LINGUAGEM DE PROGRAMAÇÃO

Orientada a objeto

Roteiro da aula

Conceitos de Programação Orientado a Objeto (POO):

- Condições de acessibilidade
- Encapsulamento
- Propriedades (get, set)
- Métodos

Condições de acessibilidade

- É a maneira como o atributo ou método será acessado
- Para ter acesso a classe deverá ser criado um objeto pertencente a classe
- O objeto é a ligação entre o usuário e a classe
- A classe fica protegida e o usuário não tem acesso direto a classe → **Encapsulamento**.
- Encapsulamento é o ato de esconder do usuário informações que não são de seu interesse (ex. SmartTV)
- A condição de acessibilidade é um dos mecanismos de encapsulamento de uma classe

Condições de Acessibilidade

Declaração	Definição
public	Acesso ilimitado
private	Acesso limitado à classe e seus membros
internal	Acesso limitado ao programa (assembly)
protected	Acesso limitado à classe, seus membros e seus derivados.
protected internal	Acesso limitado à classe, classes derivadas ou membros deste programa (assembly)

Encapsulamento

- O **encapsulamento** é uma técnica da orientação a objetos cuja finalidade é a de **proteger o acesso aos dados da classe**
- Quando criamos um atributo do tipo **private** dentro da classe, é vedado o acesso a este atributo fora da classe.
- Assim, o usuário que esteja trabalhando diretamente com a tela do programa, não teria acesso aos dados privados de uma classe.

Encapsulamento

- A **Propriedade** é um método especial que permite ao usuário acessar de maneira indireta os valores de um campo **private** de uma classe.
- A Propriedade possui dois **acessadores:**

  get (obter)

  set (atribuir)

GET (obter)

- É executado sempre que o campo **privado** é lido.
- Possui um valor de retorno que corresponde ao tipo da variável.
- Neste caso ele retorna sempre o valor do campo **privado**.
- Sintaxe:

```
private int codigo;  //atributo com iniciais minúsculo

public int Codigo   //propriedade com maiúsculo
{
        get {   return codigo;   }
}
```

SET (atribuir)

- É chamado sempre que se atribui um valor ao campo privado.
- Embora aparente não ter nenhum parâmetro, na verdade ele possui um chamado **value** (valor) que contém o valor atribuído ao campo.
- Sintaxe:

```
private int codigo;  //atributo com iniciais minúsculo

public int Codigo    //propriedade com maiúsculo
{
        set { codigo = value;   }
}
```

Padrões adotados no C#

- **camelCase**
- Significa que ao concatenar as palavras, a primeira palavra inicia em minúscula e a primeira letra de cada palavra concatenada em maiúscula. Exemplo:

```
String nomeAluno;
int codigo;
```

- O padrão camelCase é utilizado em **variáveis locais ou privadas**

Padrões adotados no C#

- **PascalCase**
- Significa que ao concatenar as palavras, a primeira palavra inicia em maiúscula e a primeira letra de cada palavra concatenada em maiúscula. Exemplo

```
String NomeAluno;
int Codigo;
```

No C# utilizamos o padrão PascalCase para nomes de **classes, métodos e propriedades**.

Métodos

- É importante criar métodos para acessar os dados, impedindo o acesso aos objetos.
- Um método irá determinar o comportamento dos objetos
- É o método que determina a ação que a classe do objeto deverá processar
- Poderá não retornar um valor → public **void** Abrir() { }
- Poderá retornar um valor

→ public **int** Abrir()
{ ...comandos;
return valor;
}

Regras básicas para criar métodos

- Os nomes dos métodos devem ser significativos, ou seja, ao ler o nome do método, não deixar duvidas sobre qual é a sua finalidade.
- Utilizar sempre o padrão **PascalCase**
- O método deve ter uma **finalidade bem especifica.**
- O método pode ou não retornar um tipo de dados. Caso não retorne nada, utilizar a palavra **void**, antes do nome do método.
- Quando criamos o método dentro da classe, utilizamos a variável privada e não o nome da propriedade. As propriedades são utilizadas para atribuir e receber valores das classes

Método sem Parâmetro

- Ocorre quando nenhum valor é enviado para dentro dele no momento em que é chamado:

```
class Cliente
{
    private int produtosAdquiridos;
     public bool ClienteEspecial()
     {
        if (produtosAdquiridos < 250)
            return false;
        return true;
      }
}
```

Método com passagem de Parâmetro

```
class Cliente
{
    private int produtosAdquiridos;

   public void DefineProdutosAdquiridos(int quantidade)
   {
            produtosAdquiridos = quantidade;
    }
}
```

Programa de Loteamento

Implemente o projeto da atividade 1 - Loteamento - com as seguintes informações:

Determinar a área de cada cômodo da casa da seguinte maneira:

- Casa: 80% do terreno
- Quarto: 20% da área da casa. Quantidade: 2 quartos.
- Sala: 25% da área da casa. Quantidade: 1 sala.
- Banheiro: 8 % da área da casa. Quantidade: 2 banheiros.
- Cozinha: 14 % da área da casa. Quantidade: 1 cozinha.
- Área de Serviço: 5% da área da casa. Quantidade: 1 area
- Informe o resultado de cada cálculo. Todos os cálculos deverão ser desenvolvidos dentro de uma classe chamada Casa.