

02

JavaScript Orientado a Objetos



String
Boolean
Number
Array

Variáveis

for
while
do...while

Laços de
Repetição

NO CAPÍTULO ANTERIOR

Introdução a Lógica de Programação

Funções

parâmetros
anônimas

Funções
dos Arrays

.forEach
.map
.filter



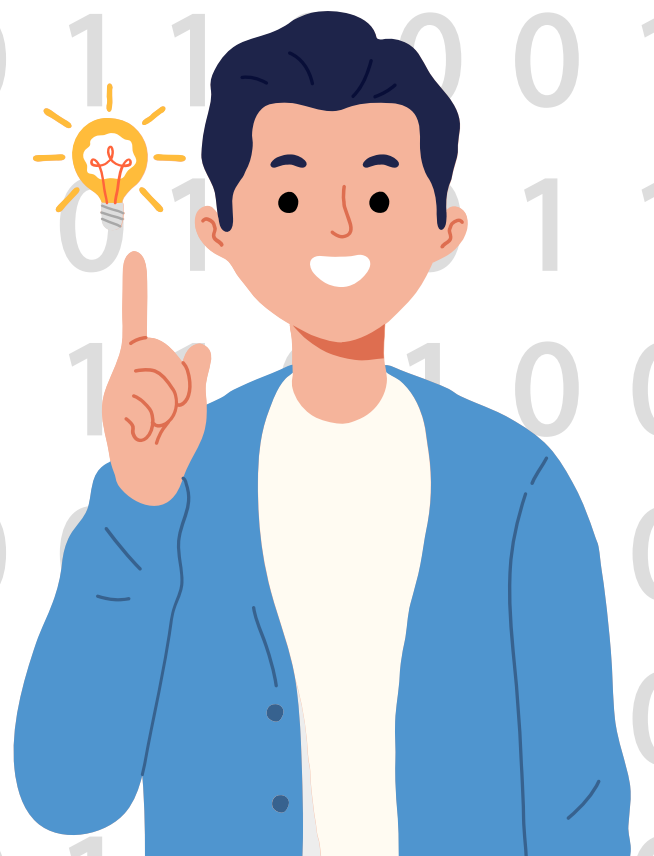
```
var nomeAluno = 'João'  
var matricula = '12345'  
var notas = [10, 4, 6]
```



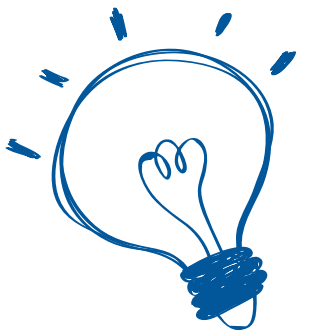
```
notas.forEach(function(nota){  
  console.log(nota)  
})
```



```
function calcularMedia(notas)  
{  
  var soma = 0  
  notas.forEach(function(nota){  
    soma = soma+nota  
  })  
  
  return (soma/notas.length)  
}
```



Objetos

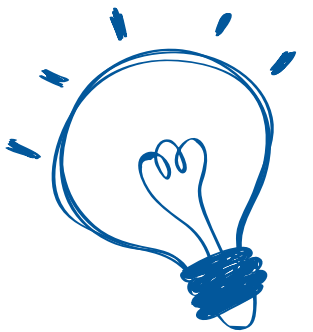


Conjunto de atributos aninhados a uma variável denomina-se um objeto.

Ele agrega diversos valores e permite armazenar e recuperar esses valores pelo nome.



Objetos



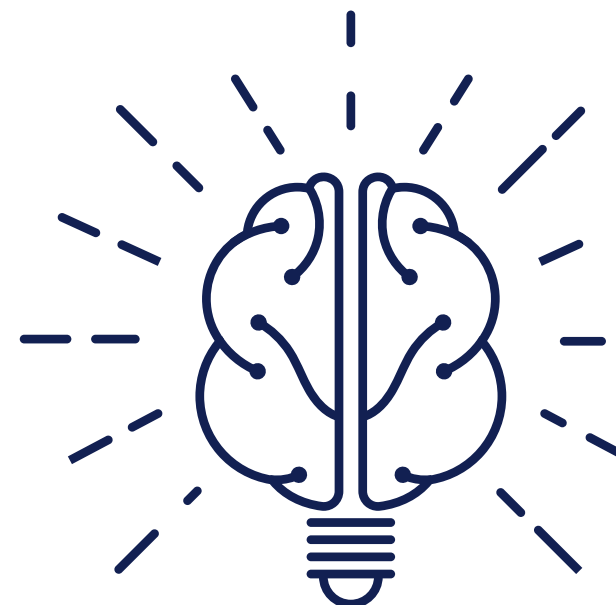
```
var pessoa = [ 'João', '33', 'Manaus' ]
```

```
console.log(pessoa[0])
```

```
console.log(pessoa[1])
```

```
console.log(pessoa[2])
```

