INSTITUTO FEDERAL
Paraná
Campus
Foz do Iguaçu

Orientação a Objetos

Prof. Daniel Di Domenico

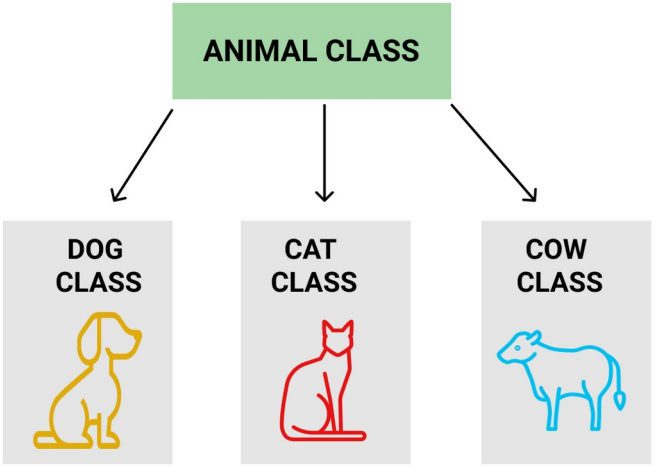
Polimorfismo

O que já sabemos?

INSTITUTO FEDERAL
Paraná
Campus
Foz do Iguaçu

- Linguagem PHP

- Orientação a objetos
  - Abstração
  - **Classes e Objetos**
    - Atributos e Métodos
    - Listas
  - **Encapsulamento**
  - **Associação**
  - **Interfaces**
  - **Herança**



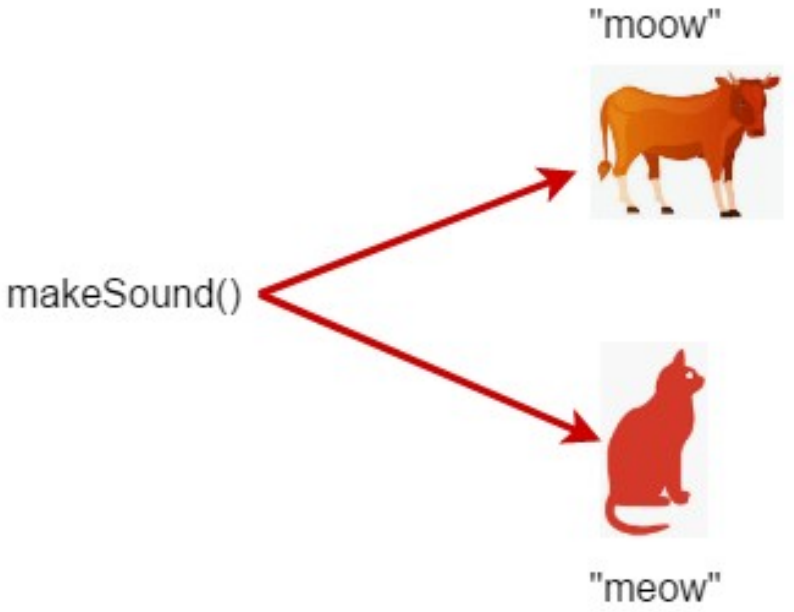
Objetivos da aula

INSTITUTO FEDERAL
Paraná
Campus
Foz do Iguaçu

- Conhecer o conceito de **polimorfismo**

- Conhecer os **tipos de polimorfismo**
  - Sobrecarga
  - Sobrescrita (com herança)

- **Polimorfismo: quarto** pilar da OO
  - 1º: Abstração
  - 2º: Encapsulamento
  - 3º: Herança



Polimorfismo

INSTITUTO FEDERAL
Paraná
Campus
Foz do Iguaçu

- **Polimorfismo** é um recurso da Orientação a Objetos que permite utilizar o **mesmo nome para mais de um método**
  - Termo grego: *Poli* = Muitas, *Morphos* = Formas
  - Características:
    - Com o polimorfismo, **uma mesma tarefa poderá ser implementada de formas diferentes**
      - Isso produzirá resultados diferentes
    - O programador precisa conhecer as regras do polimorfismo a fim de efetuar as chamadas corretas durante a implementação de uma aplicação

Tipos de Polimorfismo

INSTITUTO FEDERAL
Paraná
Campus
Foz do Iguaçu

- Existem diversos tipos de polimorfismo
  - Focaremos em dois deles

- **Sobrecarga** (*overloading*):
  - Também conhecido como polimorfismo horizontal
  - **Não requer herança**
    - Métodos criados com o mesmo nome, porém com parâmetros diferentes
      - Deve-se diferenciar a quantidade ~~ou o tipo dos parâmetros~~

Não suportado pelo PHP

- **Sobrescrita** (*overriding*):
  - Também conhecido como polimorfismo vertical ou sobreposição
  - **Requer herança**
    - Método da superclasse (pai) é sobrescrito na subclasse (filha)
      - O nome do método e quantidade/tipo de parâmetros nas classes pai/filha devem ser iguais

