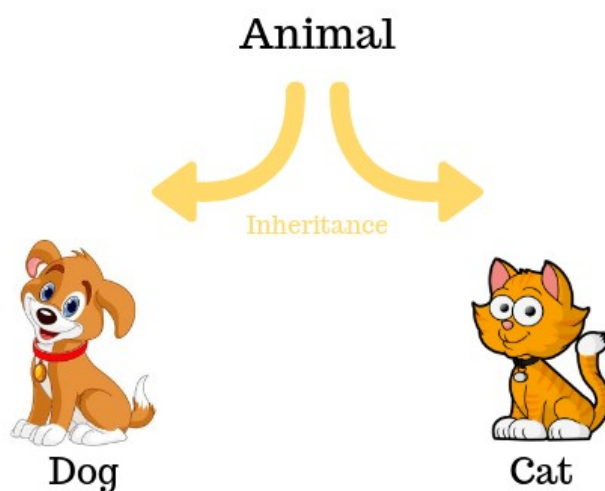


Orientação a Objetos

Prof. Daniel Di Domenico

Herança



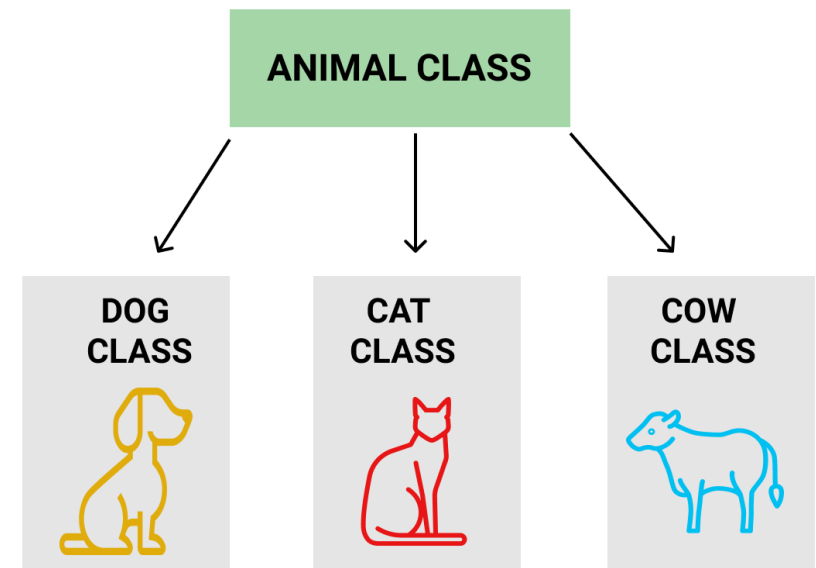
O que já sabemos?

- Linguagem PHP
- Orientação a objetos
 - Abstração
 - **Classes e Objetos**
 - Atributos e Métodos
 - **Encapsulamento**
 - **Arrays de objetos**
 - **Associação**



Objetivos da aula

- Conhecer o conceito de **herança**
- Conhecer o modificador de acesso **protegido**
 - Símbolo: #
- **Herança**: terceiro pilar da OO
 - 1º: Abstração
 - 2º: Encapsulamento