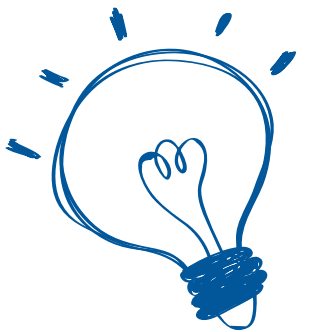
Arrays são valores compostos, ordenados e possuem índices



Objetos

```
var pessoa = {  
  nome: 'João',  
  idade: '33',  
  cidade: 'Manaus'  
}  
  
console.log(pessoa.nome)
```

Um objeto é um valor composto, não ordenado e possui propriedades.
Ele agrega diversos valores e permite armazenar e recuperar esses valores pelo nome.







Objetos

Crie um sistema onde deverá ter o cadastro de 2 alunos, deve conter nome, matricula e 3 notas.

Uma função que irá calcular as notas e no final do sistema deve informar qual aluno teve a maior média



Crie um sistema onde deverá ter o cadastro de 2 alunos, deve conter nome, matricula e 3 notas. Uma função quer irá calcular as notas e no final do sistema deve informar qual aluno teve a maior média

```
//noprotect
var aluno_01 = {
  nome: 'João',
  matricula: '123123',
  notas: [7,4,3],
}

var aluno_02 = {
  nome: 'Maria',
  matricula: '123124',
  notas: [5,3,2],
}

function calcularMedia(notas)
{
  var soma = 0
  notas.forEach(function(nota){
    soma = soma+nota
  })

  return (soma/notas.length)
}

var media_aluno01 = calcularMedia(aluno_01.notas)
var media_aluno02 = calcularMedia(aluno_02.notas)

if (media_aluno01 > media_aluno02)
{
  console.log(aluno_01.nome+ " tirou a maior nota")
}else{
  console.log(aluno_02.nome+ " tirou a maior nota")
}
```



Objetos



```
var pessoa = {  
  nome: 'Gabriel',  
  sobrenome: 'Faria',  
  idade: '33',  
  cidade: 'Manaus',  
  nomeCompleto(){  
    return this.nome + " " + this.sobrenome  
  }  
}  
  
console.log(pessoa.nomeCompleto())
```

Um objeto é um valor composto, não ordenado e possui propriedades.
Ele agrega diversos valores e permite armazenar e recuperar esses valores pelo nome.





Objetos

Crie um objeto "Pessoa". Ele deve conter nome, idade, estado civil, nacionalidade

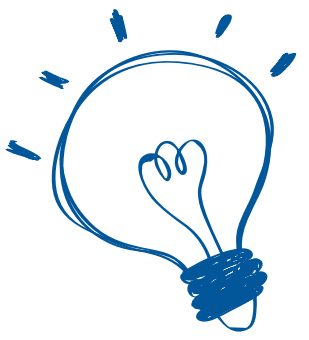
Crie um objeto "Pessoa com endereço". Ele deve conter nome, idade, estado civil, nacionalidade e endereço.

Endereço deve conter rua, numero, cidade, cep.

Agora crie uma função no objeto que retorne o endereço completo da pessoa.



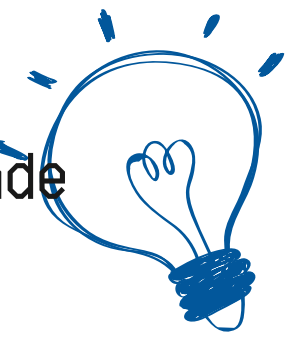
Crie um objeto "Pessoa". Ele deve conter nome, idade, estado civil, nacionalidade



```
var pessoa = {  
  nome: 'Gabriel',  
  idade: '33',  
  estadoCivil: 'Casado',  
  nacionalidade: 'Brasileiro',  
}
```



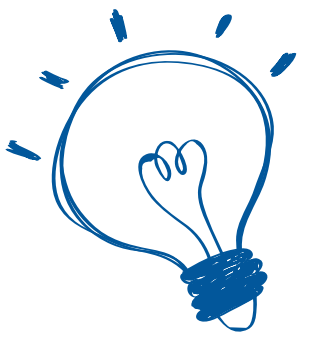
Crie um objeto "Pessoa com endereço". Ele deve conter nome, idade, estado civil, nacionalidade e endereço. Endereço deve conter rua, numero, cidade, cep.



```
var pessoa = {  
  nome: 'Gabriel',  
  idade: '33',  
  estadoCivil: 'Casado',  
  nacionalidade: 'Brasileiro',  
  endereco: {  
    rua: 'rua a',  
    numero: '111',  
    cidade: 'Manaus',  
    cep: '023823'  
  }  
}
```



Agora crie uma função no objeto que retorne o endereço completo da pessoa.

