Classes e Objetos

Apresentação dos conceitos na Programação Orientada à Objetos

CONTEÚDOS

01

O que são?

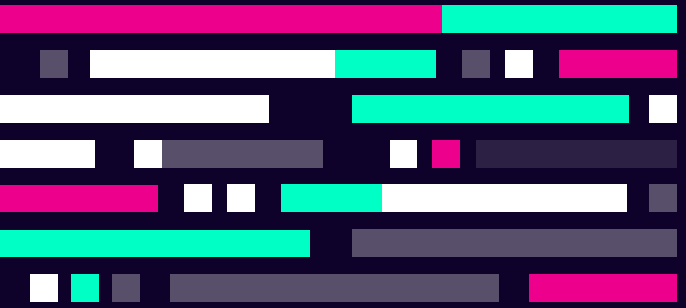
CLASSE

É um **conjunto de objetos** que compartilham de características **semelhantes**. Um **molde abstrato** que descrevem propriedades e comportamentos de entidades do mundo real.

Class Carro

Veículo com **4 rodas**, possui um **motor**, utiliza **combustível**, acomoda **motorista e passageiros*** e possibilita a **locomoção** dos mesmos para **longas distâncias.**

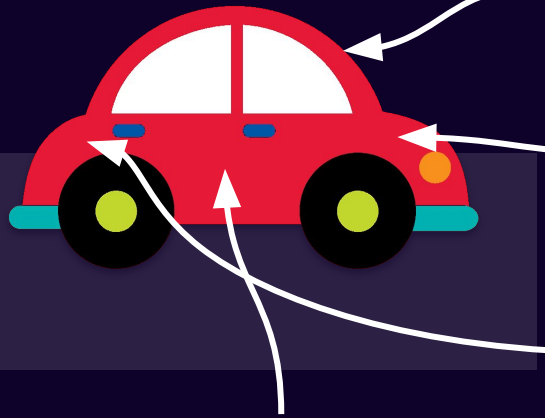
Atributos	Funcionalidades
Potência do motor	Ligar/Desligar carro
Cor do carro	Acelerar/Frear
Tipo de combustível	Abastecer
Número de portas	Ligar rádio*



OBJETO

É um tipo de dado, variável, função ou estrutura de dados que possui **características e funcionalidades**, é a referência de uma **entidade real**, onde é possível alterar seus **atributos** a partir dos **métodos** da classe em que o objeto pertence.

