



SW-I

Sistemas Web I

Prof. Anderson Vanin

MÉTODOS CONSTRUTORES

Um método construtor, como o próprio nome já diz, é responsável pela criação do objeto daquela classe, iniciando com valores seus atributos ou realizando outras funções que possam vir a ser necessárias.

A inicialização de objetos por meio de construtores é necessária para evitar erro de referência nula quando usando objetos que foram declarados mas não inicializados.



Exemplo – Método Construtor

Vamos criar uma classe chamada Pessoa e instanciar alguns registros.



Pessoa.cs > {} projeto_pessoa > projeto_pessoa.Pessoa

```
1 namespace projeto_pessoa;
2 references
3 {
4     2 references
5     private string? Nome {get;set;}
6     2 references
7     private string? Sobrenome {get;set;}
8     2 references
9     private int Idade {get;set;}
10
11     1 reference
12     public void insereNome(string nome){
13         this.Nome = nome;
14     }
15     1 reference
16     public void insereSobreNome(string sobrenome){
17         this.Sobrenome = sobrenome;
18     }
19     1 reference
20     public void insereIdade(int idade){
21         this.Idade = idade;
22     }
23
24     1 reference
25     public void mostraDados(){
26         Console.WriteLine("Dados Cadastrados:");
27         Console.WriteLine("O nome é: " + this.Nome);
28         Console.WriteLine("O sobrenome é: " + this.Sobrenome);
29         Console.WriteLine("A idade é: " + this.Idade);
30     }
31 }
```

Program.cs > {} projeto_pessoa > projeto_pessoa.Program > Main(string[] args)

```
1 namespace projeto_pessoa;
2 0 references
3 class Program
4 {
5     0 references
6     static void Main(string[] args)
7     {
8         Pessoa p1 = new Pessoa();
9         p1.insereNome("Anderson");
10        p1.insereSobreNome("Vanin");
11        p1.insereIdade(48);
12        p1.mostraDados();
13    }
14 }
```

```
C:\Users\Anderson\Desktop\projeto_pessoa>dotnet run
Dados Cadastrados:
O nome é: Anderson
O sobrenome é: Vanin
A idade é: 48
```

Exemplo – Método Construtor

Agora vamos supor, que caso eu não insira nenhum valor para os atributos, os dados a serem mostrados como padrão serão:

Dados Cadastrados:

O nome é: Fulano

O sobrenome é: Silva

A idade é: 18

Para que isso seja possível, vamos agora criar um método construtor da classe Pessoa.

```

Pessoa.cs > {} projeto_pessoa > projeto_pessoa.Pessoa > Pessoa()
1 namespace projeto_pessoa;
  2 references
2 class Pessoa
3 {
  1 reference
4     public Pessoa(){
5         this.Nome = "Fulano";
6         this.Sobrenome = "Silva";
7         this.Idade = 18;
8     }
  3 references
9     private string? Nome {get;set;}
  3 references
10    private string? Sobrenome {get;set;}
  3 references
11    private int Idade {get;set;}
12
  1 reference
13    public void insereNome(string nome){
14        this.Nome = nome;
15    }
  1 reference
16    public void insereSobreNome(string sobrenome){
17        this.Sobrenome = sobrenome;
18    }
  1 reference
19    public void insereIdade(int idade){
20        this.Idade = idade;
21    }
22
  1 reference
23    public void mostraDados(){
24        Console.WriteLine("Dados Cadastrados:");
25        Console.WriteLine("O nome é: " + this.Nome);
26        Console.WriteLine("O sobrenome é: " + this.Sobrenome);
27        Console.WriteLine("A idade é: " + this.Idade);
28    }
29 }

```

```

Program.cs > {} projeto_pessoa > projeto_pessoa.Program
• 1 namespace projeto_pessoa;
  0 references
2 class Program
3 {
  0 references
4     static void Main(string[] args)
5     {
6         Pessoa p1 = new Pessoa();
7         //p1.insereNome("Anderson");
8         //p1.insereSobreNome("Vanin");
9         //p1.insereIdade(48);
10        p1.mostraDados();
11    }
12 }

```

```

C:\Users\Anderson\Desktop\projeto_pessoa>dotnet run
Dados Cadastrados:
O nome é: Fulano
O sobrenome é: Silva
A idade é: 18

```

Exemplo – Método Construtor

Agora uma outra situação. Vamos supor que agora seja obrigatório informar o nome e a idade, mas o sobrenome seja opcional.



