> DESENVOLVER APLICAÇÕES BACK-END PARA WEB

Apresentações

Lucas Rodrigues Monteiro

Bacharel em Sistemas da Informação e Analista de Sistemas

lucas.monteiro@educacao.senac.rio

21 97024-9609

Fórum: www.fasim.com.br

GitHub lucasrm1981

Currículo Lates

LinkedIn

> Orientação a Objeto no C#

> O que é Orientação a Objeto

A orientação a objetos é uma ponte entre o mundo real e virtual, a partir desta é possível transcrever a forma como enxergamos os elementos do mundo real em código fonte, a fim de nos possibilitar a construção de sistemas complexos baseados em objetos.

O primeiro passo para utilizar o paradigma da orientação a objetos é perceber o universo em que se deseja trabalhar, uma vez que possuímos essa informação é necessário descrever a estrutura desses elementos, ou seja quais são as características mais marcantes destes para o desenvolvimento do sistema proposto.

Vejamos como descrever um objeto “Pessoa”. As perguntas a serem feitas são:
“O que é uma pessoa?”,
”Quais são as características de uma pessoa?”,
”Como uma pessoa se comporta?”.

> Orientação a Objeto no C#

> O que é Orientação a Objeto

Após responder a este questionário, teremos condições de modelar o objeto da forma como queremos. Vamos responder essas perguntas:

O que é uma pessoa?

Quais são as características de uma pessoa?

Como uma pessoa se comporta?

> Orientação a Objeto no C#

> O que é Orientação a Objeto

Após responder a este questionário, teremos condições de modelar o objeto da forma como queremos. Vamos responder essas perguntas:

O que é uma pessoa?

A pessoa é um ser do mundo real que interage com toda a natureza.

Quais são as características de uma pessoa?

Uma pessoa possui: nome, olhos, boca, braços, pernas, cabelos e etc.

Como uma pessoa se comporta?

Uma pessoa corre, anda, fala, pula, come e etc.

> Classe Orientação a Objeto

Projetamos o nosso elemento do mundo real, agora como iremos traduzir isto para o código fonte? Antes disso é preciso compreender o que foi feito até agora veja os passos:

· Descrever uma estrutura de uma pessoa

· Descrever as características de uma pessoa

· Descrever o comportamento de uma pessoa

Quando descrevemos de estrutura de um elemento, na verdade criamos uma especificação básica do que todo elemento daquele tipo deve ter, isso se chama **Classe**.

Tudo que faz parte de uma pessoa deve estar contido neste bloco de chaves.

```
class Pessoa
        {

        }
```

> Orientação a Objeto no C#

> Atributos Orientação a Objeto

Uma pessoa possui características que as diferem de outros seres do mundo real, isso no mundo da programação é chamado de **Atributos.**

```
class Pessoa
   {
      string nome;
      int olhos,bracos,pernas;
      string cor_olhos;
      string cor_cabelos;
    }
```

Os atributos devem vir acompanhados dos tipos de dados e também dos modificadores de acesso, neste caso não estamos declarando nenhum modificador de acesso, o C# implicitamente introduz a palavra “private” antes dos atributos e métodos (veremos a seguir). Classes só podem ser “public” ou “internal”.

> Métodos Públicos

Uma pessoa também possui um comportamento diferenciado de todos os outros elementos da natureza, na POO isso é conceituado como **métodos** .

Criamos e modelamos o objeto. Isso pode parecer simples mas no entanto é fundamental para que um projeto orientado a objetos tenha sucesso que o programador de sistemas conheça os conceitos abordados anteriormente.

```
class Pessoa
    {
        string nome;
        int olhos, bracos, pernas;
        string cor_olhos;
        string cor_cabelos;
         public void andar(int
velocidade)
        {
            // Ande um pouco
        }
        public void falar()
        {
            // Converse mais
        }
         public void comer()
        {
            // alimente-se mais
        }
    }
```

> Orientação a Objeto no C#

> Reuso de Código

Um dos pontos fortes da Orientação a objetos é o reuso de código, ou seja, permitir que outras rotinas utilizem funções predefinidas no sistema.

Ou seja, evitar escrever uma rotina várias vezes em locais diferentes.