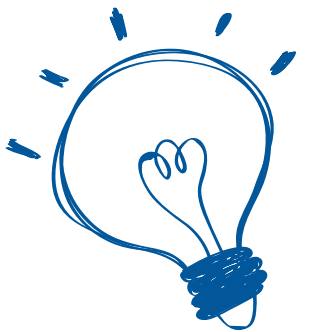


```
var pessoa = {
  nome: 'Gabriel',
  idade: '33',
  estadoCivil: 'Casado',
  nacionalidade: 'Brasileiro',
  endereco: {
    rua: 'rua a',
    numero: '111',
    cidade: 'Manaus',
    cep: '023823'
  },
  enderecoCompleto(){
    return "Cep: "+this.endereco.cep + ", Rua: " +this.endereco.rua + ", Nº: " +this.endereco.numero+
    ", Cidade: "+this.endereco.cidade
  }
}

console.log(pessoa.enderecoCompleto())
```



Programação Orientada a Objetos



É uma forma de modelar um sistema como uma coleção de objetos, onde cada objeto representa algum aspecto particular do sistema. Esses objetos contêm dados (atributos) e funções (métodos).

☒ Classe

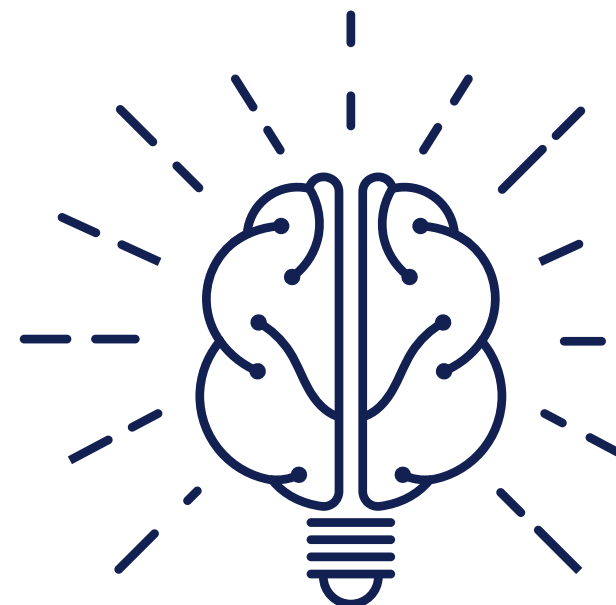
☒ Atributos

☒ Herança

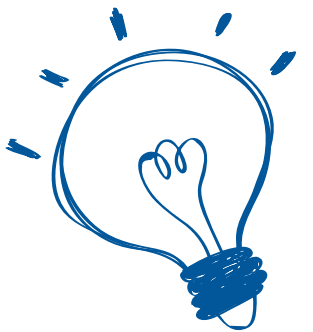
☒ Métodos

☒ Encapsulamento

☒ Abstração



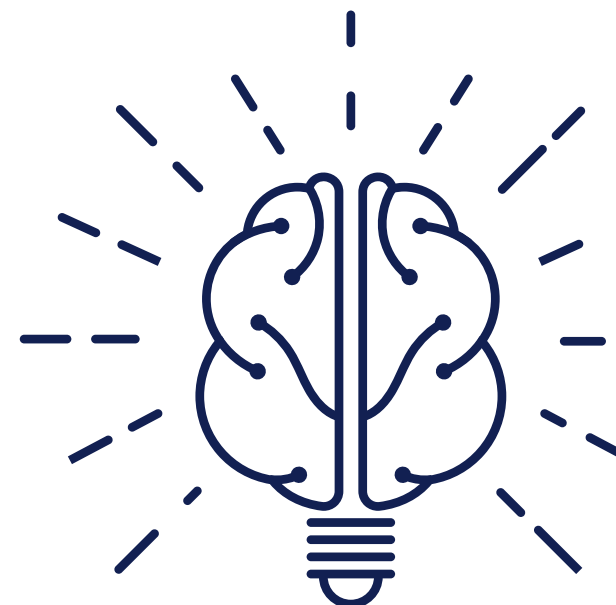
Programação Orientada a Objetos



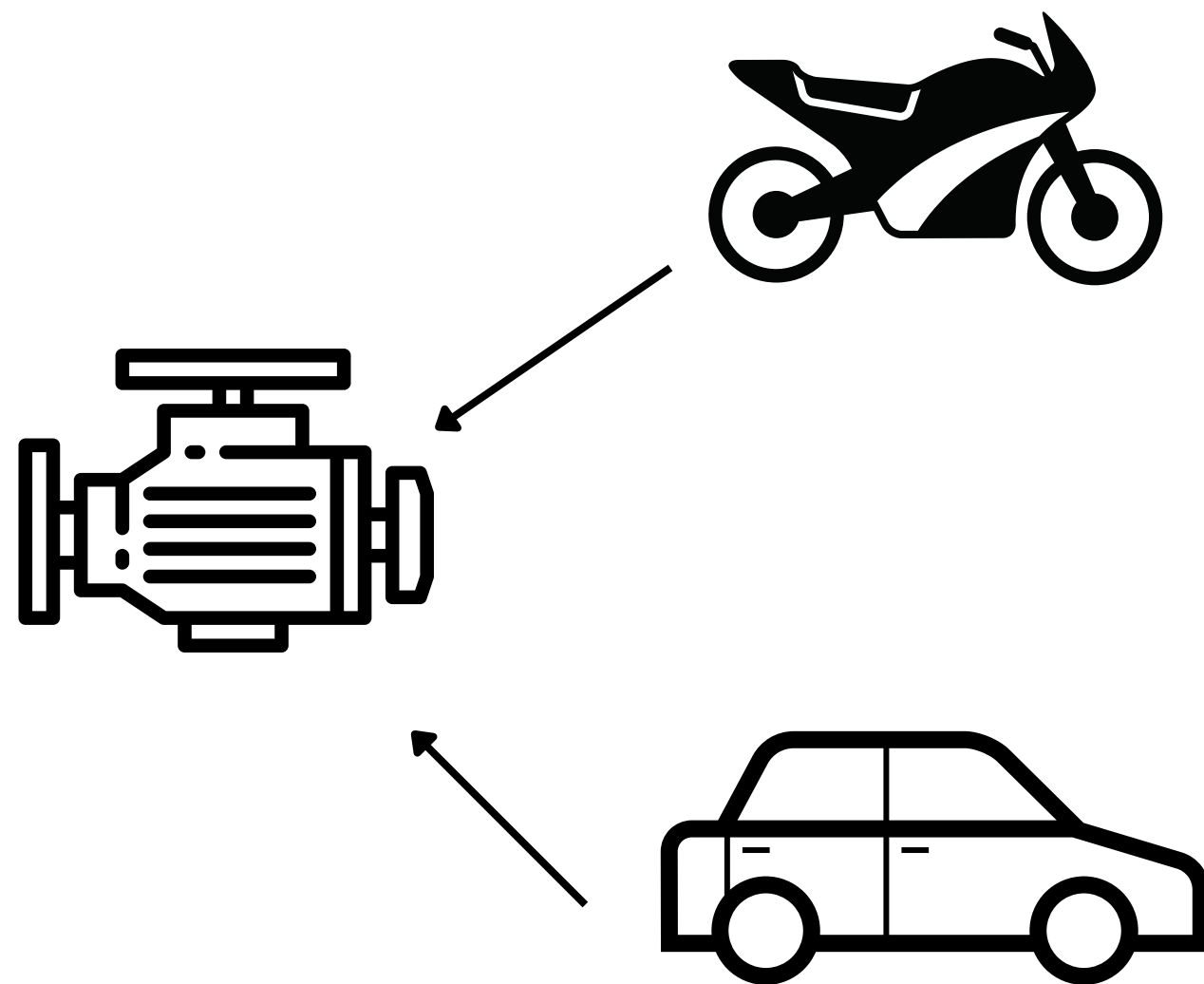
Pessoa

```
var pessoa = {  
  nome: 'Gabriel',  
  idade: '33',  
  estadoCivil: 'Casado',  
  nacionalidade: 'Brasileiro',  
  endereco: {  
    rua: 'rua a',  
    numero: '111',  
    cidade: 'Manaus',  
    cep: '023823'  
  }  
}
```

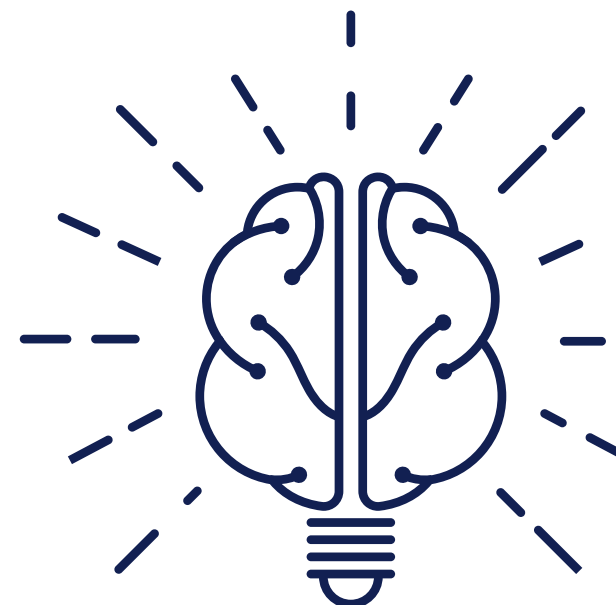
Uma abstração de dados consiste de um conjunto de dados e funções
que caracterizam o comportamento dos objetos