Pessoa.cs > {} projeto_pessoa > projeto_pessoa.Pessoa > Sobrenome

```
1 namespace projeto_pessoa;
  2 references
2 class Pessoa
3 {
  0 references
4     public Pessoa(){
5         this.Nome = "Fulano";
6         this.Sobrenome = "Silva";
7         this.Idade = 18;
8     }
9
10     1 reference
11     public Pessoa(string nome, int idade){
12         this.Nome = nome;
13         this.Idade = idade;
14         this.Sobrenome = "NÃO INFORMADO";
15     }
16     4 references
17     private string? Nome {get;set;}
18     4 references
19     private string? Sobrenome {get;set;}
20     4 references
21     private int Idade {get;set;}
22
23     0 references
24     public void insereNome(string nome){ ...
25     0 references
26     public void insereSobreNome(string sobrenome){ ...
27     0 references
28     public void insereIdade(int idade){ ...
29     1 reference
30     public void mostraDados(){ ...
31
32 }
33
34
35 }
```

Program.cs > {} projeto_pessoa > projeto_pessoa.Program > Main(string[] args)

```
1 namespace projeto_pessoa;
  0 references
2 class Program
3 {
  0 references
4     static void Main(string[] args)
5     {
6         //Pessoa p1 = new Pessoa();
7         Pessoa p1 = new Pessoa("Anderson",48);
8         //p1.insereNome("Anderson");
9         //p1.insereSobreNome("Vanin");
10        //p1.insereIdade(48);
11        p1.mostraDados();
12    }
13 }
```

```
C:\Users\Anderson\Desktop\projeto_pessoa>dotnet run
Dados Cadastrados:
O nome é: Anderson
O sobrenome é: NÃO INFORMADO
A idade é: 48
```


Exemplo

Vamos supor que você queira aplicar um desconto ao valor de um produto somente se o valor deste desconto for informado.

Teremos três situações:

1. Se não passarmos nenhuma informação, o nome do produto será “TicToc” e seu preço será R\$ 15.
 2. Se passarmos somente o nome do produto, este produto terá um preço padrão de R\$ 10.
 3. Se passarmos o nome e o valor do desconto (em %), esse desconto será aplicado ao valor de R\$ 10.
-

Produto.cs > {} projeto_desconto > projeto_desconto.Produto > mostraDados()

```
1 namespace projeto_desconto;
  2 references
2 class Produto
3 {
  0 references
4 public Produto(){
5     this.Nome = "TICTOC";
6     this.Preco = 15;
7 }
  0 references
8 public Produto(string nome){
9     this.Nome = nome;
10    this.Preco = 10;
11 }
  1 reference
12 public Produto(string nome, double desconto){
13     this.Nome = nome;
14     this.Preco = 10*(1-(desconto/100));
15 }
  5 references
16 private string? Nome {get;set;}
  5 references
17 private double Preco {get;set;}
18
  0 references
19 public void CadastraNome(string nome){
20     this.Nome = nome;
21 }
  0 references
22 public void CadastraPreco(double preco){
23     this.Preco = preco;
24 }
25
  1 reference
26 public void mostraDados(){
27     Console.WriteLine("Produto: " + this.Nome);
28     Console.WriteLine("Preço: " + this.Preco);
29 }
30 }
```

Program.cs > {} projeto_desconto > projeto_desconto.Program

```
1 namespace projeto_desconto;
  0 references
2 class Program
3 {
  0 references
4     static void Main(string[] args)
5     {
6         // SITUAÇÃO 1
7         //Produto p1 = new Produto();
8         // SITUAÇÃO 2
9         //Produto p1 = new Produto("Caneta");
10        // SITUAÇÃO 3
11        Produto p1 = new Produto("Caneta",50);
12        p1.mostraDados();
13    }
14 }
```

Exercício

Ao modelar um sistema de controle de aviões em um aeroporto, todos os aviões possuem, **obrigatoriamente, um código e uma empresa**, além disso, **opcionalmente, uma cidade de entrada e saída**.

Implemente uma solução em C# orientada a Objetos para atender ao enunciado acima.

