UniCesumar
EDUCAÇÃO PRESENCIAL E A DISTÂNCIA

Programação Back End II

SISTEMAS PARA INTERNET

#2022 Ressignificar Realizar Reinventar

AADD

Rafael Alves Florindo

- Mestre em Gestão do Conhecimento nas Organizações.
- Especialista em Desenvolvimento de Sistemas para Web.
- Bacharel em Ciências da Computação.
- Professor de TI desde 2009 pela SEED e desde 2014 pela Unicesumar.

/rafaelflorindo @profrafaelflorindo

Desenvolvimento Back End com OO e BD

Prof. Me. Rafael Florindo

Proposta de cronograma dos encontros:

- **Aula 01: Programação Orientada a Objetos**

Unidade I

- **Aula 02: Manipulação de dados**

Unidade I, IV e V

- **Aula 03: Pilares da Orientação a Objetos**

Unidade I e II.

- **Aula 04: Relacionamentos**

Unidade I e III

Prof. Me. Rafael Florindo
@rafaelflorindo

❖ Aula 01

Unidade 01

Conteúdo

- Programação Orientada a Objetos
  - Classe, Atributo e Método (Concreto, Construtor e Destrutor)
  - Objeto
  - Modificadores de Acesso

Requisitos

- Aula de Estudo de Caso

- Aulas conceituais Unidade I

Prof. Me. Rafael Florindo
@rafaelflorindo

Programação Orientada a Objetos

Paradigma: O termo paradigma pode ser compreendido como um modelo ou padrão a ser seguido para a resolução de um problema.

- Programação Estrutural
- Orientada a Objetos

Prof. Me. Rafael Florindo
@rafaelflorindo

Programação Estrutural

- O **desenvolvimento de software com a abordagem estrutural consiste na construção de um sistema sequencial.**
- Assim, os programas são criados baseados em quais funções, procedimentos e variáveis são necessárias para resolver um problema.

Programação Orientada a Objetos

- A **Orientação a Objetos tem, como principal característica, a forma natural de tratar a realidade**, pois considera que o mundo real é formado por objetos.
- Permite aos **desenvolvedores agruparem tarefas semelhantes em classes.**
- **Construção de módulos independentes ou objetos que podem ser facilmente substituídos**, modificados e reutilizados.

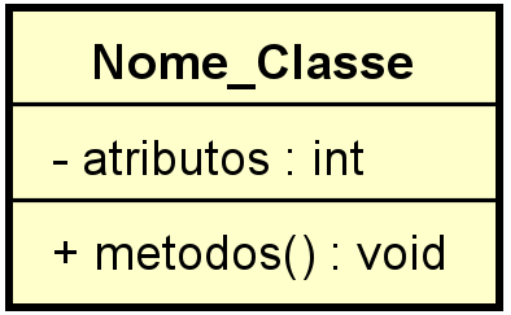
Prof. Me. Rafael Florindo
@rafaelflorindo

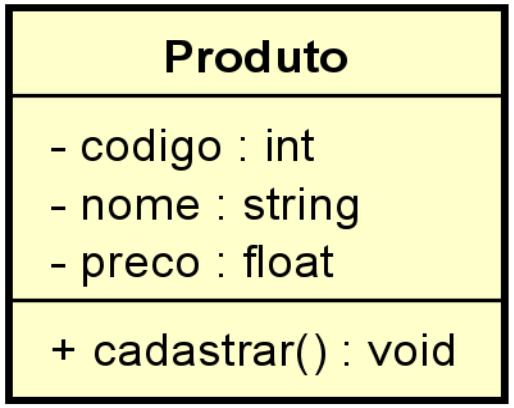
Programação Orientada a Objetos

Classe

A **classe é uma estrutura que definirá um tipo de dado**, podendo conter atributos e, também, métodos para manipular os atributos da classe.

Uma **classe representa a abstração de um conjunto de objetos do mundo real** que possui comportamentos e características comuns (DALL'OGLIO, 2009).



