Linguagem de Programação II

Ciência da Computação

Prof. Me. Renato Carioca Duarte

Encapsulamento

- O encapsulamento esconde detalhes de implementação do objeto (métodos) e o que sobra visível é a sua interface, isto é, o conjunto de todas as mensagens a que ele pode responder.
- Uma vez que o objeto é encapsulado, seus detalhes de implementação não são mais imediatamente acessíveis.
- Ao invés disso, eles são empacotados e são somente indiretamente acessíveis através da interface do objeto.
- A única forma de acessar a um objeto encapsulado é através da troca de mensagens: é enviada uma mensagem ao objeto, o objeto mesmo seleciona o método pelo qual ele irá reagir à mensagem.

Encapsulamento

- Os modificadores de acesso são palavras-chave usadas para especificar a acessibilidade declarada de um membro ou de um tipo. Esta seção apresenta os quatro modificadores de acesso:
- Os seguintes níveis de acessibilidade podem ser especificados usando os modificadores de acesso:

  - **Public**: Com este modificador, o acesso é livre em qualquer lugar do programa.
  - **Private**: Com este modificador, o acesso é permitido somente dentro da classe onde ele foi declarado. Por padrão, é a visibilidade definida para métodos e atributos em uma classe.
  - **Protected**: Com este modificador, apenas a classe que contém o modificador e os tipos derivados dessa classe tem o acesso (herança).

Atributos privados

- Atributos de uma classe são detalhes da implementação que não devem ser visíveis fora da classe.

```
class Conta
{
    // numero, titular e saldo são atributos do objeto
    private int numero;
    private Cliente titular;
    private double saldo;

    public bool Saca(double valor)
    {
        if (this.saldo >= valor)
        {
```

Atributos privados

- Note que com private, as atribuições do atributo estão dando erro.
- Como podemos atribuir valores para os atributos da classe?
- Agora o programador é forçado a passar pelos métodos para conseguir manipular o saldo

```
private void Button1_Click(object sender, EventArgs e)
{
    Cliente cli = new Cliente();

    Conta c = new Conta();
    c.numero = 1;
    c.saldo = 100;
    c.titular = cli;
```

Métodos SET

- Com atributos private, precisamos de métodos para ATRIBUIR valor para eles.
- São métodos popularmente chamados de "SET".
- Precisamos fazer 3 métodos novos:
  - setNumero
  - setSaldo
  - setTitular

Métodos SET

- Com atributos private, precisamos de métodos para ATRIBUIR valor para eles.
- São métodos popularmentes chamados de "SET"

```
class Conta
{
    // numero, titular e saldo são atributos do objeto
    private int numero;
    private Cliente titular;
    private double saldo;

    public void setNumero (int n)
    {
        numero = n;
    }
```

Métodos SET

- Usando **this**, podemos ter o argumento com mesmo nome do atributo

```
class Conta
{
    // numero, titular e saldo são atributos do objeto
    private int numero;
    private Cliente titular;
    private double saldo;

    public void setNumero (int numero)
    {
        this.numero = numero;
    }
```

Exercícios

1. Fazer os métodos setSaldo e setTitular

2. Alterar o método *Button1_Click* visando atribuição de valores para número, saldo e titular da conta.

Resposta Exercícios

1. Definição de métodos SET

```
public void setNumero (int numero)
{
    this.numero = numero;
}

public void setSaldo (double saldo)
{
    this.saldo = saldo;
}

public void setTitular (Cliente titular)
{
    this.titular = titular;
}
```

Resposta Exercícios

2. Utilização de métodos SET

```
private void Button1_Click(object sender, EventArgs e)
{
    Cliente cli = new Cliente();
    cli.nome = "Victor";
    cli.cpf = "888888889";
    cli.endereco = "Rua XXXX, 21";
    cli.rg = "2222222222";

    Conta c = new Conta();
    c.setNumero (1);
    c.setSaldo (100);
    c.setTitular (cli);
```

Métodos GET

- Com atributos private, precisamos de métodos para CONSULTAR os valores que eles possuem.
- São métodos popularmente chamados de "GET".
- Precisamos fazer 3 métodos novos:
  - getNumero
  - getSaldo
  - getTitular

Métodos GET

- O método apenas retorna o valor que está dentro do atributo.

```
public int getNumero()
{
    return this.numero;
}
```

Exercícios

3. Fazer os métodos getSaldo e getTitular

4. Alterar o método *Button1_Click* visando exibição no Console dos valores para número, saldo e nome titular da conta.

Resposta Exercícios

```
public double getSaldo()
{
    return this.saldo;
}

public Cliente getTitular()
{
    return this.titular;
}
```

Resposta Exercícios

```
private void Button1_Click(object sender, EventArgs e)
{
    Cliente cli = new Cliente();
    cli.nome = "Victor";
    cli.cpf = "888888889";
    cli.endereco = "Rua XXXX, 21";
    cli.rg = "2222222222";

    Conta c = new Conta();
    c.setNumero (1);
    c.setSaldo (100);
    c.setTitular (cli);

    Console.WriteLine(c.getNumero());
    Console.WriteLine(c.getSaldo());
    Console.WriteLine(c.getTitular().nome);

    if (c.Saca(30.0))
```

Propriedade (Property)

- Uma propriedade (Property) é um membro de uma classe que fornece um mecanismo flexível para ler, gravar ou calcular o valor de um dado em particular.
- As propriedades permitem que uma classe exponha de uma maneira pública a obtenção e definição destes valores.