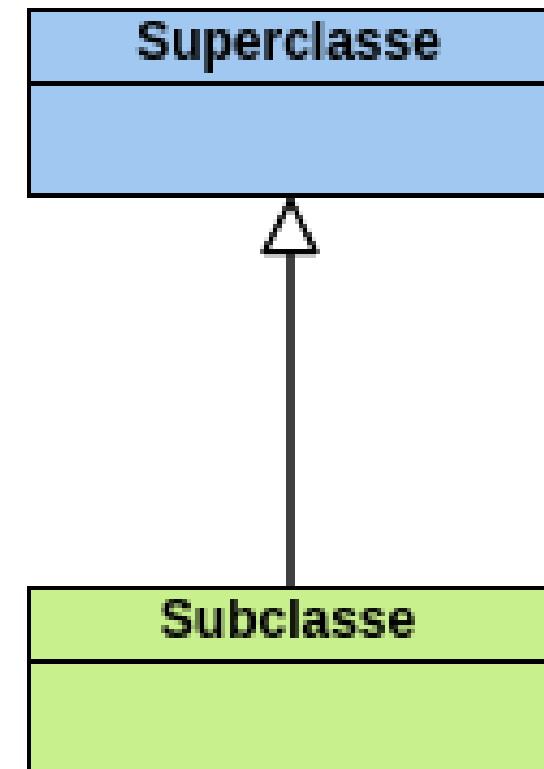


Herança

- **Herança** é um mecanismo da Orientação a Objetos que possibilita a uma classe **usar os atributos e métodos definidos em outra classe**
 - Características:
 - Compartilhamento de atributos/métodos
 - Relação hierárquica
 - Uma classe **pai/mãe** empresta suas definições para as classes **filhas**

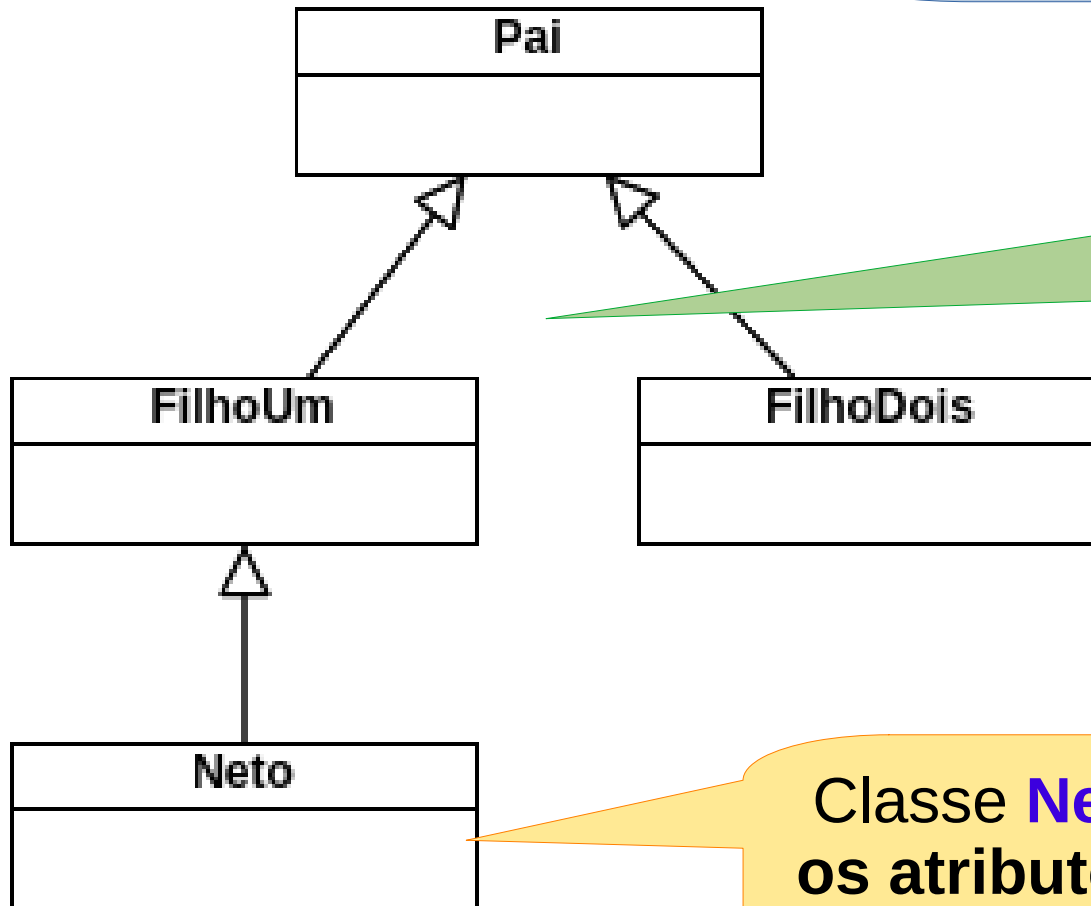
Herança

- Conceitos:
 - **Superclasse:** classe pai/base
 - **Compartilha** seus atributos/métodos com a(s) classe(s) filha(s)
 - **Subclasse:** classe filha
 - **Herda** os atributos de uma classe pai



