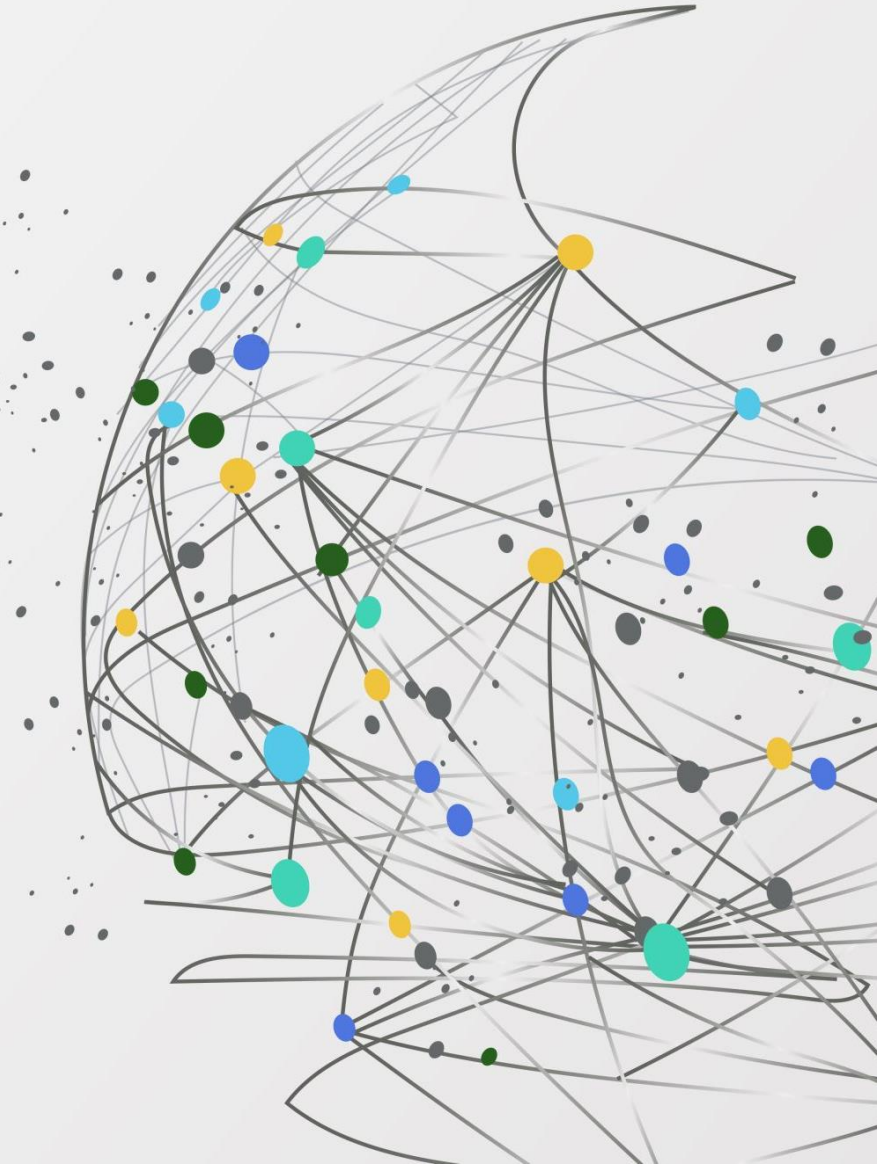

SW-I
Sistemas Web I

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

```
namespace projeto_pessoa;
2 references
class Pessoa
{
    2 references
    private string? Nome {get;set;}
    2 references
    private string? Sobrenome {get;set;}
    2 references
    private int Idade {get;set;}

    1 reference
    public void insereNome(string nome){
        this.Nome = nome;
    }
    1 reference
    public void insereSobreNome(string sobrenome){
        this.Sobrenome = sobrenome;
    }
    1 reference
    public void insereIdade(int idade){
        this.Idade = idade;
    }

    1 reference
    public void mostraDados(){
        Console.WriteLine("Dados Cadastrados:");
        Console.WriteLine("O nome é: " + this.Nome);
        Console.WriteLine("O sobrenome é: " + this.Sobrenome);
        Console.WriteLine("A idade é: " + this.Idade);
    }
}
```