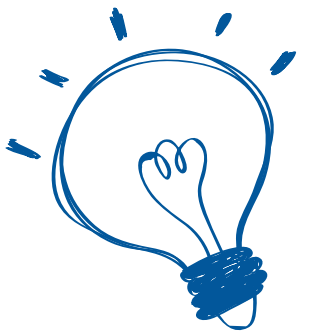
Programação Orientada a Objetos



```
var veiculo = {  
  rodas: '...',  
  modelo: '...',  
  andar() {},  
  parar() {}  
}  
  
var carro = {  
  rodas: 4,  
  modelo: 'Siena',  
  andar() {},  
  parar() {}  
}  
  
var moto = {  
  rodas: 2,  
  modelo: 'Yamaha Fazer',  
  andar() {},  
  parar() {}  
}
```

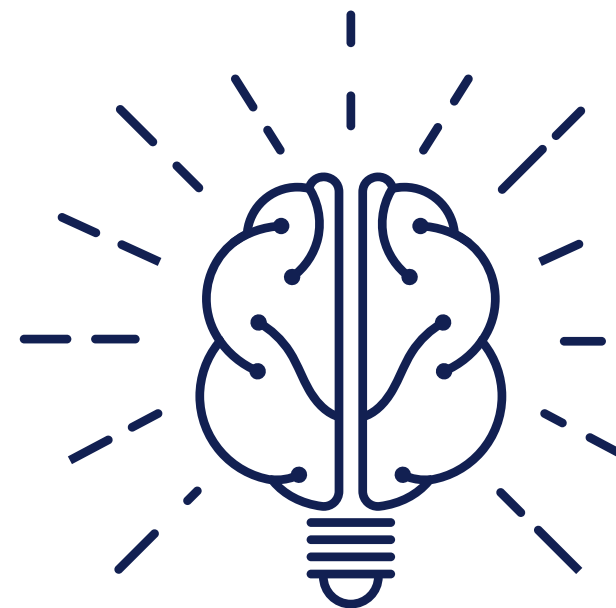


Programação Orientada a Objetos



```
var veiculo = {  
  rodas: '...',  
  modelo: '...',  
  andar() {},  
  parar() {}  
}  
  
var carro = {  
  rodas: 4,  
  modelo: 'Siena',  
  andar() {},  
  parar() {}  
}  
  
var moto = {  
  rodas: 2,  
  modelo: 'Yamaha Fazer',  
  andar() {},  
  parar() {}  
}
```

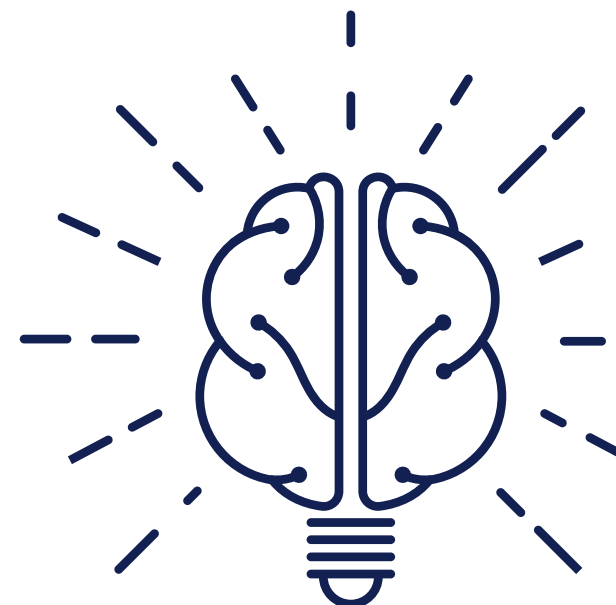
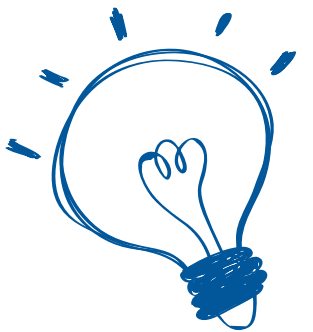
```
class veiculo {  
  constructor(rodas, modelo)  
  {  
    this.rodas = rodas  
    this.modelo = modelo  
  }  
  
  andar() {}  
  freiar() {}  
}
```



