Classes e Atributos

DEVinHouse

Parcerias para desenvolver a sua carreira

SENAI

<LAB365>

AGENDA

- Revisão
- Programação Orientada à Objetos
- Classes e Atributos

O que é Programação Orientada a Objetos ou POO?

POO ou OOP é uma forma de paradigma de programação. E o que é um paradigma de programação? É uma forma de se programar.

No dia a dia como um desenvolvedor front-end por exemplo, trabalhamos utilizando a programação estruturada, que é composta de sequências, condições(if, else, etc) e repetição(for, while, etc).

O que é Programação Orientada a Objetos ou POO?

Mas o que é o POO finalmente? É uma forma de se programar utilizando classes e objetos, como o próprio nome sugere.

POO X Programação Estruturada

A diferença principal é que na programação estruturada, um programa é tipicamente escrito em uma única rotina (ou função) podendo, é claro, ser quebrado em subrotinas. Mas o fluxo do programa continua o mesmo.

Além disso, o acesso às variáveis não possuem muitas restrições na programação estruturada.

POO X Programação Estruturada

Esse novo paradigma se baseia principalmente em dois conceitos chave: classes e objetos. Todos os outros conceitos, igualmente importantes, são construídos em cima desses dois.

O que são classes e objetos?

Imagine que você comprou um carro recentemente e decide modelar esse carro usando programação orientada a objetos. O seu carro tem as características que você estava procurando: um motor 2.0, azul escuro, quatro portas, câmbio automático etc.

O que são classes e objetos?

Ele também possui comportamentos que, provavelmente, foram o motivo de sua compra, como acelerar, desacelerar, acender os faróis, buzinar e tocar música. Podemos dizer que o carro novo é um objeto, onde suas características são seus atributos (dados atrelados ao objeto) e seus comportamentos são ações ou métodos.

O que são classes e objetos?

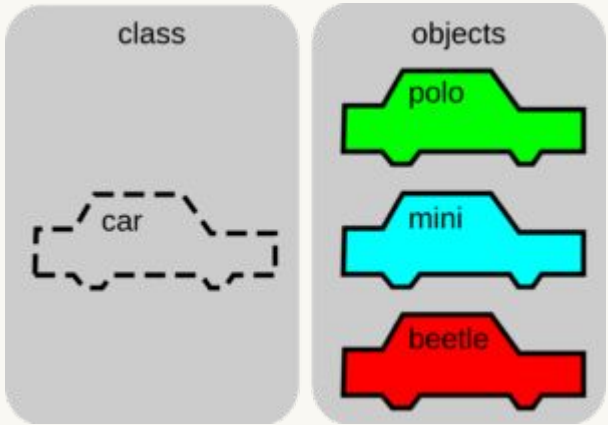
Seu carro é um objeto seu mas na loja onde você o comprou existiam vários outros, muito similares, com quatro rodas, volante, câmbio, retrovisores, faróis, dentre outras partes.

O que são classes e objetos?

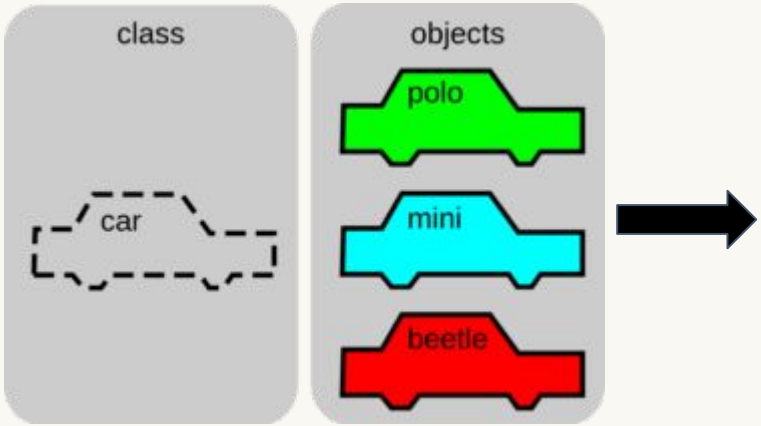
Observe que, apesar do seu carro ser único (por exemplo, possui um registro único no Departamento de Trânsito), podem existir outros com exatamente os mesmos atributos, ou parecidos, ou mesmo totalmente diferentes, mas que ainda são considerados carros.

O que são classes e objetos?

Podemos dizer então que seu objeto pode ser classificado (isto é, seu objeto pertence à uma classe) como um carro, e que seu carro nada mais é que uma instância dessa classe chamada "carro".