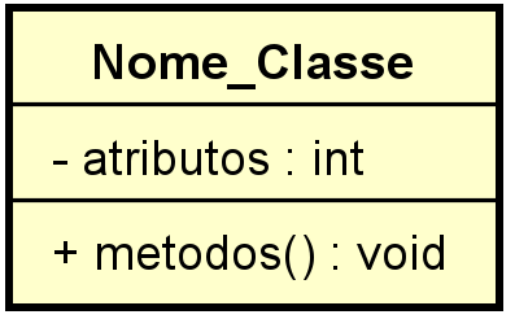


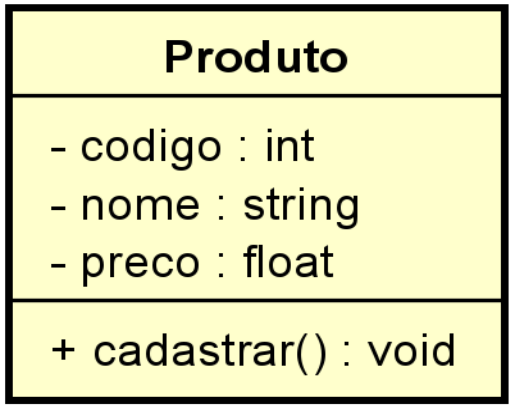
Prof. Me. Rafael Florindo
@rafaelflorindo

Programação Orientada a Objetos

Atributo

Atributo **é uma característica ou propriedade particular que os objetos de uma classe possuem**, precedido da visibilidade e assumindo valores diferentes para cada objeto.



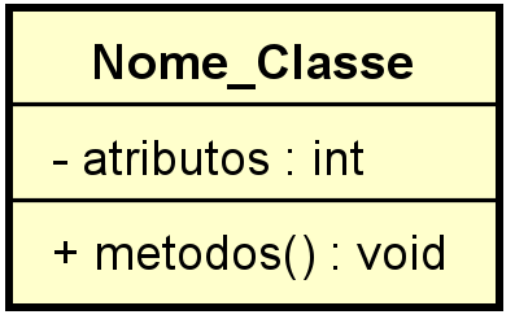


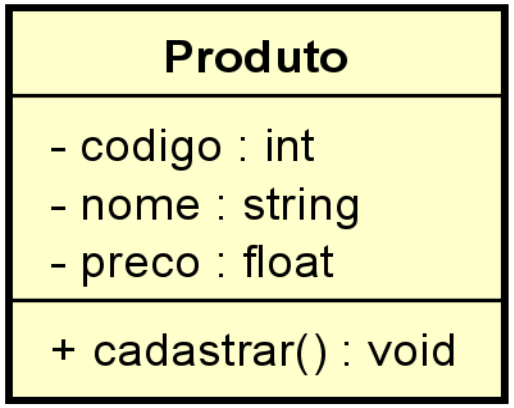
Prof. Me. Rafael Florindo
@rafaelflorindo

Programação Orientada a Objetos

Método

Um **método é considerado um comportamento ou uma funcionalidade específica**, ou seja, deve ser único, de forma que possua apenas uma única funcionalidade. É por esse motivo que os métodos são considerados de responsabilidades das classes.





Prof. Me. Rafael Florindo
@rafaelflorindo

Programação Orientada a Objetos

Objeto

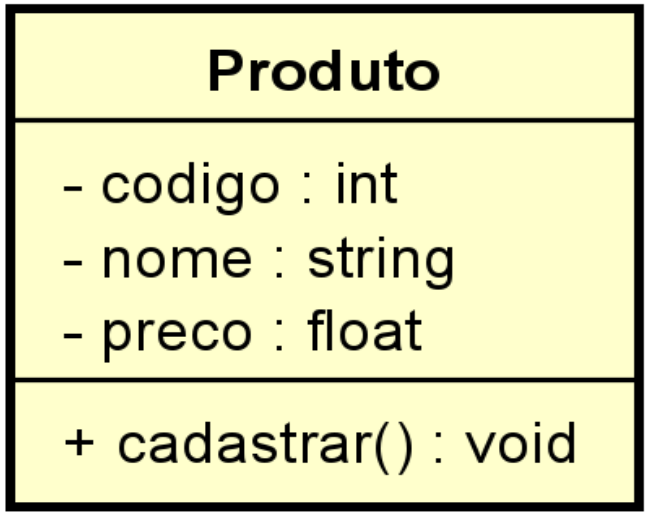
Um objeto é qualquer elemento concreto (físico) ou abstrato (não físico) que existe no mundo real, e possui uma estrutura dinâmica originada com base em uma classe.