Programação Orientada a Objetos

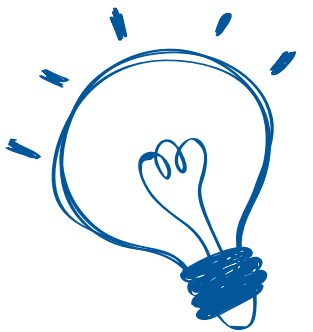
Class

```
class veiculo {  
  constructor(rodas, modelo)  
  {  
    this.rodas = rodas  
    this.modelo = modelo  
  }  
  
  andar(){}  
  freiar(){}  
}
```



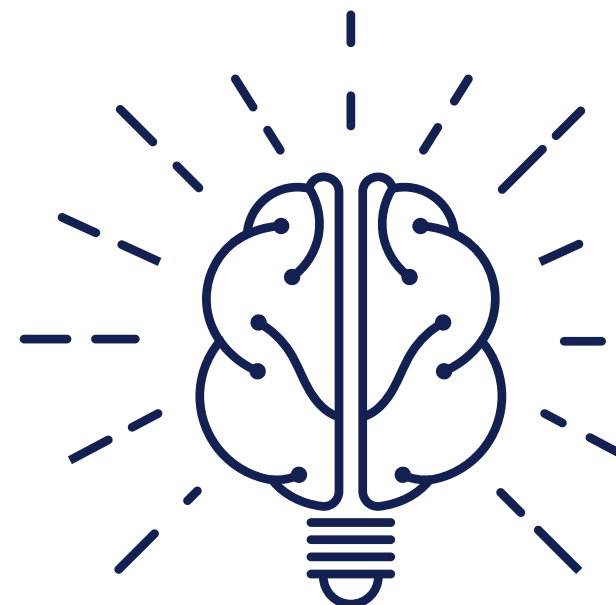
Programação Orientada a Objetos

Atributos Propriedades



Um atributo é uma variável que pertence a um objeto. Os dados de um objeto são armazenados nos seus atributos.

```
class veiculo {  
  rodas;  
  modelo;  
  
  constructor(rodas, modelo)  
  {  
    this.rodas = rodas;  
    this.modelo = modelo;  
  }  
  
  andar(){}  
  freiar(){}  
}
```

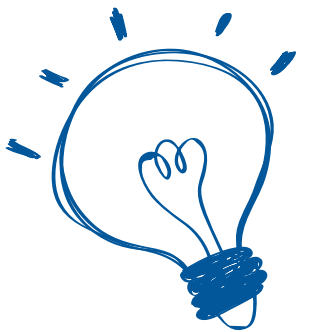


Programação Orientada a Objetos

Constructor

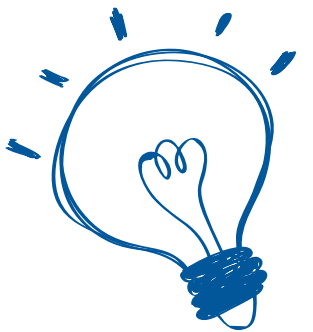
É a função que inicia a classe com os parâmetros necessário para funcionar.

```
class veiculo {  
    rodas;  
    modelo;  
  
    constructor(rodas, modelo)  
    {  
        this.rodas = rodas;  
        this.modelo = modelo;  
    }  
  
    andar(){}  
    freiar(){}  
}
```



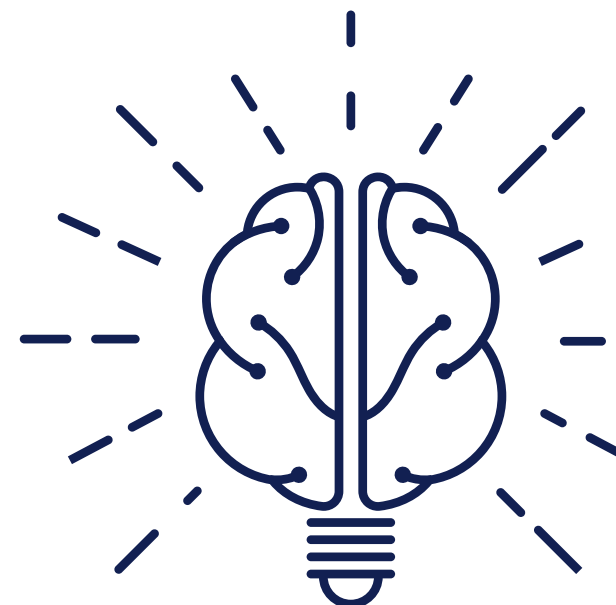
Programação Orientada a Objetos




Métodos Comportamento




Métodos são as ações que podem ser realizadas.
As ações são feitas por essas funções.

```
class veiculo {  
    rodas;  
    modelo;  
  
    constructor(rodas, modelo)  
    {  
        this.rodas = rodas;  
        this.modelo = modelo;  
    }  
  
    andar(){}  
    freiar(){}  
}
```



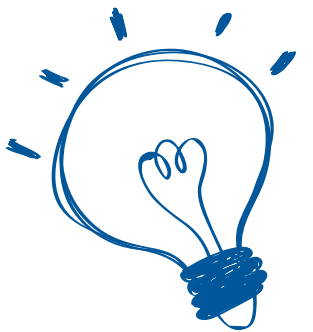


Crie uma class de veículo com quantidade de rodas, modelo, tipo de veículo, e funções para andar e parar.



