**Introdução à Orientação a Objetos em C#**

A Orientação a Objetos é um paradigma de programação que busca representar o mundo real através de objetos. Esses objetos possuem características (atributos) e comportamentos (métodos) que podem ser utilizados para modelar sistemas complexos.

Em C#, a Orientação a Objetos é amplamente utilizada e é uma das principais características da linguagem. Nesta aula, vamos explorar os conceitos básicos da Orientação a Objetos em C# e apresentar alguns exemplos de código.

**Classes e Objetos**

Em C#, um objeto é uma instância de uma classe. Uma classe é uma estrutura que define as características e comportamentos de um objeto. Por exemplo, podemos ter uma classe chamada "Pessoa" que define os atributos de uma pessoa (nome, idade, endereço) e os métodos que a pessoa pode executar (andar, falar, comer).

Para criar um objeto em C#, precisamos primeiro definir a classe correspondente. Vamos criar uma classe simples chamada "Carro":

```csharp

public class Carro {

// Atributos

public string marca;

public string modelo;

public int ano;

private int velocidadeAtual;

// Construtor

public Carro(string marca, string modelo, int ano) {

this.marca = marca;

this.modelo = modelo;

this.ano = ano;

this.velocidadeAtual = 0;

}

// Métodos

public void Acelerar(int quantidade) {

this.velocidadeAtual += quantidade;

Console.WriteLine("Acelerando. Velocidade atual: " + this.velocidadeAtual + " km/h");

}

public void Frear(int quantidade) {

this.velocidadeAtual -= quantidade;

if (this.velocidadeAtual < 0) {

this.velocidadeAtual = 0;

}

Console.WriteLine("Freando. Velocidade atual: " + this.velocidadeAtual + " km/h");

}

public void Ligar() {

Console.WriteLine("Carro ligado!");

}

public void Desligar() {

Console.WriteLine("Carro desligado!");

}

}

```

Nesse exemplo, a classe Carro possui os seguintes atributos:

- `marca`: uma string que representa a marca do carro

- `modelo`: uma string que representa o modelo do carro

- `ano`: um inteiro que representa o ano de fabricação do carro

- `velocidadeAtual`: um inteiro que representa a velocidade atual do carro (em km/h)

A classe também possui um construtor que recebe a marca, o modelo e o ano do carro como parâmetros e inicializa os atributos correspondentes. Além disso, a classe possui os seguintes métodos:

- `Acelerar`: recebe uma quantidade de aceleração (em km/h) e aumenta a velocidade atual do carro em função dessa quantidade.

- `Frear`: recebe uma quantidade de frenagem (em km/h) e diminui a velocidade atual do carro em função dessa quantidade. Caso a velocidade atual fique menor que zero, ela é ajustada para zero.

- `Ligar`: exibe uma mensagem indicando que o carro foi ligado.

- `Desligar`: exibe uma mensagem indicando que o carro foi desligado.

Vou mostrar como utilizar a classe `Carro` que criamos no programa `Main` em C#:

```csharp

class Program {

static void Main(string[] args) {

// Criando um objeto da classe Carro

Carro meuCarro = new Carro("Fiat", "Uno", 2010);

// Utilizando os métodos da classe Carro

meuCarro.Ligar();

meuCarro.Acelerar(20);

meuCarro.Acelerar(30);

meuCarro.Frear(10);

meuCarro.Frear(50);

meuCarro.Desligar();

}

}

```

Nesse exemplo, começamos criando um objeto da classe `Carro` chamado `meuCarro`, passando a marca `"Fiat"`, o modelo `"Uno"` e o ano de fabricação `2010` como parâmetros para o construtor.

Em seguida, utilizamos alguns métodos da classe `Carro` para ligar o carro (`Ligar()`), acelerar em duas etapas (`Acelerar(20)` e `Acelerar(30)`), frear em duas etapas (`Frear(10)` e `Frear(50)`) e finalmente desligar o carro (`Desligar()`).

Ao executar esse programa, o resultado esperado seria a exibição das mensagens correspondentes a cada ação realizada pelo carro.

**Encapsulamento**

O encapsulamento é um dos princípios fundamentais da Orientação a Objetos. Ele consiste em esconder os detalhes internos de uma classe e expor apenas uma interface pública para o mundo exterior. Isso permite que o código que utiliza a classe não precise conhecer os detalhes internos da implementação.

Em C#, podemos utilizar modificadores de acesso para controlar o acesso aos membros de uma classe. Os principais modificadores de acesso são:

- public: o membro pode ser acessado por qualquer código que utilize a classe.

- private: o membro só pode ser acessado dentro da própria classe.

- protected: o membro só pode ser acessado dentro da própria classe ou por classes derivadas (herdadas).

Os métodos GET e SET são uma forma de implementar o encapsulamento em programação orientada a objetos (POO).

O encapsulamento é um dos pilares da POO e tem como objetivo proteger os atributos de uma classe, tornando-os privados e permitindo que somente os métodos da própria classe tenham acesso a eles. Isso evita que os atributos sejam modificados de maneira indevida por outras partes do programa.

Os métodos GET e SET são usados para permitir o acesso controlado aos atributos privados de uma classe. O método GET é utilizado para obter o valor de um atributo privado, enquanto o método SET é utilizado para definir o valor de um atributo privado.

Por exemplo, suponha que temos uma classe `Pessoa` com um atributo privado `nome`. Para permitir que outros objetos acessem esse atributo de maneira controlada, podemos criar um método GET chamado `getNome` que retorna o valor do atributo `nome`. Da mesma forma, podemos criar um método SET chamado `setNome` que recebe um parâmetro `novoNome` e define o valor do atributo `nome` para esse novo valor.

Dessa forma, outros objetos podem acessar o nome de uma pessoa somente através desses métodos, garantindo que o valor do atributo seja modificado somente através do método SET e que o valor seja obtido somente através do método GET. Isso permite que a classe mantenha o controle sobre seus atributos e evita que esses atributos sejam modificados de maneira indevida por outras partes do programa.

Vou mostrar um exemplo de como usar os métodos GET e SET em uma classe em C#:

```csharp

public class Pessoa {

// Atributos privados

private string nome;

private int idade;

// Método GET para o atributo nome

public string getNome() {

return nome;

}

// Método SET para o atributo nome

public void setNome(string novoNome) {

nome = novoNome;

}

// Método GET para o atributo idade

public int getIdade() {

return idade;

}

// Método SET para o atributo idade

public void setIdade(int novaIdade) {

idade = novaIdade;

}

}

```

Nesse exemplo, a classe `Pessoa` possui dois atributos privados (`nome` e `idade`) e quatro métodos públicos (`getNome`, `setNome`, `getIdade` e `setIdade`).

Os métodos `getNome` e `getIdade` são métodos GET que retornam o valor dos atributos privados `nome` e `idade`, respectivamente. Já os métodos `setNome` e `setIdade` são métodos SET que definem o valor dos mesmos atributos.

Dessa forma, se quisermos criar um objeto da classe `Pessoa` e definir seu nome e idade, podemos fazer da seguinte forma:

```csharp

Pessoa pessoa = new Pessoa();

pessoa.setNome("João");

pessoa.setIdade(30);

```

E se quisermos obter o nome e a idade dessa pessoa, podemos fazer da seguinte forma:

```csharp

Console.WriteLine("Nome: " + pessoa.getNome());

Console.WriteLine("Idade: " + pessoa.getIdade());

```

Isso garante que os valores dos atributos `nome` e `idade` só podem ser acessados e modificados através dos métodos GET e SET, garantindo o encapsulamento do objeto `Pessoa`.