Tipos de Polimorfismo

INSTITUTO FEDERAL
Paraná
Campus
Foz do Iguaçu

- Existem diversos tipos de polimorfismo
  - Focaremos em dois deles

- **Sobrecarga** (*overloading*):
  - Também conhecido como polimorfismo horizontal
  - **Não requer herança**
    - Métodos criados com o mesmo nome, porém com parâ
      - Deve-se diferenciar a quantidade ou o tipo dos par
- **Sobrescrita** (*overriding*):
  - Também conhecido como polimorfismo vertical ou sobreposição
  - **Requer herança**
    - Método da superclasse (pai) é sobrescrito na subclasse (filha)
      - O nome do método e quantidade/tipo de parâmetros nas classes pai/filha devem ser iguais

Sobrescrita faz com que os objetos **comportem-se de formas diferentes**

Polimorfismo

INSTITUTO FEDERAL
Paraná
Campus
Foz do Iguaçu

- **Vantagens de utilizar Polimorfismo:**
  - Permite especializar os métodos de acordo com a necessidade das classes filhas
  - Possibilita tratar os objetos instanciados a partir da mesma classe base (pai) de maneira igual
  - Organização e aproveitamento de código
    - **Caso o método herdado da classe pai não atenda a necessidade da classe filha, pode-se sobrescrevê-lo**
      - Assim, não há necessidade de definir novas classes para tratar casos específico

Polimorfismo - Implementação

INSTITUTO FEDERAL
Paraná
Campus
Foz do Iguaçu

- Implementação de **sobrecarga (classe/modelo)**

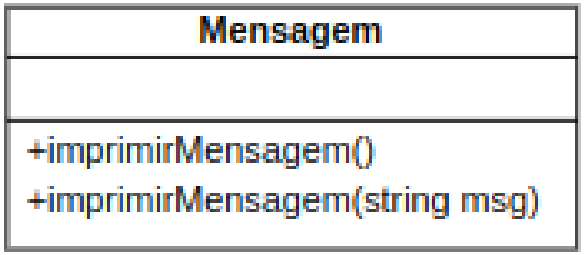
```php
class Mensagem {

    //Código da classe
    public function imprimirMensagem(
            string $msg="Mensagem padrão!") {

        echo $msg . "\n";
    }

}
```

Para sobrecarga no PHP, utilizam-se valores padrão



Polimorfismo - Implementação

INSTITUTO FEDERAL
Paraná
Campus
Foz do Iguaçu

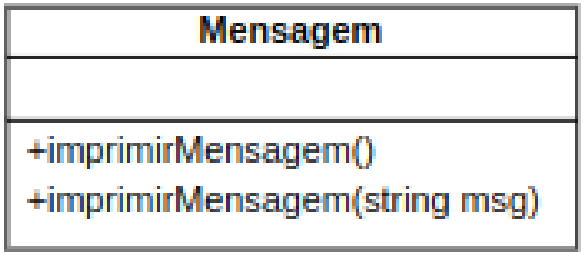
- Implementação de **sobrecarga (execução)**

```php
include_once("modelo/Mensagem.php");

$mensagem = new Mensagem();

//Executa o método sem parâmetros
$mensagem->imprimirMensagem();

//Executa o método que recebe uma String
$texto = "Olá classe!";
$mensagem->imprimirMensagem($texto);
```



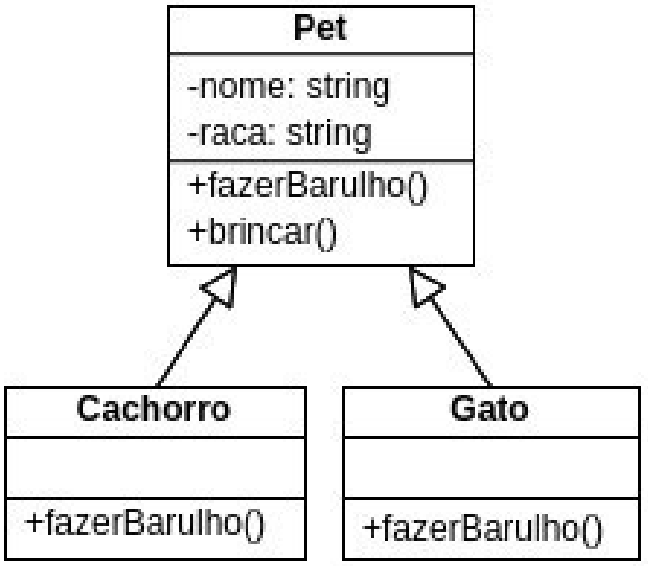
Polimorfismo - Implementação

INSTITUTO FEDERAL
Paraná
Campus
Foz do Iguaçu

- Implementação **sobrescrita (classes/modelo)**

```php
class Pet {

    private string $nome;
    private string $raca;
    public function fazerBarulho() {
        echo "Pet fazendo barulho!\n";
    }
    public function brincar() {
        echo "Pet brincando!\n";
    }
}
```




```php
class Cachorro extends Pet {
    public function fazerBarulho() {
        echo "auauauau\n";
    }
}
```

```php
class Gato extends Pet {
    public function fazerBarulho() {
        echo "miau\n";
    }
}
```