Object carro_vermelho



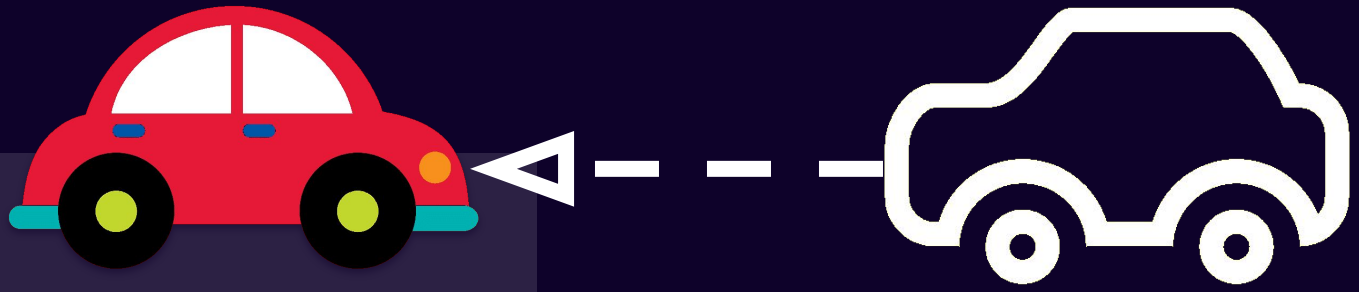
Cor: Vermelho

Motor: 1.0

Combustível: Gasolina

Nº de portas: 4

Object carro_vermelho

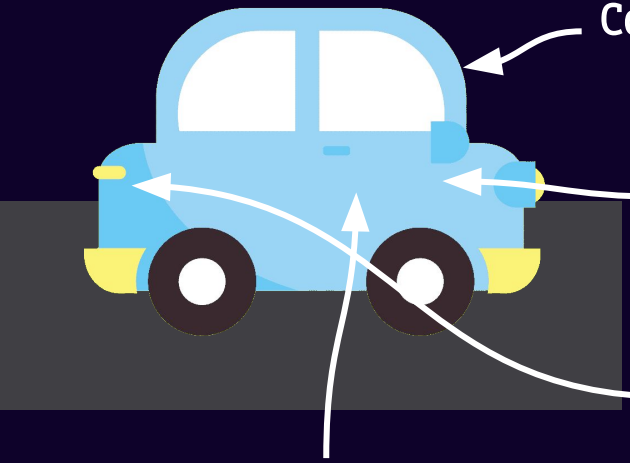


Carro real

Molde do carro

Relação entre Classes e Objetos

carro_azul



Cor: Azul

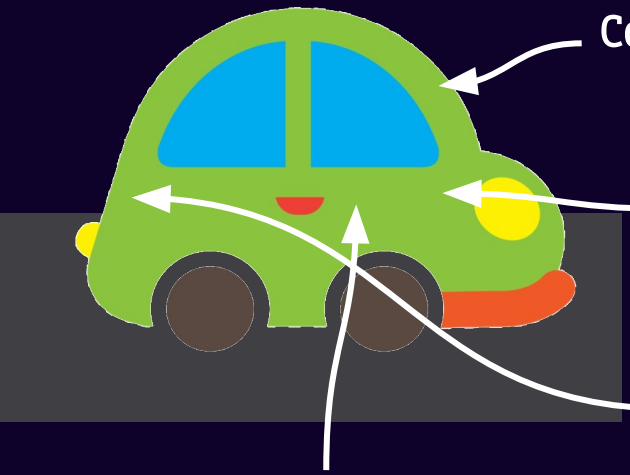
Motor: 1.6

Combustível: Flex

Nº de portas: 2

Relação entre Classes e Objetos

carro_verde

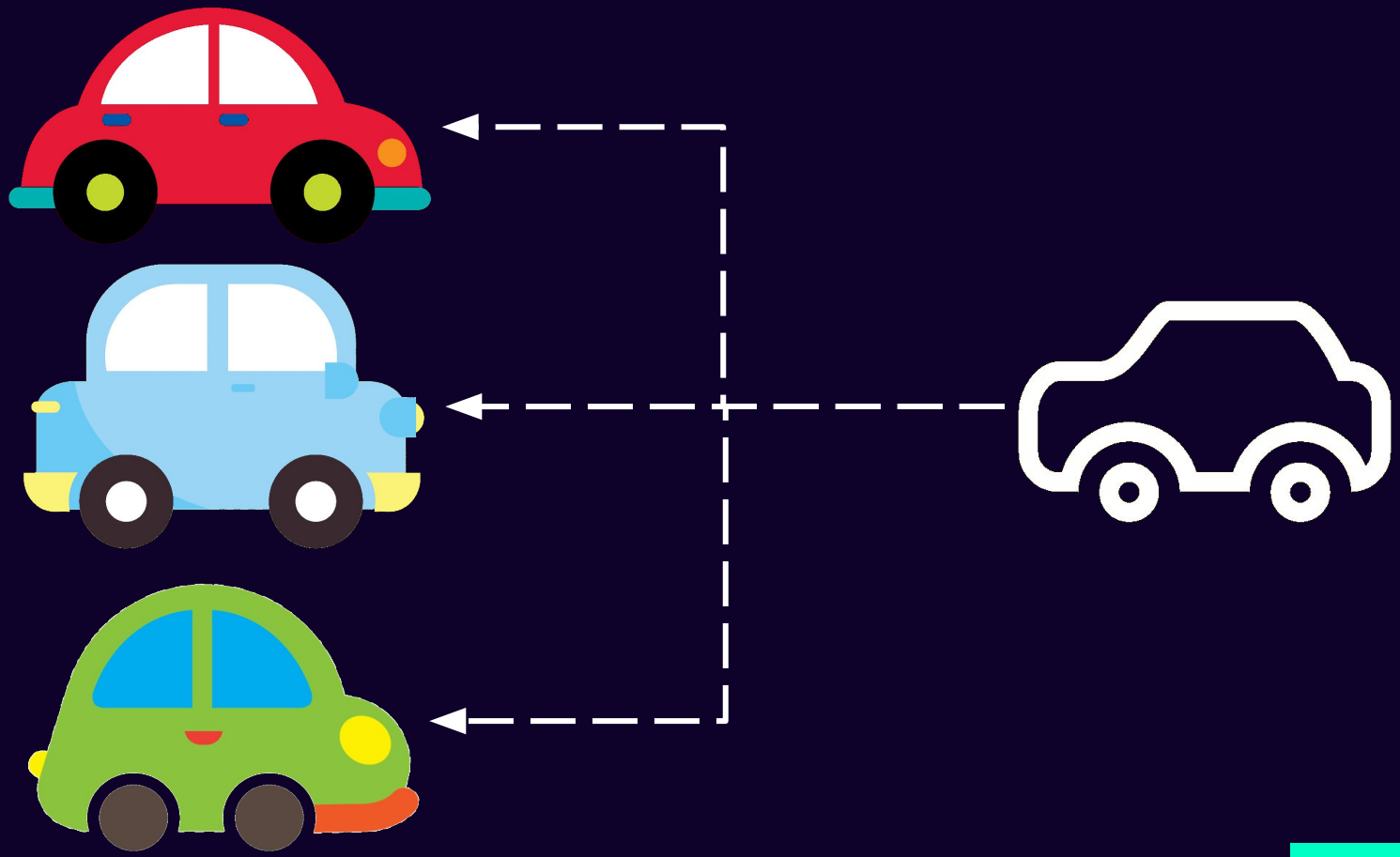


Cor: Verde

Motor: 2.0

Combustível: Gasolina

Nº de portas: 2



Possui motor, rodas, faróis, utiliza combustível...

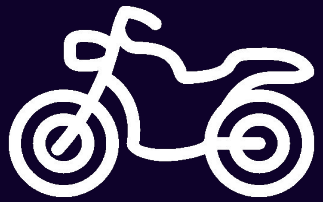
?

Tamanho, carroceria, transporte de produtos...

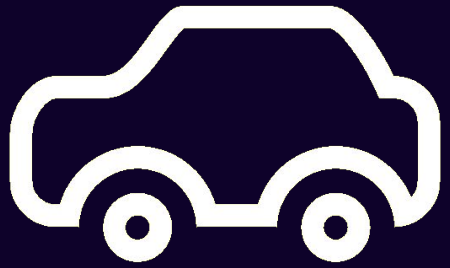
Classe Veiculo

Classe **mãe**

Classe **Caminhao**



Classe **Moto**



Classe **Carro**

02

Instância de uma classe

"[...] objects are instantiated from a class, and each object is referred to as an instance of the class".

– Diane Zak

Definição

A definição de instância de classe e objeto é um tanto quanto confusa, há uma linha tênue entre as duas definições. De forma mais clara, instanciar a classe é equivalente à ação de atribuir as propriedades ao objeto, e objeto é a instanciação da classe, o produto físico que veio do molde.