Representando a classe Carro



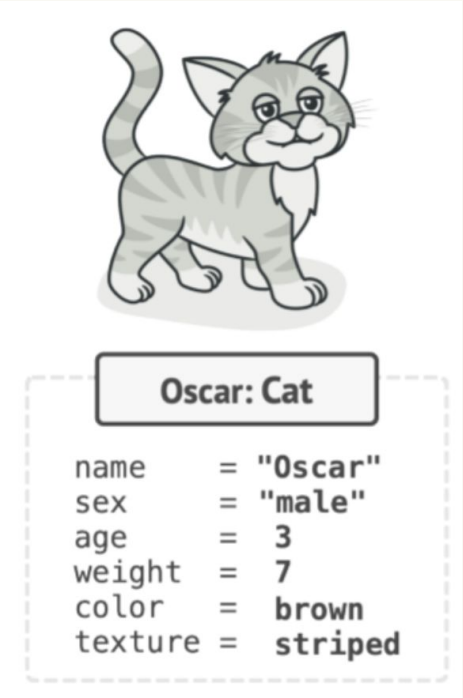

```
class Carro {
    constructor(modelo, velocidade) {
      this.modelo = modelo;
      this.velocidade = velocidade;
    }

    acelerar() {
        /* código do carro para acelerar */
    }

    frear() {
        /* código do carro para frear */
    }

    acenderFarol() {
        /* código do carro para acender o farol */
    }
}
```

Classes - Conceito

Oscar é um objeto, uma instância da classe "Gato".

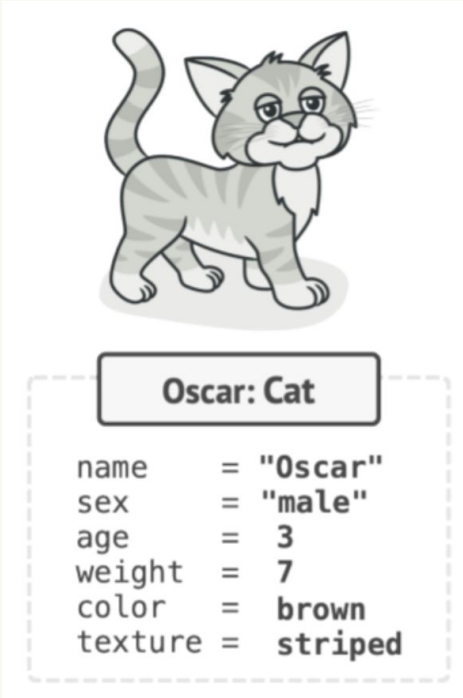
Cada gato tem vários atributos que todos os outros tem:

- Nome
- Sexo
- Idade
- Peso
- Cor
- Comida favorita
- Etc.



Classes - Conceito

Esses atributos, que todos os objetos possuem em comum, são os campos da classe.

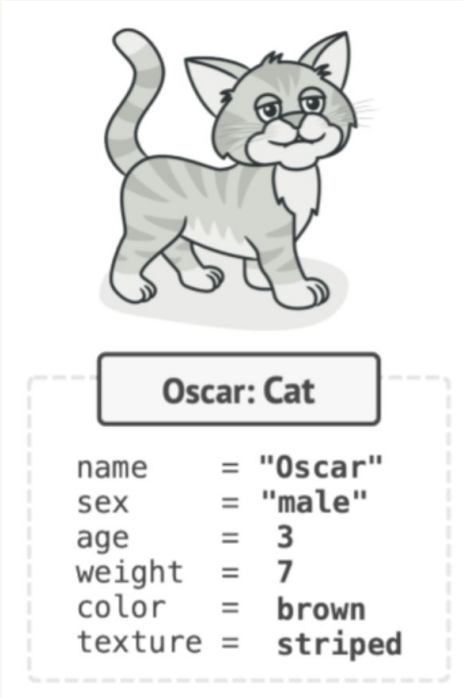


Classes - Conceito

Cada gato também se comporta de forma similar ao outro:

Respiram
Comem
Correm
Dormem
Miam

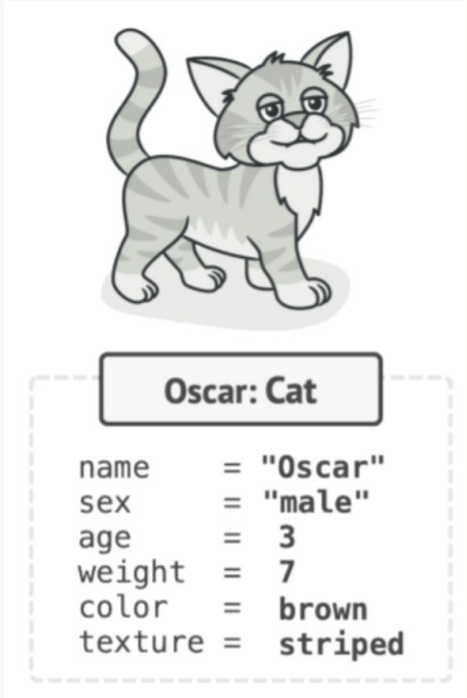
Estes são os métodos da classe.