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

```
namespace projeto_pessoa;
0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        Pessoa p1 = new Pessoa();
        p1.insereNome("Anderson");
        p1.insereSobreNome("Vanin");
        p1.insereIdade(48);
        p1.mostraDados();
    }
}

```

```
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()

```
namespace projeto_pessoa;
2 references
class Pessoa
{
    1 reference
    public Pessoa(){
        this.Nome = "Fulano";
        this.Sobrenome = "Silva";
        this.Idade = 18;
    }
    3 references
    private string? Nome {get;set;}
    3 references
    private string? Sobrenome {get;set;}
    3 references
    private int Idade {get;set;}

    1 reference
    public void insereNome(string nome){
        this.Nome = nome;
    }
    1 reference
    public void insereSobreNome(string sobrenome){
        this.Sobrenome = sobrenome;
    }
    1 reference
    public void insereIdade(int idade){
        this.Idade = idade;
    }

    1 reference
    public void mostraDados(){
        Console.WriteLine("Dados Cadastrados:");
        Console.WriteLine("O nome é: " + this.Nome);
        Console.WriteLine("O sobrenome é: " + this.Sobrenome);
        Console.WriteLine("A idade é: " + this.Idade);
    }
}
```

Program.cs > {} projeto_pessoa > projeto_pessoa.Program

```
namespace projeto_pessoa;
0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        Pessoa p1 = new Pessoa();
        //p1.insereNome("Anderson");
        //p1.insereSobreNome("Vanin");
        //p1.insereIdade(48);
        p1.mostraDados();
    }
}
```

```
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

```
namespace projeto_pessoa;
2 references
class Pessoa
{
    0 references
    public Pessoa(){
        this.Nome = "Fulano";
        this.Sobrenome = "Silva";
        this.Idade = 18;
    }

    1 reference
    public Pessoa(string nome, int idade){
        this.Nome = nome;
        this.Idade = idade;
        this.Sobrenome = "NÃO INFORMADO";
    }
    4 references
    private string? Nome {get;set;}
    4 references
    private string? Sobrenome {get;set;}
    4 references
    private int Idade {get;set;}

    0 references
    public void insereNome(string nome){ ···
    0 references
    public void insereSobreNome(string sobrenome){ ···
    0 references
    public void insereIdade(int idade){ ···

    1 reference
    public void mostraDados(){ ···
}
```

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

```
namespace projeto_pessoa;
0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        //Pessoa p1 = new Pessoa();
        Pessoa p1 = new Pessoa("Anderson",48);
        //p1.insereNome("Anderson");
        //p1.insereSobreNome("Vanin");
        //p1.insereIdade(48);
        p1.mostraDados();
    }
}
```

```
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()

```
namespace projeto_desconto;
2 references
class Produto
{
    0 references
    public Produto(){
        this.Nome = "TICTOC";
        this.Preco = 15;
    }
    0 references
    public Produto(string nome){
        this.Nome = nome;
        this.Preco = 10;
    }
    1 reference
    public Produto(string nome, double desconto){
        this.Nome = nome;
        this.Preco = 10*(1-(desconto/100));
    }
    5 references
    private string? Nome {get;set;}
    5 references
    private double Preco {get;set;}

    0 references
    public void CadastraNome(string nome){
        this.Nome = nome;
    }
    0 references
    public void CadastraPreco(double preco){
        this.Preco = preco;
    }

    1 reference
    public void mostraDados(){
        Console.WriteLine("Produto: " + this.Nome);
        Console.WriteLine("Preço: " + this.Preco);
    }
}
```

Program.cs > {} projeto_desconto > projeto_desconto.Program

```
namespace projeto_desconto;
0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        // SITUAÇÃO 1
        //Produto p1 = new Produto();
        // SITUAÇÃO 2
        //Produto p1 = new Produto("Caneta");
        // SITUAÇÃO 3
        Produto p1 = new Produto("Caneta",50);
        p1.mostraDados();
    }
}
```

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.