vamos aprender

PHP + POO

DigitalHouse > Coding School

Criando classes

escrevemos a palavra reservada **class**

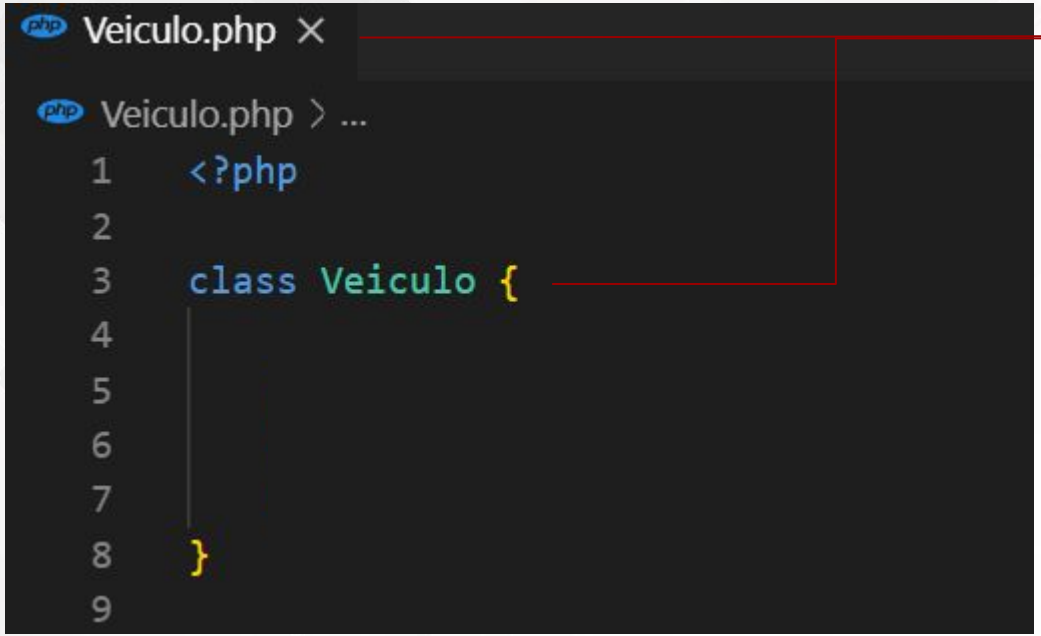
```php
<?php

class Veiculo {

}

```

nunca se esqueça de abrir e fechar as chaves **{ }**

seguido pelo **nome da classe** que queremos criar

DigitalHouse > Coding School

Criando classes



nome do arquivo igual ao **nome da classe**

DigitalHouse > Coding School

Criando classes

Boas práticas na hora de codar:

- ➔ Nomear os arquivos com o ***mesmo nome das classes***
- ➔ Nomear classes como ***substantivo*** e ***iniciando com letra maiúscula*** (não significa que sempre tenha que ser assim okay?)
- ➔ Caso o nome da classe contenha mais que uma palavra cada uma deve ***começar com letra maiúscula***, sem espaços ou hífen separando-as.
  Ex: class **ImagemBackground** { }

Criando classes (atributos)

Veiculo.php

```php
<?php

class Veiculo {

    public $rodas;
    protected $blindagem;
    private $chassi;
}
```

nome do atributo igual ao é uma variável

não se esqueça de dizer qual o **nível de privacidade** (**encapsulamento**)ele terá

DigitalHouse > Coding School

Criando classes (atributos)

Boas práticas na hora de codar:

- Costuma-se deixar os atributos na seguinte ordem: públicos seguidos pelos protegidos e então os privados.

- Declará-los antes dos métodos

Criando classes (métodos)

```php
<?php

class Veiculo {

    public $rodas;
    protected $blindagem;
    private $chassi;

    public function movimentar(){
        echo "vruum!";
    }

    public function buzinar(){
        echo "bi bi!";
    }
}

```

nome do método
lembre-se que é uma função, terá parênteses **( )** e chaves **{ }**

não se esqueça de dizer qual o **nível de privacidade** (**encapsulamento**)ele terá

DigitalHouse > Coding School

Criando classes (métodos)

Boas práticas na hora de codar:

- Precisam ser ***verbos***. **Ex:** corre() | anda()

- Costuma-se iniciá-los com ***letra minúscula***. Caso haja mais de uma palavra, começarem com letra maiúscula, sem uso de espaços ou hífen. **Ex:** getNome()

- Outra prática é pular uma linha após cada método.

Métodos Setters e Getters

Por que utilizar?

- São métodos que nos permitem ***acessar (get)*** e ***modificar (set)*** tributos da nossa classe;

- Precisamos deste acesso por causa do encapsulamento (segurança das propriedades e seus 'níveis de visibilidade');

- É uma forma de ***padronizar o nome destas funcionalidades***, colocar apenas set ou get no nome do método *não faz eles funcionarem* nós quem iremos definir isto dentro da função;

Métodos Setters e Getters

Como fazer Getters:

```
12
13      public function getRodas(){
14          return $this->rodas;
15      }
16
```

$this refere-se a classe em que o método está contido. Assim conseguimos ***acessar atributos e métodos*** dela

DigitalHouse > Coding School

Métodos Setters e Getters

Como fazer Getters:

- Não precisa de parâmetro por iremos buscar o elemento (atributo ou método) da classe através da variável **$this;**

- Precisamos utilizar a palavra reservada **return** para entregarmos o atributo “requisitado”;

Métodos Setters e Getters

Como fazer Setters:

variável (**parâmetro**) que irá **alterar valor do atributo**

```
8
9       public function setRodas($valorRoda){
10          $this->rodas = $valorRoda;
11      }
12
```

$this refere-se a classe em que o método está contido. Assim conseguimos ***acessar atributos e métodos*** dela

DigitalHouse > Coding School

Métodos Setters e Getters

Como fazer Setters:

- Precisamos de parâmetro para recebermos o novo valor que o atributo terá **$this;**

- A função será ***void***. Não precisamos retornar nada para o sistema ou cliente, por isso basta apenas alterarmos o valor e encerrarmos a função alí.

Criando Objetos (instância de uma classe)

nome da classe instanciada

palavra reservada **new**

variável que representará nosso **objeto**

DigitalHouse > Coding School

Criando Objetos (instância de uma classe)

```
18
19    $carro = new Veiculo();
20    $carro->rodas = 4;
21
22
```

os **atributos**,apesar de serem variáveis na classe, não precisam de **$** para acessá-los

com esta seta **—>** acessamos as ***propriedades*** e ***métodos*** do nosso objeto (as mesmas que existem na classe que foi instanciada)

não se esqueça de sempre ***chamar o objeto*** quando precisar acessar algo dele

DigitalHouse > Coding School

Até a próxima!