A nossa classe pessoa possui alguns atributos e métodos que podem ser utilizados por outras classes, no entanto falta um pequeno detalhe para que essa comunicação seja realizada.

> Orientação a Objeto no C#

> Objeto

Para utilizar a classe Pessoa é necessária a criação de objetos vinculados a esta classe. Para fazer isto seguimos os seguintes passos:

· Declarar a classe em que deseja que o objeto pertença;

· Crie um nome para o objeto, neste caso declaramos como ”p”;

· Acrescente o sinal de “=”, que neste caso não é uma comparação e sim uma atribuição;

· “**new**” para a criação de todo objeto;

```
Pessoa p = new Pessoa();
```

> Orientação a Objeto no C#

> Executando

Serão iniciadas todas as classes descritas até o momento.

```
 namespace POO
{
    class Program
    {
        static void Main(string[] args)
        {
            Pessoa p = new Pessoa();
            p.nome("Tiago de Oliveira
Vale");
            Console.WriteLine("A
pessoa: " + p.nome() +
             " foi criada com
sucesso");
            Console.ReadKey();
        }
    }
}
```

> Orientação a Objeto no C#

> Construtor padrão

Existem objetos que possuem valores padrões, ou seja, nós queremos que toda instância de uma classe apresente os mesmos valores. No nosso exemplo declaramos variáveis de tipos primitivos com os seguintes nomes: bracos, pernas, olhos.

Para fazer isto podemos utilizar os getters and setters, mas só isso não ia adiantar, seria necessário declarar manualmente a quantidade de braços, olhos e pernas.

Assumindo que nosso sistema não terá ninguém com três olhos, dez pernas e cinco braços, faz-se necessário a criação de um construtor.

Observe que em nosso exemplo não criamos este construtor, no entanto se precisarmos de mais construtores é necessário que especifique o “construtor padrão”. Todo construtor deve possuir o mesmo nome da classe.

```
public Pessoa()
    {

    }
```

> Orientação a Objeto no C#

> Classe Pessoa final com construtores

Agora a estrutura está mondada com os dois construtores e suas informações.

```
 namespace POO
{
    public class Pessoa
    {
         public Pessoa(string cabelo)//Valor
obrigatório
        {
            Olhos = 2;//Valor default
            Bracos = 2;
            Pernas = 2;
            CorCabelo = cabelo;
         }
        public Pessoa() {
          }
        public string Nome { get; set; }
        public int Olhos { get; set; }
        public string CorCabelo { get; set; }
        public int Bracos { get; set; }
        public int Pernas { get; set; }
     }
}
```

> Orientação a Objeto no C#

> Passagem dos valores para o Objeto Pessoa

Agora a estrutura está mondada com os dois construtores e suas informações.

```
namespace POO
{
    class Program
    {
        static void Main(string[] args)
        {
            Pessoa p = new Pessoa();//UTILIZANDO CONSTRUTOR
DEFAULT
            p.Nome = "Tiago de Oliveira Vale";
            p.Bracos = 2;
            p.Pernas = 2;
            p.Olhos = 2;
            p.CorCabelo = "Preto";
            Console.WriteLine(p.Nome + " possui " + p.Bracos +
            " braços," + "\n" + p.Pernas + " pernas, \n " + p.Olhos +
            " olhos e cabelo " + p.CorCabelo+"\n");

            Pessoa p1 = new Pessoa("Loiro");
            //UTILIZANDO CONSTRUTOR PERSONALIZADO
            p1.Nome = "Fulano";

            Console.WriteLine(p1.Nome+" possui "+p1.Bracos+
            " braços"+",\n"+p1.Pernas+" pernas, \n "+p1.Olhos+
            " olhos e cabelo "+p1.CorCabelo);


            Console.ReadKey();
        }
    }
}
```

_ sit dolor amet

> Links de referência

https://docs.microsoft.com/pt-br/dotnet/csharp/whats-new/csharp-version-history

https://www.microsoft.com/pt-br/store/collections/visualstudio?atc=true&icid=SMB_CP1_visualstudio&rtc=2

https://visualstudio.microsoft.com/pt-br/vs/pricing/?tab=individual

https://visualstudio.microsoft.com/pt-br/vs/community/