DIO
CEZAR
GO

PHP

parte 4

PHP OO

PHP ORIENTADO A OBJETOS

- temos métodos específicos para construtores e destrutores;
  - construtor → `__construct()`
  - destrutor → `__destruct()`

EXEMPLO DE CLASSE EM PHP

```
<?php
      class Person{
            public function __construct(){
                  // construtor;
            }
            public function __destruct(){
                  // destrutor
            }
      }
      $p = new Person();    // invoca o construtor
      unset($p);            // invoca o destrutor
?>
```

TIPOS DE ACESSO

- temos os seguintes modificadores de acesso para nossas variáveis ou métodos:
  - `public` → todos podem acessar, dentro ou fora da classe;
  - `private` → só é acessado dentro da classe;
  - `protected` → só é acessado dentro da classe ou por seus herdeiros;

O OPERADOR FINAL

- pode-se indicar que um método é "final".
  - com isso, não se permite sobrescrever esse método, em uma nova classe que o herde;
- se uma classe é "final" indica-se que esta classe não permite ser herdada por outra classe;

OPERADOR STATIC

- podemos fazer uso de atributos e métodos "`static`";
- permitem acesso a partir do nome da classe, sem necessidade da criação de um objeto;
- pode-se dizer que tais métodos ou atributos pertencem a classe e não a um de seus objetos;

O OPERADOR ABSTRACT

- as classes abstratas não são instanciadas, costumam ser utilizadas para herdá-las de outras classes que implementarão seu conteúdo;
- todos os métodos declarados como abstract, deverão ser implementados na classe filha;
- pode-se implementar uma função;
- pode-se declarar atributos;
- protected, será visível na classe herdeira;

EXEMPLO CONTA CORRENTE

CONTA-CORRENTE.PHP

```php
<?php
     class ContaCorrente{
          public $saldo;
          function __construct($valor){
               $this->saldo = $valor;
          }
          function saque($valor){
               if($this->saldo >= $valor){
                    $this->saldo -= $valor;
               }
          }
          function deposito($valor){
               $this->saldo += $valor;
          }
     }
?>
```

EXEMPLO CONTA CORRENTE

INDEX.PHP

```
<?php
     include("class/conta-corrente.php");
     $cc = new ContaCorrente(1000);
     $cc->saque(500);
     $cc->saque(500);
     $cc->saque(10);
     $cc->deposito(150);
     echo $cc->saldo; // imprime 150
?>
```

HERANÇA EM PHP

- uma classe pode estender (`extends`) outra classe qualquer;
- todos os atributos e métodos estão disponíveis imediatamente, pela variável `$this;`
- o construtor da superclasse deve ser chamado explicitamente pelo construtor da subclasse → `parent::__construct();`
- a definição da subclasse deve incluir a definição da superclasse → `Class1 extends Class2;`

EXEMPLO CONTA CORRENTE

CONTA-CORRENTE.PHP

```php
<?php
     class ContaCorrente{
          public $saldo;
          function __construct($valor){
               $this->saldo = $valor;
          }
          function saque($valor){
               if($this->saldo >= $valor){
                    $this->saldo -= $valor;
               }
          }
          function deposito($valor){
               $this->saldo += $valor;
          }
     }
?>
```

EXEMPLO CONTA ESPECIAL

CONTA-ESPECIAL.PHP

```php
<?php
     class ContaEspecial extends ContaCorrente{
          private $limite;
          function __construct($valor, $limite){
               parent::__construct($valor);
               $this->limite = $limite;
          }
          function saque($valor){
               if($this->saldo + $this->limite >= $valor)
                    $this->saldo  -= $valor;
          }
          public function __toString(){...}
     }
?>
```

EXEMPLO STATIC

EXEMPLO CLASSE ESTÁTICA

```
<?php
     class Estatica{
          static $var = "Variável Estática";
          static function getStatic(){
               return Estatica::$var;
          }
     }
     echo Estatica::getStatic();
?>
```

NOTA → o operador para acessar métodos ou variáveis estáticas é ::

EXEMPLO FINAL

EXEMPLO CLASSE FINAL

```
<?php
     final class ClasseFinal{
          // essa classe não poderá ser herdada
          final function getFinal(){
          // esse método não poderá ser sobrescrito
               return "Metodo Final";
          }
     }
     $FC = new ClasseFinal();
     echo $FC->getFinal();
?>
```

EXEMPLO ABSTRACT

EXEMPLO CLASSE ABSTRACT

```
<?php
      abstract class Abstrata{
            protected $nome;
            public abstract function setNome($nome);
            public function getNome(){
                  return $this->nome;
            }
      }

      class ClasseAbstrata extends Abstrata{
            public function setNome($nome){
                  $this->nome = $nome;
            }
      }
      $classeAbstrata = new ClasseAbstrata();
      $classeAbstrata->setNome("Pedro");
      echo $classeAbstrata->getNome();
?>
```

INTERFACES

- especifica quais métodos e variáveis outras classes devem implementar, sem definir como serão tratados;
- uma classe pode implementar várias interfaces ou conjuntos de métodos;
- PHP suporta a utilização de interfaces;
- na prática, o uso de interfaces é utilizado para suprir a falta de herança múltipla de linguagens como PHP ou Java;

EXEMPLO INTERFACES

```php
<?php
	interface IPessoa{
		public function setNome($nome);
	}
	interface ITipo{
		public function setTipo($tipo);
	}
	class ClassePessoa implements IPessoa, ITipo{
		private $nome, $tipo;
		public function setNome($nome){
			$this->nome = $nome;
		}
		public function setTipo($tipo){
			$this->tipo = $tipo;
		}
		public function __toString(){
			$retorno  = "";
			$retorno .= "Nome: {$this->nome}<br>";
			$retorno .= "Tipo: {$this->tipo}";
			return $retorno;
		}
	}
?>
```

```php
$IP = new ClassePessoa();
$IP->setNome("Carlos");
$IP->setTipo("Pessoa Física");
echo $IP;
```