Classes - Conceito

Coletivamente, campos e métodos podem ser referenciados como os "membros" de sua classe.

Dado armazenado dentro dos campos do objeto, frequentemente é chamado de "estado", e todos os métodos do objeto definem seu comportamento.

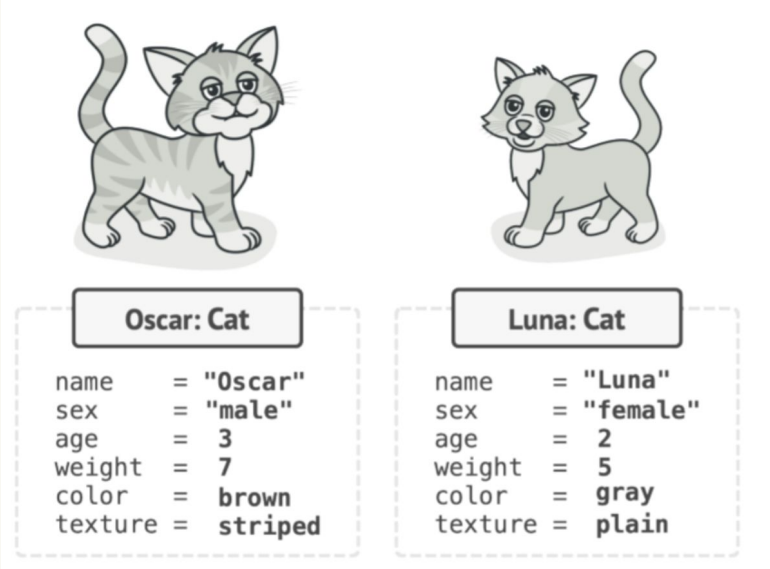


Classes - Conceito

Assim como Oscar, Luna é uma instância da clase "Gato".

Ela tem o mesmo conjunto de atributos que Oscar.

A diferença está nos valores desses atributos: O sexo de Luna é "fêmea", ela tem cor diferente e pesa menos.



Classes - Conceito

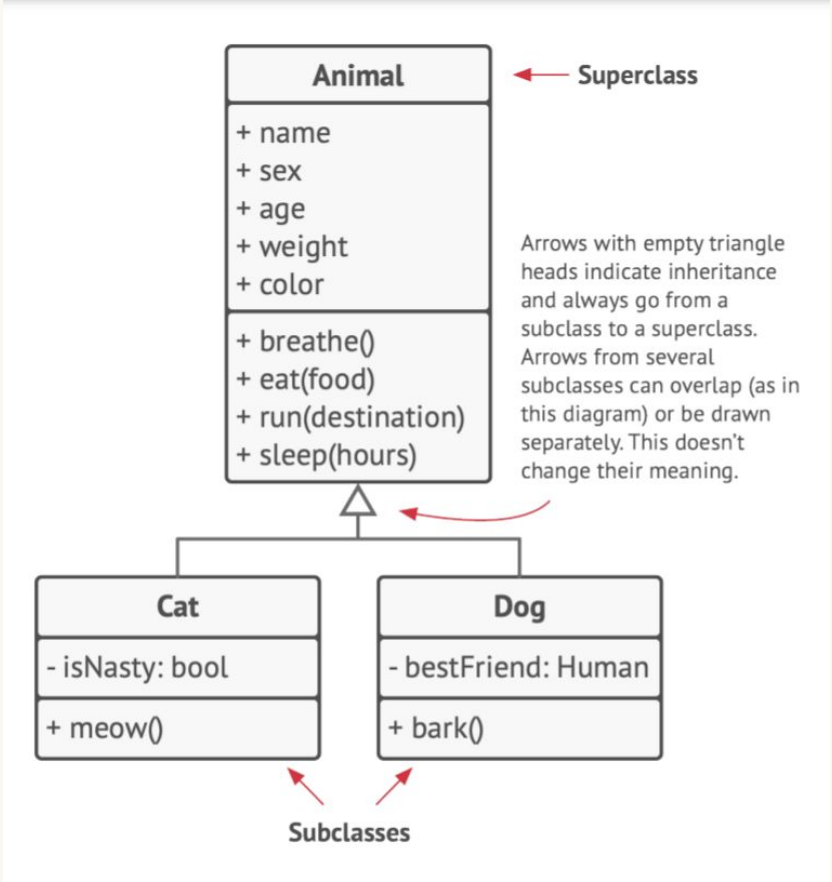
Um gato também tem muitas coisas em comum com um cachorro.

