LUCAS JOHANN Lógica.
de. Programaçao.

indice.

- 1. O QUE É POO
- 2. VANTAGENS
- 3. O QUE SÃO CLASSES
- 4. O QUE SÃO MÉTODOS
- 5. HERANÇA
- 6. POLIMORFISMO
- 7. ENCAPSULAMENTO

F Programação Orientada a Objetos - POO

Paradigma de programação baseado em objetos, onde cada objeto possui atributos (características) e métodos (realização de ações).



ENCAPSULAMENTO

Permite esconder detalhes internos de uma classe, expondo somente o necessário.

REUTILIZAÇÃO DE CÓDIGO

MODELAGEM DO MUNDO REAL

Permite representar entidades do mundo real como objetos.



Atributos e métodos



Atributos são as características que um elemento possui, os métodos são as ações que este objeto pode executar.





Seria o "modelo" para criação do objeto, ela descreve como os objetos devem ser construídos.

```
class Carro {
  constructor(cor, modelo, marca) {
    this.cor = cor;
    this.modelo = modelo;
    this.marca = marca;
  acelerar() {
    console.log("0 carro está acelerando.");
  frear() {
    console.log("O carro está freando.");
  virar(direcao) {
    console.log("0 carro está virando para " + direcao + ".");
const meuCarro = new Carro('vermelho', 'Fusca', 'Volkswagen');
meuCarro.acelerar(); // 0 carro está acelerando.
meuCarro.frear(); // 0 carro está freando
meuCarro.virar('direita'); // 0 carro está virando para
```





É quando uma classe filha/subclasse herda as propriedades e métodos de outra classe, chamada classe pai ou superclasse.

```
class Carro {
 constructor(cor, modelo, marca) {
    this.cor = cor;
    this.modelo = modelo;
    this.marca = marca;
  acelerar() {
    console.log("O carro está acelerando.");
  frear() {
    console.log("O carro está freando.");
  virar(direcao) {
    console.log("O carro está virando para " + direcao + ".");
class CarroEsportivo extends Carro {
  constructor(cor, modelo, marca, potencia) {
    super(cor, modelo, marca); // Chama o construtor da classe pai
    this.potencia = potencia; // Propriedade adicional
  acelerar() {
    console.log("O carro esportivo está acelerando rapidamente!");
const meuCarro = new Carro('azul', 'Fusca', 'Volkswagen');
meuCarro.acelerar(); // 0 carro está acelerando.
const meuCarroEsportivo = new CarroEsportivo('vermelho', 'Ferrari',
'Ferrari', '500HP');
meuCarroEsportivo.acelerar(); // 0 carro esportivo está acelerando
meuCarroEsportivo.frear(); // 0 carro está freando.
meuCarroEsportivo.virar('esquerda'); // 0 carro está virando para
```





Permite a reutilização do método em diferentes objetos.

```
class Carro {
 constructor(cor, modelo, marca) {
   this.cor = cor;
   this.modelo = modelo;
   this.marca = marca;
  acelerar() {
   console.log("O carro está acelerando.");
  frear() {
   console.log("O carro está freando.");
 virar(direcao) {
   console.log("O carro está virando para " + direcao + ".");
class CarroEsportivo extends Carro {
 acelerar() {
   console.log("O carro esportivo está acelerando rapidamente!");
  frear() {
   console.log("O carro esportivo está freando com precisão.");
class CarroFamilia extends Carro {
 acelerar() {
   console.log("0 carro de família está acelerando de forma segura.");
  frear() {
   console.log("O carro de família está freando suavemente.");
 virar(direcao) {
   console.log("O carro de família está virando para " + direcao + " com cuidado.");
const meuCarroEsportivo = new CarroEsportivo('azul', 'Ferrari');
meuCarroEsportivo.acelerar(); // 0 carro esportivo está acelerando rapidamente!
meuCarroEsportivo.frear(); // O carro esportivo está freando com precisão.
meuCarroEsportivo.virar('direita'); // 0 carro está virando para direita.
const meuCarroFamilia = new CarroFamilia('preto', 'Voyage', 'Volkswagen');
meuCarroFamilia.acelerar(); // O carro de família está acelerando de forma segura.
meuCarroFamilia.frear(); // O carro de família está freando suavemente.
meuCarroFamilia.virar('esquerda'); // O carro de família está virando para esquerda com
```



Encapsulamento



Permite "esconder" detalhes de uma classe e expor apenas quando necessário, mantendo a integridade do objeto.

•••
<pre>class Carro { #cor; // Propriedade privada #modelo; // Propriedade privada #marca; // Propriedade privada</pre>
<pre>constructor(cor, modelo, marca) { this.#cor = cor; this.#modelo = modelo; this.#marca = marca; }</pre>
<pre>// Métodos públicos para acessar propriedades privadas getCor() { return this.#cor; }</pre>
<pre>setCor(cor) { this.#cor = cor; }</pre>
<pre>getModelo() { return this.#modelo; }</pre>
<pre>setModelo(modelo) { this.#modelo = modelo; }</pre>
<pre>getMarca() { return this.#marca; }</pre>
<pre>setMarca(marca) { this.#marca = marca; }</pre>
<pre>acelerar() { console.log("O carro está acelerando."); }</pre>
<pre>frear() { console.log("O carro está freando."); }</pre>
<pre>virar(direcao) { console.log("0 carro está virando para " + direcao + "."); } }</pre>
<pre>// Exemplo de uso const meuCarro = new Carro('vermelho', 'Fusca', 'Volkswagen');</pre>
<pre>// Acessando propriedades através de métodos públicos console.log(meuCarro.getCor()); // vermelho</pre>
<pre>meuCarro.setCor('azul'); console.log(meuCarro.getCor()); // azul</pre>
<pre>console.log(meuCarro.getModelo()); // Fusca meuCarro.setModelo('Gol'); console.log(meuCarro.getModelo()); // Gol</pre>
<pre>meuCarro.acelerar(); // 0 carro está acelerando.</pre>



Exercícios