Prof. Me. Rafael Florindo
@rafaelflorindo

Programação Orientada a Objetos

Sintaxe da implementação de uma classe em PHP

```
<?php

class Produto
{
	private $codigo;
	private $nome;
	private $preco;

	public function cadastrar()
	{
		//instruções
	}
}
```

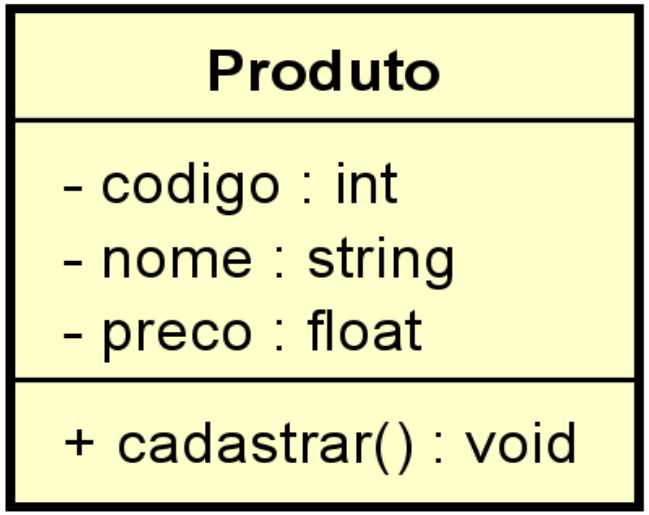


Prof. Me. Rafael Florindo
@rafaelflorindo

Programação Orientada a Objetos

Instanciando e povoando um objeto da classe Produto

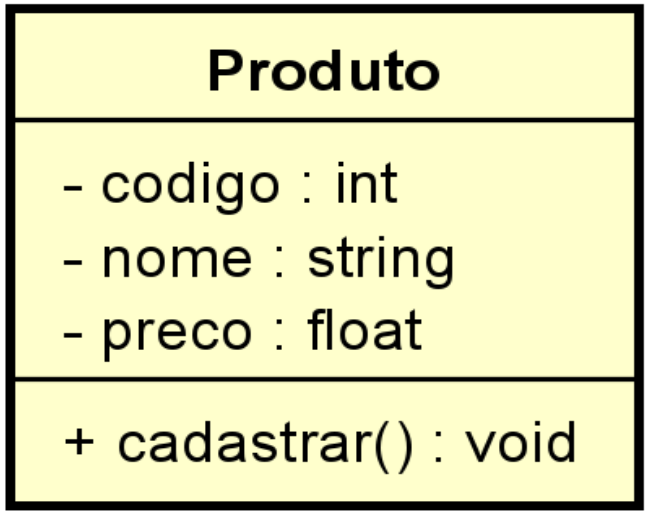
```
<?php
$celular = new Produto();

$celular->codigo = 1;
$celular->nome = "Smartphone";
$celular->preco = 800.55;
```



Prof. Me. Rafael Florindo
@rafaelflorindo

Programação Orientada a Objetos

Método Concretos




```php
<?php
class Produto
{
        private $codigo;
        private $nome;
        private $preco;

        public function cadastrar($codigo, $nome, $preco)
        {
                $this->codigo = $codigo;
                $this->nome = $nome;
                $this->preco = $preco;
        }
}
$celular = new Produto();
$celular->cadastrar(1, "Smartphone", 800.55);
```

Prof. Me. Rafael Florindo
@rafaelflorindo

Programação Orientada a Objetos

Método Construtor

O método construtor **é um método especial, executado na instância da classe pelo operador *new*,** e esse método não produz um valor de retorno, pois estará retornando o próprio objeto.

```
public function __construct([lista de parâmetros])
{
      //instruções
}
```

Prof. Me. Rafael Florindo
@rafaelflorindo

Programação Orientada a Objetos

Método Construtor

```
<?php
class Produto
{
        private $codigo;
        private $nome;
        private $preco;

        public function __construct($codigo, $nome, $preco)
        {
                $this->codigo = $codigo;
                $this->nome = $nome;
                $this->preco = $preco;
        }
}
$celular = new Produto(1, "Smartphone", 800.55);
```

Programação Orientada a Objetos

Método Destrutor

O método destrutor é um outro método especial executado automaticamente quando o objeto é desalocado da memória de forma natural, ou seja, quando terminar as chamadas do objeto à classe ou quando forçamos o PHP a liberar a referência do objeto.

```
public function __destruct()
{
        //instruções
}
```