Propriedade (Property)

```
class Pessoa
{
    private string nome;
    public string Nome
    {
        set
        {
            nome = value;
        }

        get
        {
            return nome;
        }
    }
}
```

Propriedade (Property)

- O atributo nome é privado
- A propriedade Nome é pública e é utilizada para acessar o atributo nome.

```
private void Button1_Click(object sender, EventArgs e)
{
    Pessoa p1 = new Pessoa();
    p1.Nome = "jose";
    Console.WriteLine(p1.Nome);
```

Propriedade (Property)

- Desde o C# 3.0 (estamos na 8.0) foi incluído o recurso de Automatic Properties (Propriedades Automáticas), ou seja, não necessitamos mais declarar o Get nem o Set, aquela variável interna de controle também não (é gerenciado automático).

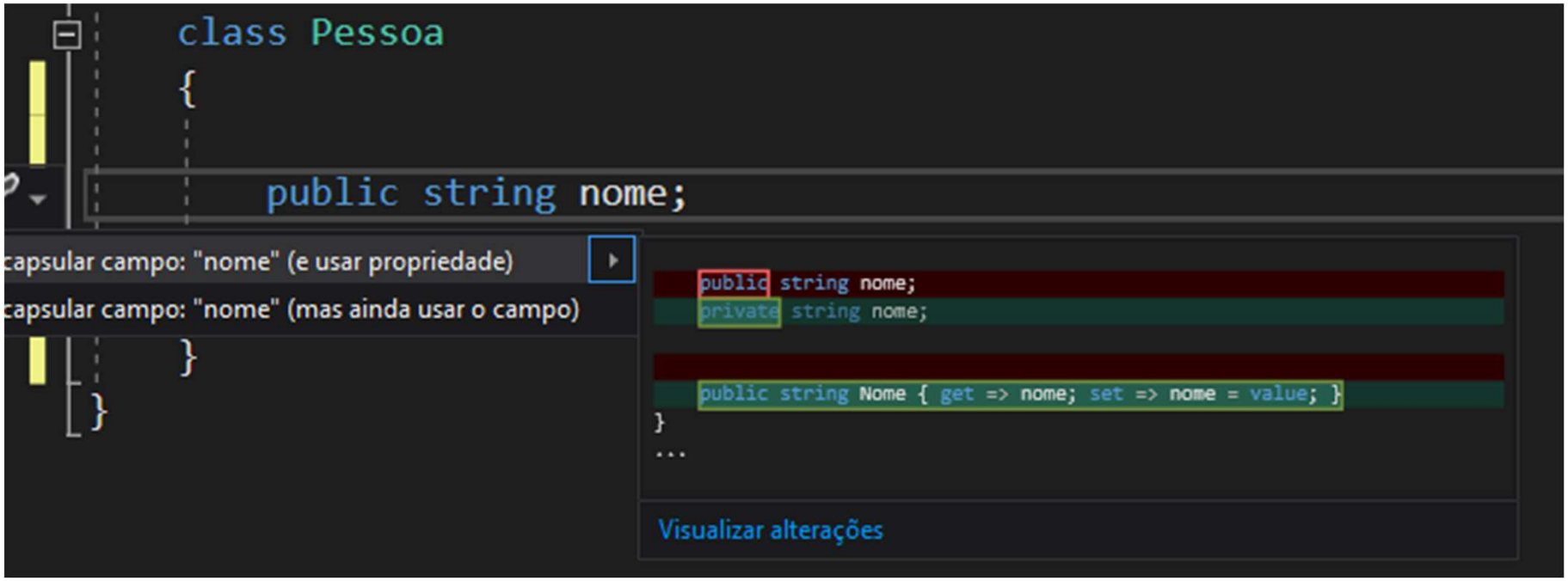
```
class Pessoa
{

    private string nome;
    public string Nome { get; set; }

}
```

Propriedade (Property)

- Se optar por permitir ao Visual Studio gerar o código encapsulado usando propriedades....



Propriedade (Property)

- .... o resultado será este:

```
class Pessoa
{

    private string nome;
    public string Nome { get => nome; set => nome = value; }

}
```

Exercício

- Encapsular os atributos da classe Cliente usando as propriedades.