Disciplina: POO - Programação Orientada a Objetos

Construtor
Variáveis: static, const, readonly

Profª. Drª. Giovana Angélica Ros Miola
giovana.miola@fatec.sp.gov.br

Construtor

- Função do construtor - inicializar o objeto

- Quando uma classe não tem nenhum construtor, C# coloca um **construtor padrão** dentro da classe

- Esse construtor não recebe argumentos e não executa nenhuma ação, ou seja, um construtor que não recebe nenhum argumento e tem o corpo vazio

Construtor de instância

- Para criar uma objeto, usa-se a palavra chave new, seguida de uma chamada ao **método construtor**, cujo nome é idêntico ao da classe:

```
NomeDaClasse nomeDoObjeto = new NomeDaClasse( );
```

- O método construtor não precisa ser incluído explicitamente na classe, mas pode ser incluído para realizar tarefas no ato da criação/instanciação de um objeto

Múltiplos Construtores Dentro da Classe

```
class Conta
{
    //declaração de atributos
    public int numero;
    public string titular;
    public double saldo;
    //declaração dos construtores
    public Conta()
    {
    }
    public Conta(int numero)
    {
        this.numero = numero;
    }
    public Conta(int numero,string titular)
    {
        this.numero = numero;
        this.titular = titular;
    }
    //declaração de métodos
    public bool Sacar(double valor)
```

```
class Program
{
    static void Main(string[] args)
    {
        //construtor padrão
        Conta c1 = new Conta();
        //construtor com um parâmetro
        Conta c2 = new Conta(1);
        //construtor com dois parâmetros
        Conta c3 = new Conta(2, "Ana");
    }
}
```

- Sobrecarga de métodos construtores

Exercício

- Crie um projeto novo chamado ConstrutorFuncionario
- Crie a classe Funcionário(código, nome, salário) e implementem o método Mostrar()
- Crie no mínimo 3 construtores
- Na main instancie objetos referentes aos 3 construtores
- Mostre quantas instâncias ocorreram na main

Variáveis de instância e variáveis de classe

- Uma **variável de instância** é uma variável cujo valor é **específico ao objeto** e não à classe. Uma variável de instância em geral possui um valor diferente em cada objeto representante da classe.

- Uma **variável de classe** é uma variável cujo valor é **comum a todos os objetos** representantes da classe. Mudar o valor de uma variável de classe em um objeto, automaticamente muda o valor para todos os objetos instâncias da mesma classe. Um exemplo óbvio de uma variável de classe seria o número de instâncias desta classe que já foram criadas.

Variável de Classe / Diretiva static

- Alguns métodos podem ser utilizados independentemente de um objeto, pois o algoritmo não precisa (ou não deve) conhecer os valores aplicados àquela instância
- Certas classes tem funções utilitárias e não faz sentido instanciar objetos para elas
- Exemplo, imagine uma função de conversão de câmbio monetário:
  - A classe câmbio poderia oferecer um método desta conversão, cujos os valores poderiam ser parâmetros da moeda desejada

Diretiva static

- Esta diretiva é aplicada também a atributos. Às vezes, alguns objetos necessitam compartilhar o valor de um ou mais atributos entre objetos da mesma classe
- Use a diretiva static para:
  - Garantir que o valor de um atributo seja o mesmo para qualquer objeto da classe
  - Garantir que um método ou atributo seja acessado independentemente da instanciação de um objeto

Variável de classe

- Uma variável é considerada como de instância por "default".

- **Para declarar uma variável de classe, acrescenta-se a palavra-chave static**.

- Exemplo:

```
static int numeroDeInstanciasDestaClasse;
```

Diretiva static / Variável de Classe

Observação

- É um erro usar um modificador **virtual**, **abstract** ou **override** em um acessador de uma propriedade que utiliza a diretiva **static**.