Prof. Me. Rafael Florindo
@rafaelflorindo

Programação Orientada a Objetos

```php
<?php
class Produto
{
        private $codigo, $nome, $preco;
        public function __construct($codigo, $nome, $preco)
        {
                $this->codigo = $codigo;
                $this->nome = $nome;
                $this->preco = $preco;
        }
        public function __destruct()
        {
                echo "Objeto finalizado!!!";
        }
}
$celular = new Produto(1, "Smartphone", 800.55);
var_dump($celular);
```

Prof. Me. Rafael Florindo
@rafaelflorindo

Programação Orientada a Objetos

Visibilidade dos Atributos

Public: é visível de qualquer lugar do sistema. No diagrama de classes, é identificado pelo símbolo “+”.

Private: é visível apenas de dentro da classe. Não pode ser acessado de fora da classe. No diagrama de classes, é identificado pelo símbolo “-“.

Protected: é visível de dentro da classe e das subclasses herdadas dessa classe, mas não de fora. No diagrama de classes, é identificado pelo símbolo “#”.

Programação Orientada a Objetos

Exemplo sem modificadores de acesso

```
<?php

include("Produto.php");

$celular = new Produto(1, "Smartphone", 800.55);

var_dump($celular);

echo "Codigo : " . $celular->codigo;

echo "Celular : " . $celular->nome;

echo "<br>Preço: " . $celular->preco;
```

Programação Orientada a Objetos

Modificadores de acesso

Com os modificadores de acesso determinamos a visibilidade (*public, private, protected*) de um método ou atributo pertencente a uma classe. Ou seja, definimos se ele pode ou não ser acessado fora da classe em que foi declarado.

```
modificador $atributo;

modificador function metodo() { }
```

Programação Orientada a Objetos

Modificadores de acesso

Para que seja possível acessar os atributos *private* e *protected*, necessitamos de métodos públicos, conhecidos como modificadores de acesso **setters** e **getters**, que setam e resgatam valores, respectivamente.

Prof. Me. Rafael Florindo
@rafaelflorindo

Programação Orientada a Objetos

Modificadores de acesso: *setters*

O método *setter* recebe, por parâmetro, o valor a ser armazenado no atributo da classe.

```
modificador function setNomeDaVariavel(<parâmetro>)
{
        $this-><atributo> = <parâmetro>;
}
```

Programação Orientada a Objetos

Modificadores de acesso: *setters*

```php
<?php
// ....
public function setNome($nome)
{
        $this->nome = $nome;
}

public function setPreco($preco)
{
        $this->nome = $nome;
}
```

Programação Orientada a Objetos

Modificadores de acesso: *getters*

O método *getter* recebe, por parâmetro, o valor a ser resgatado no atributo da classe.

```
modificador function getNomeDaVariavel()

{

return $this-><atributo>;

}
```

Programação Orientada a Objetos

Modificadores de acesso: *getters*

```
<?php
//….
public function getNome()
{
      return $this->nome;
}

public function getPreco()
{
      return $this->preco;
}
```