Representação de herança

Representação da relação de herança em UML



Classes **FilhoUm** e **FilhosDois** herdam todos os atributos/métodos da classe **Pai**

Classe **Neto** herda todos os atributos/métodos das classes **FilhoUm** e **Pai**

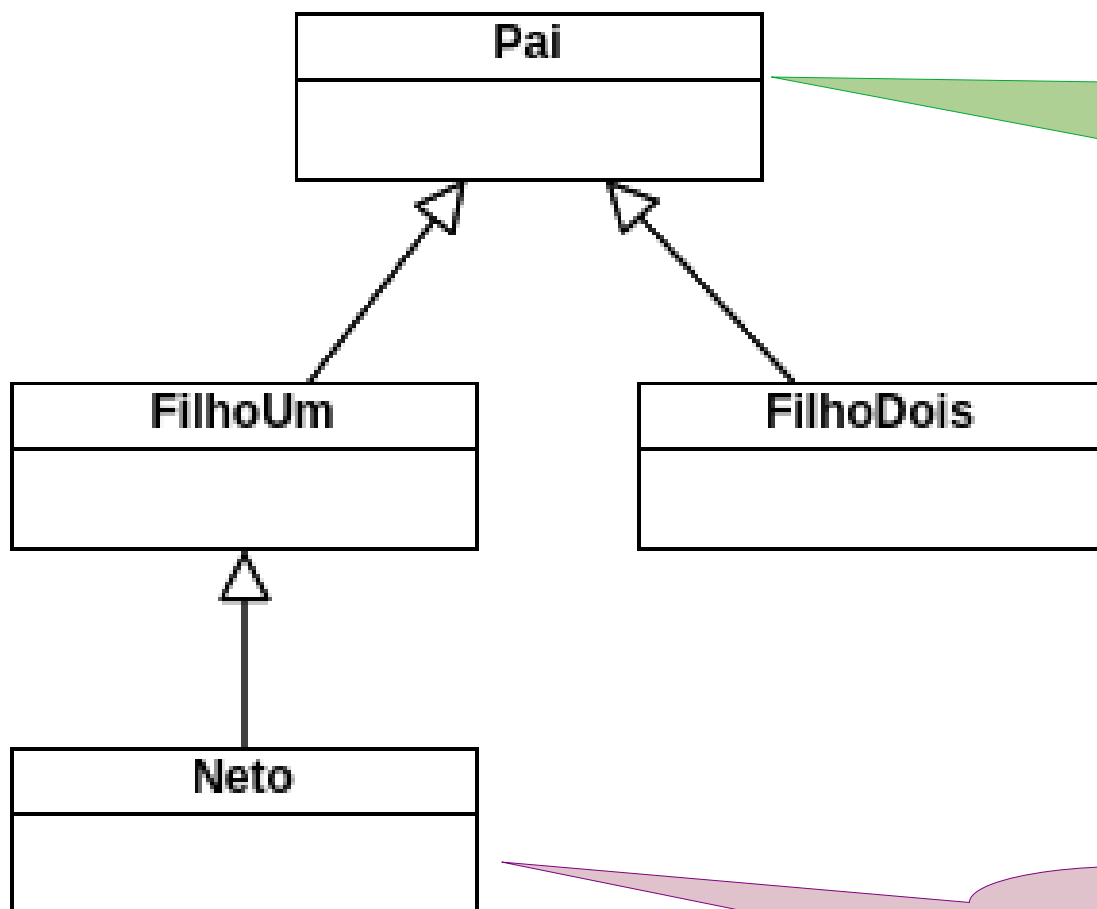
Representação de herança



INSTITUTO
FEDERAL

Paraná

Campus
Foz do Iguaçu

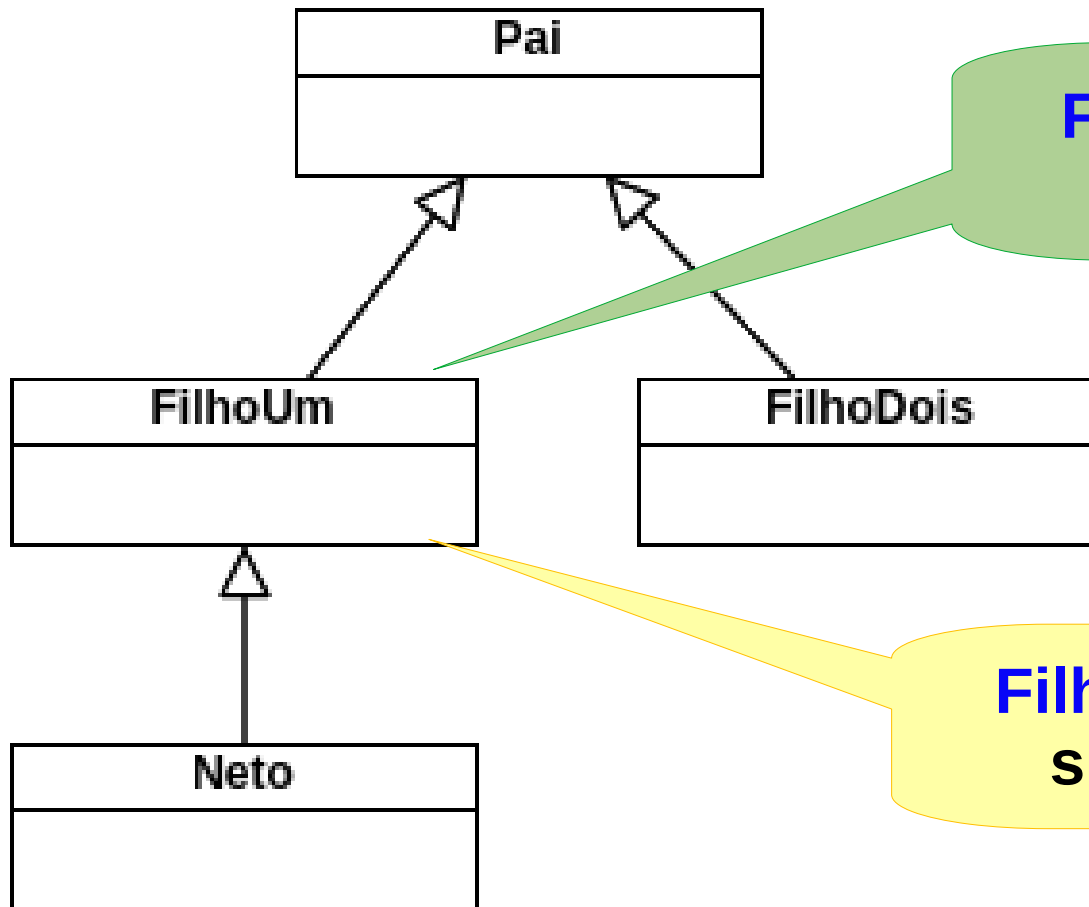


Pai: classe **base** ou **superclasse** de **FilhoUm**, **FilhoDois** e **Neto**

FilhoDois: classe **filha** ou **subclasse** de **Pai**

Neto: classe **filha** ou **subclasse** de **FilhoUm** e **Pai**

Representação de herança



FilhoUm: classe **filha** ou subclasse de **Pai**

FilhoUm: classe **base** ou superclasse de **Neto**

Herança

- Vantagens de utilizar Herança:
 - Reaproveitamento de código
 - Evitar a repetição de código
 - Atributos e métodos são compartilhados
 - Organização do código
 - **Compartilhar** os atributos/métodos que são **comuns**
 - **Isolar** os atributos/métodos que são **diferentes**



Herança - Implementação

- **Implementação em PHP**

- A herança é definida pelo termo **extends**:

```
class Pai {  
    //Código da classe  
}
```

```
class Filho extends Pai {  
    //Código da classe  
}
```

