasse Animal é redefinido nas subclasses (mamífero e voador), qual método o objeto Morcego irá usar?

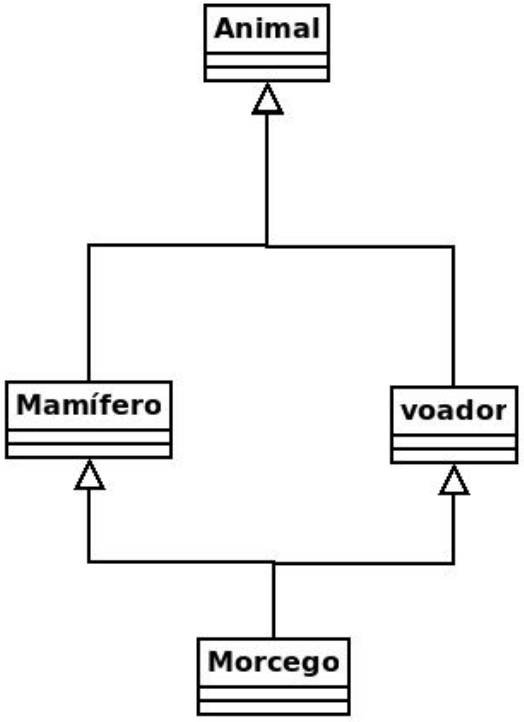


Problema do diamante (diamond problem)

Herança Múltipla – diamond problem

Opções para resolver

1. Solução Python: definir na ordem da herança múltipla (Python) - (ver jupyter notebook)
2. Solução c++: programador define
   a. amarração dinâmica (virtual)
3. Proíbe-se o uso - ambíguo (Java interfaces)



Herança Múltipla – diamond problem

Solução Java - Interfaces

• Interfaces = classes completamente abstratas
– Nenhuma implementação
– Como classes abstratas: não são instanciadas
– Podem conter constantes

Agrupam objetos que oferecem os mesmos tipos de serviços (tipos de operações). Ex:
– Interface da entidade Animal: come, anda, reproduz
– Todo e qualquer animal deve implementar estas funcionalidades, mas cada um a seu modo

Polimorfismo: Agenda

1. Introdução
2. Amarração estática e dinâmica
3. Exemplos
4. Herança Múltipla
5. **Resumo**

Polimorfismo: Resumo

- Um dos pilares da orientação a objetos
- Definição de múltiplas formas para objetos e operações - flexibilidade e extensibilidade
- Amarração estática e dinâmica (early x late)
- Herança múltipla - problema do diamante
  - Diferentes soluções

Referências

DATHAN, B.; RAMNATH, S. **Object-Oriented Analysis, Design and Implementation**. Cham: Springer, 2015

BOOCH G.; JACOBSON, I.; RUMBAUGH, J. **Object-Oriented Analysis and Design with Applications.** Third Edition. Addison-Wesley Professional, 2007