Prof. Me. Rafael Florindo
@rafaelflorindo

Programação Orientada a Objetos

Exemplo com Modificadores de acesso

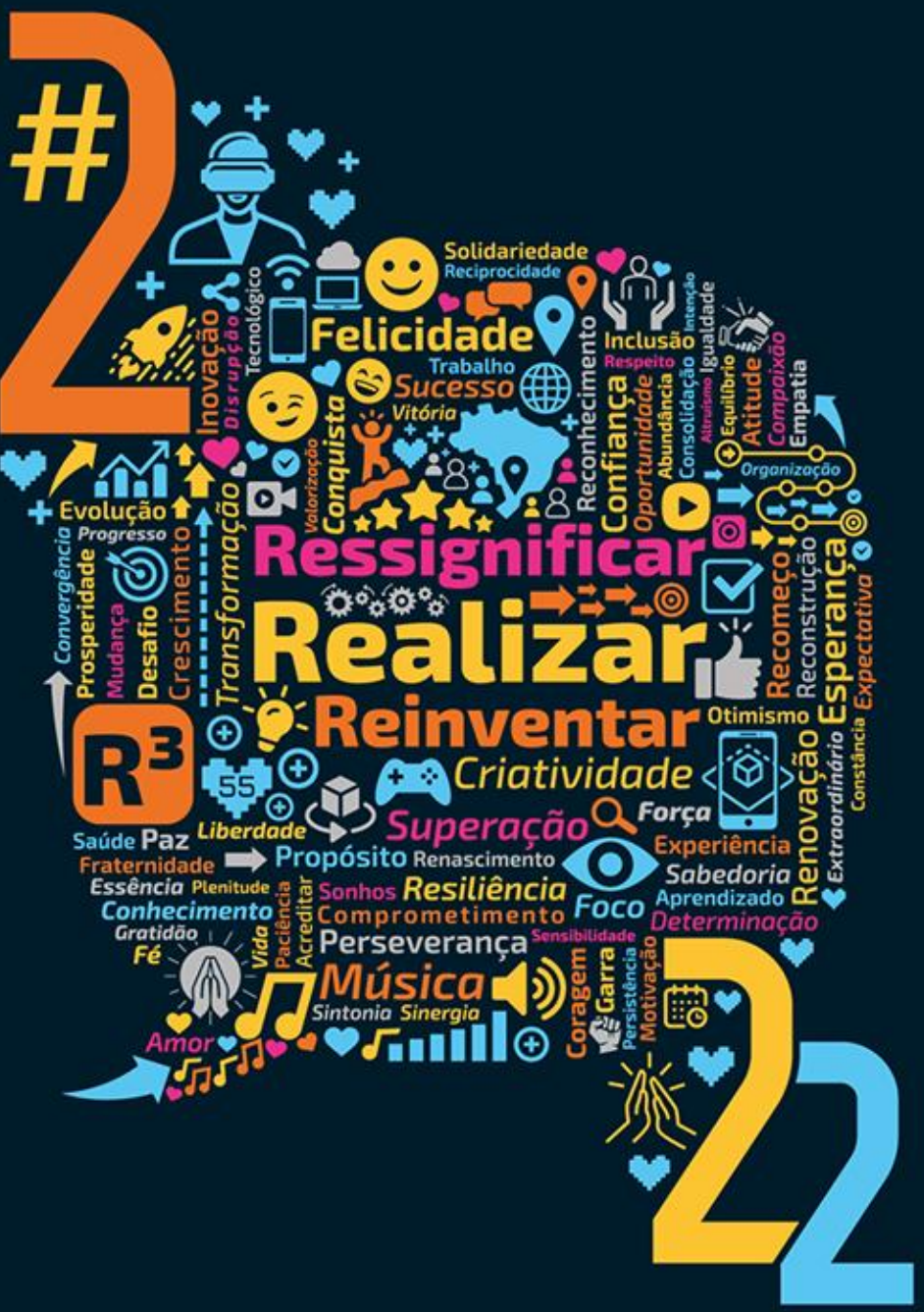
```
<?php
include("Produto.php");

$celular = new Produto(1, "Smartphone", 800.55);
var_dump($celular);

echo "Codigo : " . $celular->codigo;
echo "Celular : " . $celular->getNome();
echo "<br>Preço: " . $celular->getPreco();
```

Prática

Materiais extras

PHP do Jeito Certo: http://br.phptherightway.com/

PHP Standards Recommendations: https://www.php-fig.org/psr/


#2022
Realizar
Ressignificar
Reinventar
R3
Solidariedade
Reciprocidade
Felicidade
Trabalho
Sucesso
Vitória
Inovação
Disrupção
Tecnológico
Inclusão
Respeito
Intenção
Igualdade
Reconhecimento
Confiança
Oportunidade
Abundância
Consolidação
Altruísmo
Equilíbrio
Atitude
Compaixão
Empatia
Organização
Valorização
Conquista
Evolução
Progresso
Transformação
Convergência
Prosperidade
Mudança
Desafio
Crescimento
Recomeço
Reconstrução
Esperança
Expectativa
Extraordinário
Constância
Otimismo
Criatividade
Renovação
Força
Superação
Experiência
Sabedoria
Aprendizado
Foco
Determinação
Sensibilidade
Saúde
Paz
Liberdade
Fraternidade
Propósito
Renascimento
Essência
Plenitude
Sonhos
Resiliência
Comprometimento
Perseverança
Conhecimento
Gratidão
Fé
Vida
Paciência
Acreditar
Música
Sintonia
Sinergia
Amor
Coragem
Garra
Persistência
Motivação
#2022
Realizar
AADD
UniCesumar
EDUCAÇÃO PRESENCIAL E A DISTÂNCIA