Cachorros também tem nome, sexo, idade e cor, assim como os gatos.

Cachorros também podem respirar, dormir e correr, assim como os gatos.

Poderíamos definir a classe base Animal que teria os atributos e comportamentos comuns aos dois.

Classes - Conceito



Classes - Conceito

Uma classe é um gabarito para a definição de objetos. Ou seja, ela contém as instruções e informações(se é uma string, boolean, etc) sobre os seus dados;

Uma classe contém um ou mais atributos;

Agrupar atributos em comum em um só lugar;

Classes - Conceito

Como criar uma classe no JS

```
class Cachorro {
  constructor(nome, raca){
    this.nome = nome;
    this.raca = raca;
  }

  latir(){
    console.log("Au au! Meu nome é " + this.nome + "!");
  }
}

let rex = new Cachorro("Manfredo", "Beagle");
```

Classes - Conceito

Exemplo de Classe

```
1  class Quadrado {
2    constructor(lado) {
3      this.lado = lado
4    }
5  }
6
7  const quadrado = new Quadrado(3)
8
9  console.log(quadrado);
```

CONSOLE

```
Quadrado {
  lado: 3
}
```

Classes - Conceito

É uma função que cria uma instância da classe. Ou seja, o construtor é chamado quando passamos o parâmetro new. O objetivo do constructor é criar um objeto e setar valores se houver.

Construtor

```
1  class Quadrado {
2    constructor(lado) {
3      this.lado = lado
4    }
5  }
6
7  const quadrado = new Quadrado(3)
8
9  console.log(quadrado);
```

```
CONSOLE ×

Quadrado {
  lado: 3
}
```

Getters e Setters

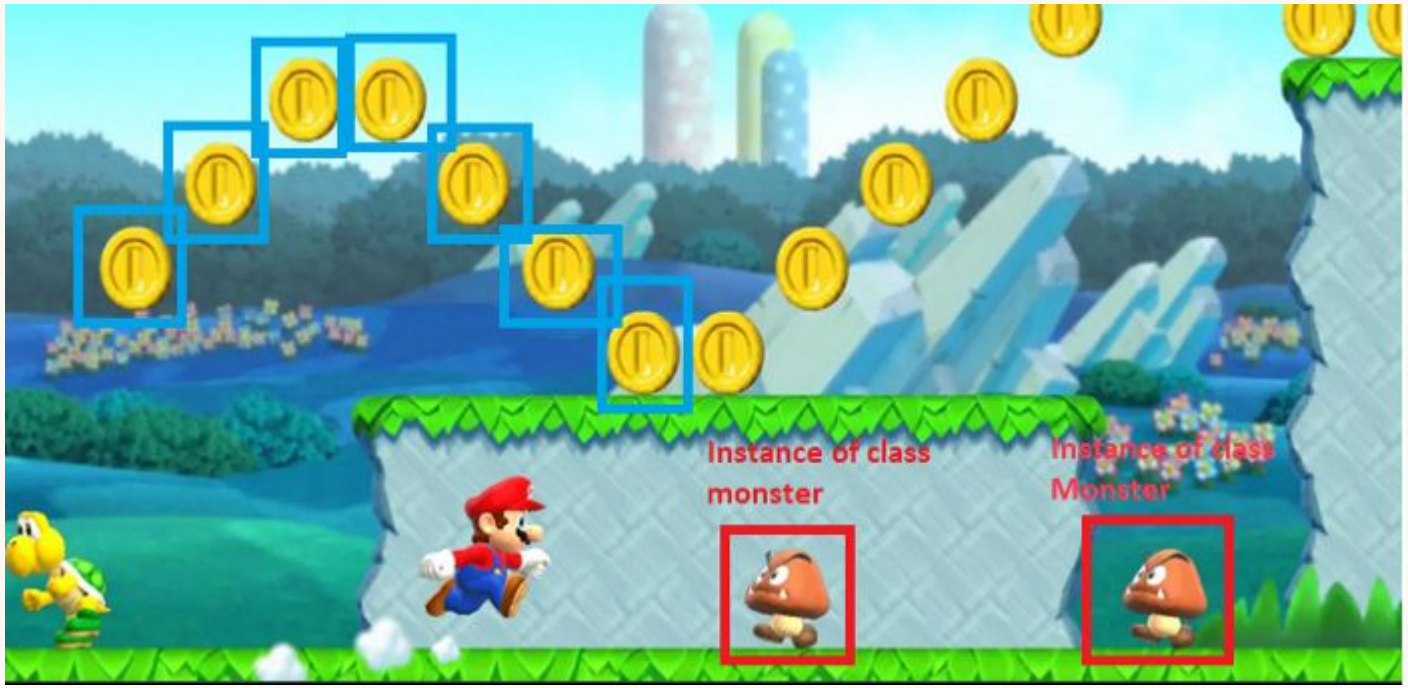
Getters e setters permitem que você obtenha e defina propriedades de objetos por meio de métodos.