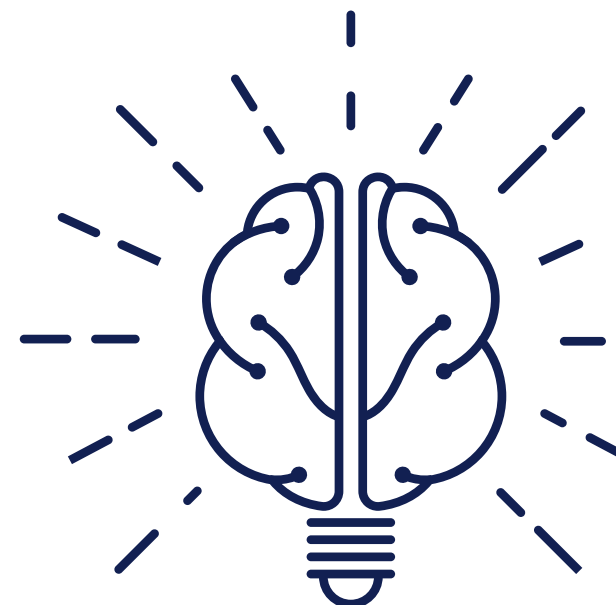
Programação Orientada a Objetos

Instanciando

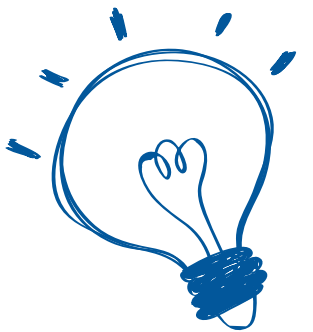


Para instanciar adicionamos à classe a uma variável utilizando:
new NomeClasse(parametros);
Podemos alterar os valores ou chamar métodos da mesma forma que os objetos.

```
class Veiculo {  
  tipo;  
  rodas;  
  modelo;  
  constructor(rodas,modelo,tipo)  
  {  
    this.tipo = tipo  
    this.rodas = rodas  
    this.modelo = modelo  
  }  
  
  andar(){  
    return 'Carro está andando...'  
  }  
  freiar(){}  
}  
  
var carro = new Veiculo('4','Fiat','Carro')  
var moto = new Veiculo('2','Yamaha','Moto')  
  
console.log(carro)  
  
carro.modelo = 'Chevrolet'  
  
console.log(carro)
```



Programação Orientada a Objetos



Encapsulamento

Se alguns desses atributos ou métodos forem facilmente visíveis e modificáveis, como o mecanismo de ligar o carro, isso pode dar liberdade para que alterações sejam feitas, resultando em efeitos colaterais imprevisíveis.

```
class Veiculo {
  tipo;
  rodas;
  modelo;
  #partida;

  constructor(rodas, modelo, tipo)
  {
    this.tipo = tipo
    this.rodas = rodas
    this.modelo = modelo
    this.#partida = false
  }

  andar(){
    console.log('Carro está andando...')
  }

  ligarVeiculo()
  {
    this.#partida = true
  }

  desligarVeiculo()
  {
    this.#partida = false
  }

  getPartida()
  {
    return this.#partida
  }
}

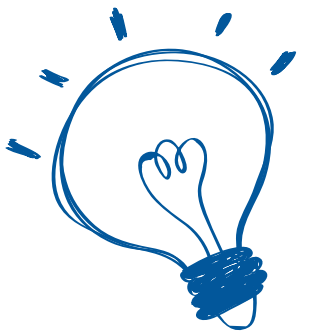
var carro = new Veiculo('4', 'Fiat', 'Carro')
var moto = new Veiculo('2', 'Yamaha', 'Moto')

carro.partida = true

console.log(carro)
console.log("O veículo está: "+carro.getPartida())
```