Herança - Implementação

- **Domínio de escola**
 - Aluno (nome, RG, idade, matricula)
 - Professor (nome, RG, idade, salario)
- Como implementar essas classes na aplicação?
 - **SEM HERANÇA:**
 - Implementar as 2 classes de forma separada
 - **COM HERANÇA:**
 - Agrupar os atributos comuns em uma terceira classe (Pessoa)
 - As classes Aluno e Professor devem herdar (estender) a classe Pessoa

Herança - Implementação

- **Domínio de escola – Aluno e Professor**
 - Implementação **SEM HERANÇA**

```
class Aluno {  
  
    private string $nome;  
    private string $rg;  
    private int $idade;  
    private int $matricula;  
  
}
```

```
class Professor {  
  
    private string $nome;  
    private string $rg;  
    private int $idade;  
    private float $salario;  
  
}
```

Herança - Implementação

- Domínio de escola – Aluno e Professor
 - Implementação **COM HERANÇA**

```
class Pessoa {  
  
    private string nome;  
    private string rg;  
    private int idade;  
  
}
```

Classe Pai

Classes Filhas

```
class Aluno extends Pessoa {  
    private int $matricula;  
  
}
```

```
class Professor extends Pessoa {  
    private float $salario;  
  
}
```

Herança - Implementação

- Domínio de escola – Aluno
 - Implementação **COM HERANÇA:** objetos

```
$aluno = new Aluno();  
  
$aluno->setNome("Hermioni Granger");  
$aluno->setIdade(18);  
$aluno->setRg("424.234.434");  
$aluno->setMatricula(234343233);
```

Atributos, GETs e SETs (nome, idade, rg) definidos na classe **Pessoa**

Herança - Implementação

- **Domínio de escola – Professor**
 - Implementação **COM HERANÇA:** objetos

```
$professor = new Professor();
```

```
$professor->setNome("Severus Snape");
```

```
$professor->setIdade(42);
```

```
$professor->setRg("654.323.677");
```

```
$professor->setSalario(2500.00);
```

Atributos, GETs e SETs (nome, idade, rg) definidos na classe **Pessoa**

Modificador - protegido

- Modificador de acesso protegido

- **protected:**

- símbolo: #
 - permite acesso a um atributo/método apenas dentro do escopo da classe **ou no escopo das classes filhas** (herança)



Modificador - protegido

- **Domínio de escola – Aluno**
 - Diferença **protected** e **private**

```
class Aluno extends Pessoa {  
  
    private int $matricula;  
  
    //Métodos  
    public function __toString() {  
        $dados = "Nome: " . $this->nome;  
        $dados .= " - RG: " . $this->getRg();  
        $dados .= " - Idade: " . $this->getIdade();  
        $dados .= " - Matríc: " . $this->matricula;  
        return $dados . "\n";  
    }  
}
```

Atributo **nome** é **protegido**: acesso direto na classe filha

```
class Pessoa {  
    protected string $nome;  
    private string $rg;  
    private int $idade;  
}
```

Exercícios

- **1-** Faça um programa que tenha uma classe pai denominada Animal com os atributos *nome* e *raca* e o método *getDados()* que retorna uma String com os valores setados para os seus atributos. Depois, crie duas classes filhas de Animal, sendo:

- Gato (com o método miar)
- Cachorro (com o método latir)

Por fim, no programa de execução, crie 4 objetos, sendo 2 do tipo Gato e 2 do tipo Cachorro, chamando, para cada um, seus métodos *getDados()* e *latir()*|*miar()*.

- **2-** Utilizando abstração e herança, crie as classes em um programa a partir da seguinte descrição do domínio:
 - No esporte Futebol 2.0, o jogo se desenvolve utilizando regras específicas para os jogadores. Tais jogadores são classificados por posições, sendo goleiro, defensor, meia e atacante. Todos os jogadores, independente da posição, possuem nome, idade e nacionalidade. Para os goleiros e defensores, é importante saber sua altura. Para os meias e atacantes, é importante saber sua velocidade. Além disso, todos os jogadores podem correr e chutar. No entanto, apenas o goleiro pode defender e apenas os defensores podem desarmar e cabecear. Já os meias e atacantes podem fazer gols, porém só os meias podem dar assistências.