Este exemplo usa uma propriedade lang para obter o valor da propriedade language:

Get e Set

```
const person = {
    firstName: "John",
    lastName: "Doe",
    language: "en",
    get lang() {
        return this.language;
    }
};

console.log(person.lang);
```

Utilizando o jogo do Mario para entendimento



Utilizando o exemplo do Mario, vamos criar uma classe de Monstro que tenha os seguintes atributos:

- vida e velocidade(os dois são números).
- Depois, crie um método chamado ataque, que vai mostrar uma mensagem dizendo que o monstro irá atacar.

```
class Monster {

  constructor(health, speed) {
      this.health = health
      this.speed = speed;
    }

  attack() {
    return 'O nosso monstro vai atacar!'
  }
  }

  const monster1 = new Monster(30, 5);
  const monsterAttack = new Monster(30,5).attack();

  console.log(monster1);
  console.log(monsterAttack);      You, seconds ago
```

```
16  console.log(monster1);
17  console.log(monsterAttack)
```

```
CONSOLE ×

Monster {
  health: 30,
  speed: 5
}

O nosso monstro vai atacar!
```

Crie uma classe Endereço. Endereço possui os seguintes atributos:

- Logradouro
- Número
- Cidade
- Estado
- País
- CEP

Retorno: `Logradouro: ${this.logradouro} \n Numero: ${this.numero}`

Crie uma classe Endereço. Endereço possui os seguintes atributos: Logradouro, Número, Cidade, Estado, País, CEP.

```
class Endereco{
    constructor(logradouro, numero, cidade, estado, pais, cep){
        this.logradouro = logradouro;
        this.numero = numero;
        this.cidade = cidade;
        this.estado = estado;
        this.pais = pais;
        this.cep = cep;
    }
}
```

Exercícios

```
class Endereco {
    constructor(logradouro, numero, cidade, estado, pais, cep) {
        this.logradouro = logradouro;
        this.numero = numero;
        this.cidade = cidade;
        this.estado = estado;
        this.pais = pais;
        this.cep = cep;

        console.log(`Logradouro: ${this.logradouro} \n Numero: ${this.numero}`);

    }
}

const endereco = new Endereco ("Ernesto Alves", 34)
```

Exercícios

Crie uma classe Veículo. Veículo possui os seguintes atributos:

- tipoVeiculo,
- marca,
- modelo,
- ano,
- placa,
- numMultas,
- velocidadeMaxima.

Construa os seguintes métodos: get tipoModelo, que retorna o tipo e o modelo do veículo; adicionaMulta(codigoDaPlaca) que, caso o código da placa passado por parâmetro for igual ao do objeto Veículo, adiciona 1 ao numMultas e retorna o mesmo.

Exercícios

```
class Veiculo{
    constructor(tipo, marca, modelo, ano, placa, numMultas, velocidadeMaxima){
        this.tipo = tipo;
        this.marca = marca;
        this.modelo = modelo;
        this.ano = ano;
        this.placa = placa;
        this.numMultas = numMultas;
        this.velocidadeMaxima = velocidadeMaxima;
    }

    get tipoModelo(){
        return (this.tipo, this.modelo)
    }

    adicionaMulta(codigoDaPlaca){
        if(codigoDaPlaca === this.placa)
        {
            this.numMultas = this.numMultas + 1;
            return(this.numMultas);
        }
    }
}
```

DEVinHouse

Parcerias para desenvolver a sua carreira

OBRIGADO!

SENAI

<LAB365>