PHP - Avançando com Orientação a Objetos: Herança, Polimorfismo e Interfaces

- Como não aprender orientação a objetos: Herança - Blog da Caelum: desenvolvimento, web, mobile, UX e ScrumBlog da Caelum: desenvolvimento, web, mobile, UX e Scrum
- protected: private para a classe atual e as filhas.

- Chamando o construtor da classe mãe dentro do construtor da classe atual para evitar repetição de código:

```
class Titular extends Pessoa
{
    public Endereco $endereco;
    public function __construct(string $cpf, string $nome, Endereco $endereco)
    {
        parent::__construct($cpf, $nome);
        $this->endereco = $endereco;
    }
}
```

- namespace: Definindo namespaces, podemos facilmente ter duas classes com o mesmo nome em namespaces diferentes. Uma classe chamada Conta, por exemplo, pode ter um significado diferente no módulo de login e no módulo de contas. Utilizando um namespace raiz para a nossa aplicação, evitamos ainda conflitos com pacotes externos que viermos a utilizar.
- \\ = \ , mas evita leituras erradas, se \ for separação de pasta, por exemplo, usar \\ garante o funcionamento correto.
- autoload: se tentamos utilizar uma classe que o PHP ainda não conhece, ele utilizará o autoload (remove necessidade de repetir use e require). Exemplo:

```
<?php

spl_autoload_register(function (string $nomeCompletoDaClasse){
    $caminhoArquivo = str_replace('Alura\\Banco', 'src', $nomeCompletoDaClasse);
    $caminhoArquivo = str_replace('\\', DIRECTORY_SEPARATOR, $caminhoArquivo);
    $caminhoArquivo .= '.php';
```

```
    if(file_exists($caminhoArquivo)) {
        require_once $caminhoArquivo;
    }
});
```

- abstract: podemos ter classes que ainda não estão prontas para serem instanciadas e precisam ser estendidas (classes abstratas);também os métodos abstratos, que são uma forma de "obrigar" que classes filhas implementem determinado método; apenas classes abstratas podem ter métodos abstratos;

- Simplificando importação de namespace:

```
use Alura\Banco\Modelo\Conta\Conta;
use Alura\Banco\Modelo\Conta\ContaPoupanca;
use Alura\Banco\Modelo\Conta\Titular;
use Alura\Banco\Modelo\CPF;
use Alura\Banco\Modelo\Endereco;
```

-------->

```
use Alura\Banco\Modelo\Conta\{Conta, ContaPoupanca, Titular};
use Alura\Banco\Modelo\{CPF, Endereco};
```

- Uma classe de serviço representa uma funcionalidade, enquanto uma classe de modelo representa algo real em nosso modelo domínio
- Não é possível instanciar uma classe abstrata.
- Uma classe pode ser abstrata mesmo sem ter métodos abstratos, isso só quer dizer que é uma classe não instanciável, como Pessoa, pois só existem Funcionarios ou Titulares, e ambos são Pessoa.
- Herança múltipla não é possível no PHP (nem em Java, é possível, por exemplo, em C++).
- Problema de herança múltipla é resolvido com interfaces:
    - Todas as classes que decidirem implementar uma interface precisam implementar todos os métodos nela definidos.
    - No fundo, interfaces são apenas classes abstratas que contém apenas métodos abstratos;

```
interface Autenticavel
{
    public function podeAutenticar(string $senha): bool;
}

class Gerente extends Funcionario implements Autenticavel
{
}
```

- Sempre que tentarmos representar um objeto como string (seja com echo, passando por parâmetro ou retornando) o método __toString é procurado, e se a classe o tiver, seu retorno é utilizado.
- __get( )
- final : não permite herança (tanto de classe quanto de método).
- Traits: uma forma do próprio PHP copiar código de algum local e injetá-lo na classe desejada:

```
trait AcessoPropriedades
{
    public function __get(string $nomeAtributo)
    {
        $metodo = 'recupera' . ucfirst($nomeAtributo);
        return $this->$metodo();
    }
}

final class Endereco
{
    use AcessoPropriedades;
    private $cidade;
    private $bairro;
    private $rua;
    private $numero;
//...
```

Dúvidas e assuntos a pesquisar

- PSR-4
- Meu post no fórum da Alura: Não está encontrando a classe na execução | Avançando com Orientação a Objetos com PHP: Herança, Polimorfismo e Interfaces | Alura - Cursos online de tecnologia
- Polimorfismo: Caelum Escola de Tecnologia Cursos Online
- Problema do diamante: Caelum Escola de Tecnologia Cursos Online ; Multiple inheritance - Wikipedia
- Traits: PHP: Traits - Manual