Polimorfismo - Implementação

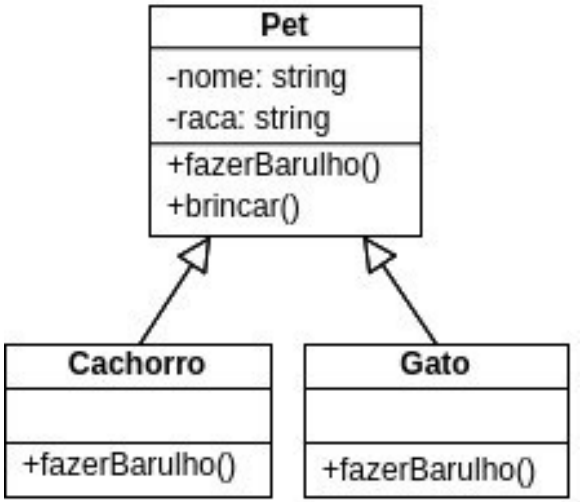
INSTITUTO FEDERAL
Paraná
Campus
Foz do Iguaçu

- Implementação de **sobrescrita (objetos)**

```
$pet = new Pet();
$pet->setNome("Jujuba");
$pet->setRaca("Inglês");
$pet->fazerBarulho();
$pet->brincar();

$cao = new Cachorro();
$cao->setNome("Scoob");
$cao->setRaca("Pinscher");
$cao->fazerBarulho();
$cao->brincar();

$gato = new Gato();
$gato->setNome("Felix");
$gato->setRaca("Persa");
$gato->fazerBarulho();
$gato->brincar();
```



ATENÇÃO:
sempre será chamado
o método da classe
mais específica

Polimorfismo - Implementação

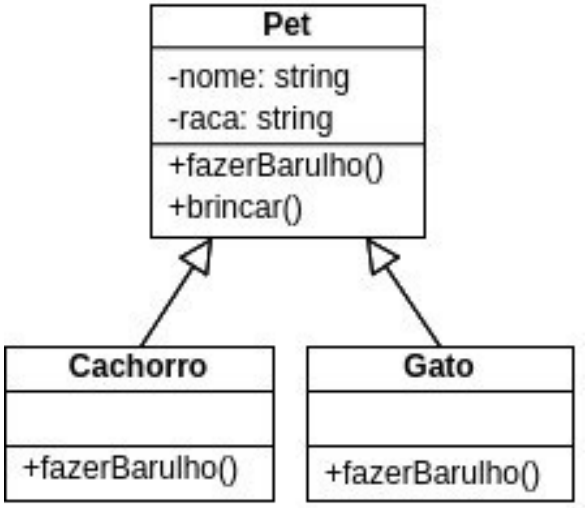
INSTITUTO FEDERAL
Paraná
Campus
Foz do Iguaçu

- Implementação de **sobrescrita (objetos/execução)**

```
$pet = new Pet();
$pet->setNome("Jujuba");
$pet->setRaca("Inglês");
$pet->fazerBarulho();
$pet->brincar();

$cao = new Cachorro();
$cao->setNome("Scoob");
$cao->setRaca("Pinscher");
$cao->fazerBarulho();
$cao->brincar();

$gato = new Gato();
$gato->setNome("Felix");
$gato->setRaca("Persa");
$gato->fazerBarulho();
$gato->brincar();
```



Imprime: Pet fazendo barulho!

Imprime: auauauau

Imprime: miau

Exercícios

INSTITUTO FEDERAL
Paraná
Campus
Foz do Iguaçu

- **1-** Uma loja de departamentos vende diversos tipo de produtos. Considerando isso, crie uma classe **Produto** com os atributos *descricao* e *unidadeMedida* e o método *getDados()* (retorna todos os dados do produto concatenados em uma String). Após, crie as subclasses de **Produto**, sendo:
  - Livro: atributo *autor* e sobrescrita do método *getDados*()
  - Computador: atributos *processador* e *memória*, além da sobrescrita do método *getDados*()
  - Balde: atributo *capacidade* e sobrescrita do método *getDados*()

  Por fim, no arquivo de execução, instancie um objeto para cada uma das 4 classes, chamando o método *getDados()* de todos eles.

- **2-** Uma pessoa que coleciona mídias quer catalogar os seus CDs e DVDs. Diante disso, pensou-se na seguinte estrutura de classes:
  - Classe **Midia**, com os atributos *descricao*, *precoPago* e métodos *getDados()* (retorna os dados da mídia em uma String) e *getTipo()* (retorna uma String com o tipo da mídia)
  - Classes **CD** e **DVD**, que estendem a classe **Mídia** e sobrescrevem o método *getTipo()*

  Por fim, crie uma classe de execução para ler os dados de 5 mídias (tipo, descricao e precoPago), adicionando cada objeto criado em uma lista (vetor) de mídias. Após a leitura, percorra a lista e imprima os dados e o tipo de todas as 5 mídias previamente lidas e armazenadas na lista.