Classe Carro

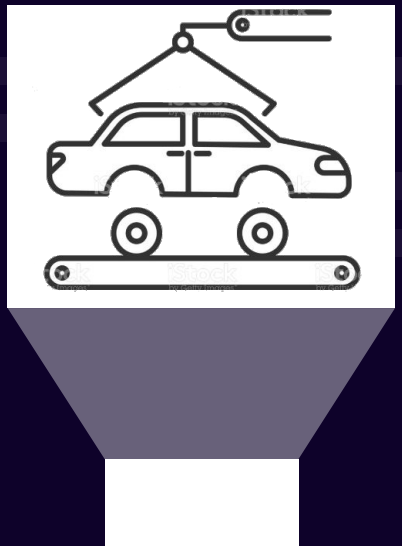
O molde do carro, abstração do objeto

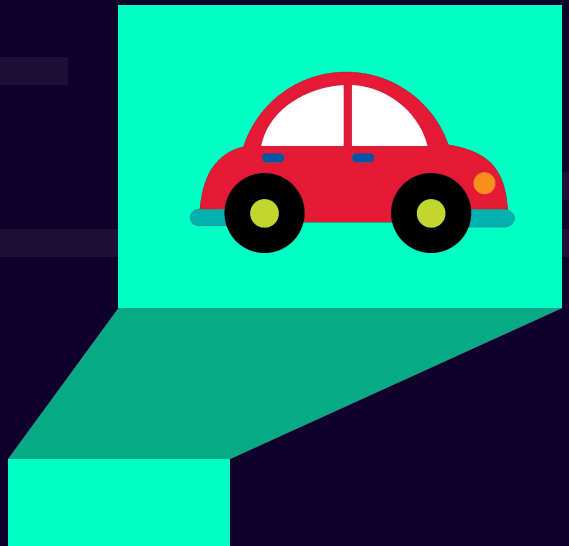
Instância da Classe

É a atribuição das propriedades para o objeto, a fábrica do carro

Objeto carro

É o produto final da instanciação, com atributos e identificação única





“Mas e se os atributos forem iguais?”

É possível ter dois objetos iguais, mas são objetos diferentes alocados na memória

BRA2E21

Placa de identificação de um carro.

0x0018F36C

Endereço de memória de um objeto.

03

Declaração de Classe

INTRODUÇÃO

```
class MyClass {
    ...
}
```

INTRODUÇÃO

```
class Carro {
    string Marca;
    string Modelo;
    int Ano;
}
```

INTRODUÇÃO

```
class Carro {
    ...
    public void GetInfo() {
        Console.WriteLine("Informações
do carro...");
    }
}
```

Declaração da classe Carro

```
class Carro {
    string Marca;     // Atributos do carro
    string Modelo;
    int Ano;

    // O método construtor tem o mesmo nome da classe mas sem o tipo de retorno
    public Carro(string marca, string modelo, int ano) {
        this.Marca = marca;
        this.Modelo = modelo;
        this.Ano = ano;
    }

    // Método que exibe as informações do carro
    public void GetInfo() {
        Console.WriteLine($"Marca: {this.Marca}\nModelo: {this.Modelo}\nAno: {this.Ano}");
    }
}
```

O uso do prefixo *this* é uma referência à variável interna da classe, se faz necessário caso o nome dos argumentos e dos atributos sejam iguais, assim o prefixo faz referência ao atributo da classe e não ao argumento.

OBS.: O uso não era necessário no exemplo

Criação do objeto carro

```
class Program {
    static void Main() {
        Carro carro = new Carro(marca: "Fiat", modelo: "Uno", ano: 2010);
        carro.GetInfo();
    }
}
```

```
> dotnet run
Marca: Fiat
Modelo: Uno
Ano: 2010
```

O uso do nome dos parâmetros do construtor não é necessário, foi utilizado apenas para fim didático.

04

Propriedades e Métodos

Carro:
Ligar/Desligar
Obter informações
Acelerar/Freiar
Ligar rádio*
...

Métodos

São funções que utilizam ou alteram as propriedades para um fim específico

Propriedades

Características de um objeto, descrevem aspectos e estados do objeto

Carro:
Marca
Modelo
Ano
Cor
Potência do motor
...

classe Carro

```
class Carro {
    private string Marca;    // Atributos do carro
    private string Modelo;
    private int Ano;

    // O método construtor tem o mesmo nome da classe mas sem o tipo de retorno
    public Carro(string marca, string modelo, int ano) {
        this.Marca = marca;
        this.Modelo = modelo;
        this.Ano = ano;
    }

    // Método que exibe as informações do carro
    public void GetInfo() {
        Console.WriteLine($"Marca: {this.Marca}\nModelo: {this.Modelo}\nAno: {this.Ano}");
    }
}
```

Programa Principal

```
class Program {
   static void Main() {
       Carro carro = new Carro("Fiat", "Uno", 2010);
       carro.Ano = 2020; ERRO: atributo inacessível
   }
}
```

Ao tornar um atributo privado, não é possível obter o valor do mesmo fora da classe, para isto, existem as propriedades

PROPRIEDADE

```
class Carro {
    public string Marca { get; }
    public string Modelo { get; }
    public int Ano { get; }
    public Cor Cor { get; set; }
    ...
}

public enum Cor {
    Vermelho, Azul, Verde, Branco,
Preto
}
```

DEFINIÇÃO

No exemplo, as palavras chave ***get*** e ***set*** definem o tipo de acesso da propriedade. ***get*** é o modo de *leitura*, e ***set*** o modo de *gravação*, a vantagem de dividir os dois modos, é a possibilidade de proteger atributos sigilosos e que não podem ser mutáveis diretamente.