Exemplo

```
class Conta
{
    //declaração de atributos
    public int numero;
    public string titular;
    public double saldo;
    public static int qtdeContas=0;
    //declaração dos construtores
    public Conta()
    {
        qtdeContas++;
    }
    public Conta(int numero)
    {
        this.numero = numero;
        qtdeContas++;
    }
    public Conta(int numero,string titular)
    {
        this.numero = numero;
        this.titular = titular;
        qtdeContas++;
    }
    //declaração de métodos
    public static int MostraQtdeContas()
    {
        return qtdeContas;
    }
```

```
class Program
{
    static void Main(string[] args)
    {
        //construtor padrão
        Conta c1 = new Conta();
        //construtor com um parâmetro
        Conta c2 = new Conta(1);
        //construtor com dois parâmetros
        Conta c3 = new Conta(2, "Ana");
        Console.WriteLine("Quantidade de contas cadastradas: "+
            Conta.MostraQtdeContas());
        Console.ReadKey();
    }
```

```
Quantidade de contas cadastradas: 3
```

Construtores estáticos

- Inicializa membros/atributos estáticos
- Construtores estáticos não têm parâmetros
- Não pode ser usado modificadores de acesso
- Ele é executado implicitamente primeiro que os outros construtores

```
class Conta
{
    //declaração de atributos
    public int numero;
    public string titular;
    public double saldo;
    public static int qtdeContas=0;
    //declaração de construtor estático
    static Conta()
    {
        qtdeContas=10;
    }
    //declaração dos construtores
    public Conta()
    {
        qtdeContas++;
    }
    public Conta(int numero)
    {
```

```
static void Main(string[] args)
{
    //construtor padrão
    Conta c1 = new Conta();
    Console.WriteLine(c1.MostraQtdeContas());
```

```
11
```

Métodos estáticos

- Por "default", um método efetua uma determinada operação sobre um determinado objeto, ou seja uma instância da classe na qual o método está declarado.
- Existem métodos que realizam **operações genéricas**, não relativas a uma instância particular. Tais métodos, são chamados estáticos e são declarados, acrescentando-se a palavra-chave **static** à declaração do método.
- Métodos, podem ser definidos como **static** e quando for usá-lo no método Main(), apenas use o NomeDaClasse.NomeDoMetodo(), sem necessitar instanciar um objeto

Classe estática

- Contêm apenas membros/atributos estáticos
- Não podem ser instanciadas
- É recomendado o uso de classes estáticas para manter métodos não associados com um objeto específico
- Exemplo:

```
public static class Math
{
     //...
}
static void Main(string[] args)
{    // resposta 8
     Console.WriteLine(Math.Sqrt(64));
}
```

```
public static class Funcoes
{
    // Esse método é válido dentro de
    // uma classe estática.
     public static bool MetodoEstatico()
     {
         // implementação
     }
    // Esse método não é válido dentro
    // de uma classe estática.
    public bool MetodoInstancia()
    {
         // implementação
    }
}
```

Variáveis constantes

- Tais variáveis são caracterizadas pela palavra-chave `const`, e por convenção recebem usualmente nomes escritos inteiramente em maiúsculas.
- Quando **declaradas devem ser inicializadas**
- Por exemplo, encontramos uma constante matemática bem conhecida.

```
public const double PI = 3.14159265358979;
```

Variáveis readonly

```
class Conta
{
    //declaração de atributos
    public int numero;
    public string titular;
    public double saldo;
    public readonly string endereco;
    public void AlteraEndereco()
    {
        endereco = "Rua Terezina, 75";
    }
```

(campo) string Conta.endereco

Um campo somente leitura não pode ser atribuído (exceto em um construtor ou inicializador de variável)

```
    public Conta(string endereco)
    {
        this.endereco = endereco;
        Console.WriteLine("Endereço: " + endereco);
    }
```

- As variáveis declaradas como readonly, não podem ser alteradas por quaisquer métodos, APENAS pelo **construtor da classe**
- Pode-se alterar o valor de um atributo declarado como readonly em tempo de execução (Construtor) ou pela atribuição de um valor em tempo de execução
- Pode-se declarar um campo readonly como static

```
public static readonly string endereco = “Rua Parana, 10”;
```

Exercício

- Criar a classe funcionário que tenha como atributos: código, nome, salário.
- Implemente os conceitos de orientação a objetos utilizados até o momento, para isso instancie alguns funcionários e identifique quantos tem.
- Crie um método que calcule e mostre o novo salário aplicando um reajuste salarial a partir de um valor informado.
- Use o construtor static para predeterminar que o código do funcionário comece de 100
- Realize 3 instâncias

Exercício

- Crie a classe Produto (código, nome, preço)
- Implemente no mínimo 3 construtores.
- Na função main() instancie alguns objetos produto, correspondentes a implementação.
- Também apresente a quantidade de instâncias (use o modificador static).