

# Agenda

- ❑ Sobrecarga de Métodos
- ❑ Utilização da palavra reservada this
- ❑ Métodos Construtores
  - ❑ – Representação Gráfica do Construtor

## Sobrecarga de Métodos – (Overloading)

Um recurso usual em programação OO é o uso de sobrecarga de métodos.

Sobrecarregar um método significa prover mais de uma versão de um mesmo método.

As versões devem, necessariamente, conter parâmetros diferentes, seja no tipo ou no número desses parâmetros (o tipo do retorno é indiferente).

Ninja
~ mover(x : double) : void ~ mover(x : double, y : double) : double ~ mover(x : double, y : double, velocidade : int) : void ~ mover(destino : String) : int ~ mover(inimigoMaisProximo : Inimigo) : Inimigo

# Sobrecarga de Métodos – (Overloading)

0 referências

```
public class Ninja
{
    0 referências
    public void Mover(double x)
    {
        // ...
    }
    0 referências
    public double Mover(double x, double y)
    {
        // ...
    }
    0 referências
    public void Mover(double x, double y, int velocidade)
    {
        // ...
    }
    0 referências
    public int Mover(string destino)
    {
        // ...
    }
    0 referências
    public Inimigo Mover(Inimigo inimigoMaisProximo)
    {
        // ...
    }
}
```

## Utilização da palavra reservada **this**

Dentro de um método, o objeto pode precisar de sua própria referência.

Em C#, a palavra reservada **this** significa essa referência ao próprio objeto.

Através da palavra reservada **this** é possível acessar atributos, métodos e construtores do objeto da classe em questão.

## Utilização da palavra reservada this

0 referências

```
public class Carro
```

```
{
```

1 referência

```
public string Modelo { get; set; }
```

0 referências

```
public float Motor { get; set; }
```

0 referências

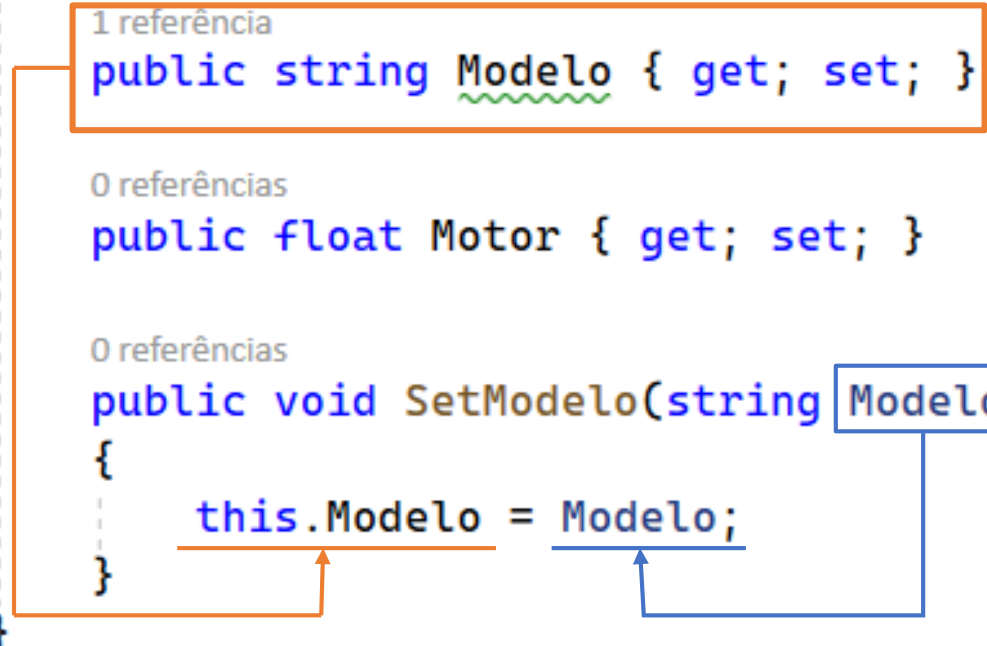
```
public void SetModelo(string Modelo)
```

```
{
```

```
    this.Modelo = Modelo;
```

```
}
```

```
}
```



# Construtores

Desempenham papel essencial no processo de instanciação de uma classe.

Os construtores também são utilizados para inicializar os atributos com valores padrão ou com valores informados

São métodos especiais que são invocados juntamente com o operador **new**.

Construtores não possuem valor de retorno (nem mesmo void ) e possuem o mesmo nome da classe.

Toda classe tem pelo menos um construtor sempre definido.

Quando não especificamos nenhum construtor, a linguagem C# fornece um construtor default (padrão) - vazio - que não recebe parâmetros.

Se declararmos algum construtor, esse construtor padrão não será mais fornecido pelo C#.

# Construtores

1 referência

```
public class Carro
```

```
{
```

0 referências

```
public string Modelo { get; set; }
```

0 referências

```
public float Motor { get; set; }
```

0 referências

```
public Carro()
```

```
{
```

```
    Console.WriteLine("criando instancia de um objeto...");
```

```
}
```

```
}
```



Construtor

Utilização do construtor declarado acima:

```
Carro carro = new Carro();
```

# Construtores

3 referências

```
public class Carro
{
    1 referência
    public string Modelo { get; set; }

    0 referências
    public float Motor { get; set; }

    1 referência
    public Carro(string modeloDoCarro)
    {
        Modelo = modeloDoCarro;
    }
}
```



Construtor

Utilização do construtor declarado acima:

```
Carro carro = new Carro("Gol");
```



# Construtores

3 referências

```
public class Carro
```

```
{  
    1 referência  
    public string Modelo { get; set; }  
  
    1 referência  
    public float Motor { get; set; }  
  
    1 referência  
    public Carro(string modeloDoCarro, float motor)  
    {  
        Modelo = modeloDoCarro;  
        Motor = motor;  
    }  
}
```



Construtor

Utilização do construtor declarado acima:

```
Carro carro = new Carro("Gol", 1.8f);
```

# Construtores

```
1 referência
public class Pessoa
{
    1 referência
    public string Nome { get; set; }


    1 referência
    public int Idade { get; set; }

    1 referência
    public Carro Carro { get; set; }

    0 referências
    public Pessoa(string nome, int idade, Carro carro)
    {
        Nome = nome;
        Idade = idade;
        Carro = carro;
    }
}
```

Exemplos de utilização do construtor declarado ao lado:

```
Carro carro = new Carro("Fusca", 1.2f);
Pessoa pessoa = new Pessoa("Pedro", 9, carro);
```



Passando **null** como parâmetro:

```
Pessoa pessoa2 = new Pessoa("Maria", 9, null);
```



## Representação gráfica de um Construtor

Assim como atributos e métodos, também é possível representar construtores no diagrama de classes.