Programação Orientada a Objetos



Herança

No nosso exemplo, você acabou de comprar um carro com os atributos que procurava. Apesar disso ser único, existem outros objetos com algumas ações e atributos que se assemelham com os do seu carro. Como no caso de uma moto. A moto e o carro aceleram e ligam o farol, portanto os métodos e atributos podem ser reaproveitados.

```
//noprotect

class Veiculo {
  tipo;
  rodas;
  modelo;
  #partida;

  constructor(rodas,modelo,tipo)
  {
    this.tipo = tipo
    this.rodas = rodas
    this.modelo = modelo
    this.#partida = false
  }

  andar(){
    console.log('Carro está andando...')
  }

  ligarVeiculo()
  {
    this.#partida = true
  }

  desligarVeiculo()
  {
    this.#partida = false
  }

  getPartida()
  {
    return this.#partida
  }
}

class Carro extends Veiculo {
  constructor(rodas,modelo,tipo,nome){
    super(rodas,modelo,tipo);
    this.nome = nome;
  }
}

class Moto extends Veiculo {
  cilindrada;
}

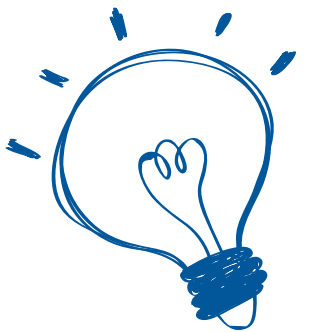
var meuCarro = new Carro('4','Fiat','Carro','Marea')

var minhaMoto = new Moto('2','Yamaha','Moto')
minhaMoto.cilindrada = 125

console.log(meuCarro)
console.log(minhaMoto)
```



Programação Orientada a Objetos



Crie um projeto de estoque de produtos: produtos deve ter marca, modelo, preco, estoque, o estoque deverá ser encapsulado. Irá cadastrar 3 tipos de produto:

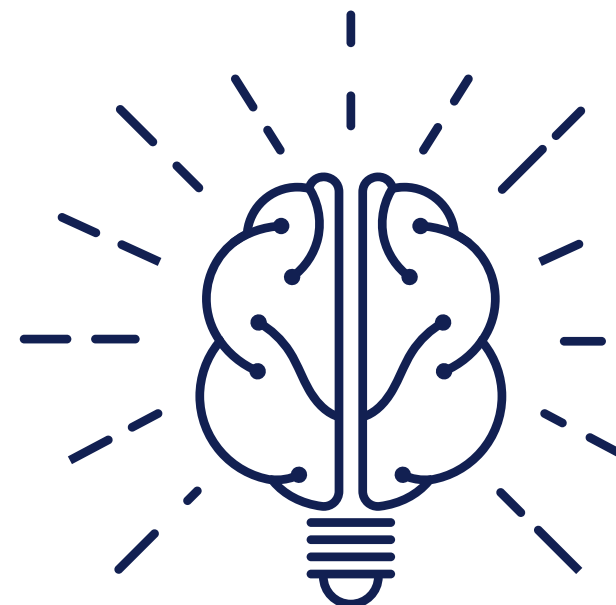
Notebook, Geladeira, Televisão.

precisará armazer a quantidade de ram do notebook,

Televisão deverá informar se é smart ou não,

Geladeira deverá informar se é FrostFree ou não.

No final mostrar todos os produtos cadastrados.



```
// noprotect
```

```
class Produto {
  marca;
  modelo;
  preco;
  #estoque = 0;
  #ano;

  constructor(marca, modelo, preco) {
    this.marca = marca;
    this.modelo = modelo;
    this.preco = preco;
  }

  getAno(ano) {
    return this.#ano;
  }

  setAno(ano) {
    this.#ano = ano;
  }

  adicionarEstoque() {
    this.#estoque++;
  }

  retirarDoEstoque() {
    this.#estoque--;
  }

  estoqueAtual() {
    return this.#estoque;
  }
}

class Notebook extends Produto {
  quantidadeRam;
}

class Televisao extends Produto {
  isSmart = false;
}

class Geladeira extends Produto {
  frostFree = false;
}

class MaquinaDeLavar extends Produto {
  capacidade;
}
```

```
var produtos = [];
while(confirm("Deseja cadastrar um produto?")) {
  var opcao = prompt("Escolha um produto para cadastrar:\n 0 - Notebook \n 1 - Geladeira \n 2 - Máquina de Lavar")
  switch (opcao) {
    case '0':
      var marca = prompt("Digite a marca");
      var modelo = prompt("Digite o modelo");
      var preco = parseFloat(prompt("Digite o preco"));
      var notebook = new Notebook(marca, modelo, preco)
      produtos.push(notebook)
      break;
    case '1':
      var marca = prompt("Digite a marca");
      var modelo = prompt("Digite o modelo");
      var preco = parseFloat(prompt("Digite o preco"));
      var geladeira = new Geladeira(marca, modelo, preco)
      if(confirm("Adicionar Estoque?")){
        geladeira.adicionarEstoque()
      }
      produtos.push(geladeira)
      break;
    case '2':
      console.log('Mangoes and papayas are $2.79 a pound. ');
      break;
    default:
      break;
  }
  console.log(produtos)
}

produtos.forEach((produto) => {
  var quantidade = produto.estoqueAtual();
  if (quantidade == 0) {
    alert("0 produto " + produto.marca + " está sem estoque");
  }
});
```