Acessando propriedades do objeto

```
class Program {
    static void Main() {
        Carro carro = new Carro(marca: "Fiat", modelo: "Uno", ano: 2010, cor: Cor.Verde);
        Console.WriteLine(carro.Marca); // Propriedade somente leitura
        Console.WriteLine(carro.Cor);   // Leitura do valor "get"

        carro.Cor = Cor.Azul;           // Alteração do valor "set"
        Console.WriteLine(carro.Cor);   // Leitura do valor alterado "get"
    }
}
```

```
> dotnet run
Fiat
Verde
Azul
```

No nosso exemplo, atributos que não podem ser alterados são: *Marca*, *Modelo* e *Ano*, já a *Cor* pode ser alterada.

Exemplo utilizando propriedades e métodos

```
class Carro {
    ...
    private double _velocidade;
    public double Velocidade {
        get { return _velocidade; }
        private set { _velocidade = value; }
    }

    public Carro(...) {
        ...
        Velocidade = 0; // O carro inicia parado
    }

    // Acelera o carro
    public void Accelerate(double value) {
        if (value > 0)
            Velocidade += value;
    }

    // Freia o carro
    public void Break(double value) {
        if (value > 0) {
            if (value ⩽ Velocidade)
                Velocidade -= value;
            else Velocidade = 0;
        }
    }

    // Mostra a velocidade
    public void GetSpeed() {
        if (Velocidade == 0)
            Console.WriteLine("O carro está parado!");
        else
            Console.WriteLine($"A velocidade do carro é:
{Velocidade} Km/h");
    }
}
```

Exemplo utilizando propriedades e métodos

```
class Program {
    static void Main() {
        var carro = new Carro(marca: "Fiat", modelo: "Uno", ano: 2010, cor: Cor.Verde);
        carro.Accelerate(20);    // Acelera até 20Km/h
        carro.GetSpeed();

        carro.Break(30);         // Freia 30Km/h, ou seja, o carro para
        carro.GetSpeed();

        carro.Accelerate(10);    // Acelera novamente
        carro.GetSpeed();
    }
}
```

```
> dotnet run
A velocidade do carro é: 20 Km/h
O carro está parado!
A velocidade do carro é: 10 Km/h
```

Os métodos *Accelerate* e *Break* permitem alterar a propriedade *Velocidade* de forma mais segura.

Propriedades NÃO são variáveis, não é possível utilizar elas como argumentos *ref* e *out* em funções.

Propriedades e métodos nos conceitos da linguagem

```
var variavel = new int();   // Declara variável da classe int.Int32()
Console.WriteLine(variavel.GetType());  // Mostra o tipo do objeto com o método GetType

variavel = int.Parse(Console.ReadLine());  /* Utiliza método da classe int para converter
                                              string em inteiro */
string palavra = "Objeto";  // Cria objeto da classe String
Console.WriteLine(palavra.Length);  // Obtém o tamanho da string com a propriedade Length
```

A linguagem C# trata tudo como objeto, e é por isso que qualquer variável ou estrutura de dados terão suas propriedades e métodos internos

REFERÊNCIAS

Curso Udemy: Introdução ao C# - Módulo 1

Curso Udemy: Introdução ao C# - Módulo 2

Alura

DCA/Unicamp

Wikipedia - Classe

Wikipedia - Objeto

Facom/UFU

Wikibooks

Stack Overflow

Wikipedia - Instancia de classe

Docs Microsoft - Propriedades e Métodos

Macoratti.net - Propriedades

THANKS !

github.com/marlonangeli/pesquisa_poo-classes_e_objetos