Curso de Orientação a Objetos em C#	Aula01
Professora: Luana Fernandes	Turma: Quinta – 19:30h

Conteúdo desta aula:

- Classes, atributos e objetos
- Métodos (Parte 1)

CLASSES, ATRIBUTOS E OBJETOS

Conforme seu projeto vai criando tamanho e complexidade, fica cada vez mais difícil mantê-lo. Para resolver esse problema, podemos usar o paradigma orientado a objetos, pois ele organiza seu código em classes que refletem o mundo real. Dessa forma, é muito mais fácil transformar em um programa os problemas reais que temos no mundo. Vamos imaginar o seguinte exemplo:

Crie um programa que leia as seguintes informações de um aluno: Nome, Idade, Matrícula e Endereço. Em seguida, exiba as informações na tela.

Ok, até aí poderíamos fazer um programa simples, embora já comecem a aparecer alguns problemas como: O que é Endereço? É o nome da rua? Precisa colocar o CEP? E se tiver complemento? Mas vamos ignorar isso e escrever algo.

```
Console.WriteLine("Escreva seu nome: ");
string nome = Console.ReadLine();

Console.WriteLine("Escreva sua idade: ");
int idade = int.Parse(Console.ReadLine());

Console.WriteLine("Escreva sua matrícula: ");
string matricula = Console.ReadLine();

Console.WriteLine("Escreva seu endereço: ");
string endereco = Console.ReadLine();

Console.WriteLine("Dados do aluno: ");
Console.WriteLine("Nome: " + nome);
Console.WriteLine("Idade: " + idade);
Console.WriteLine("Matrícula: " + matricula);
Console.WriteLine("Endereço: " + endereco);
```

Mas e se agora você precisar ler também os dados de um professor? Professor não tem matrícula, mas tem disciplina. Ele também tem nome, idade e endereço. Aí começa a zona, porque você vai começar a usar variáveis como:

- nomeDoAluno
- idadeDoAluno
- nomeDoProfessor
- enderecoDoProfessor

A semântica do seu código não deveria estar no nome que você escolhe para uma variável, isso é algo bem ruim. Para resolver esse problema, podemos criar **classes** com **atributos** ao invés de termos variáveis soltas sem nenhuma relação computacional entre elas. Veja como seria a criação das classes Professor e Aluno:

```
public class Professor { // classe
    public string nome;  // atributo
    public int idade;
    public string disciplina;
    public string endereco;
}
```

```
public class Aluno { // classe
    public string nome;  // atributo
    public int idade;
    public string matricula;
    public string endereco;
}
```

Parece muito mais limpo e organizado que aquelas variáveis nomeDoProfessor e enderecoDoAluno, né? Mas e agora como usar isso? Vamos por exemplo criar um professor e capturar suas informações.

```
Professor p = new Professor();  //criando um professor

Console.WriteLine ("Insira os dados do Professor: ");
Console.WriteLine ("Nome: ");
p.nome = Console.ReadLine();//salva no atributo nome
Console.WriteLine ("Idade: ");
p.idade = int.Parse(Console.ReadLine());
Console.WriteLine ("Disciplina: ");
p.disciplina = Console.ReadLine();
Console.WriteLine ("Endereco: ");
p.endereco = Console.ReadLine();

Console.WriteLine ("Dados do Professor: ");
Console.WriteLine ("Nome: " + p.nome); //exibe o atributo nome
Console.WriteLine ("Idade: " + p.idade);
Console.WriteLine ("Disciplina " + p.disciplina);
Console.WriteLine ("Endereço: " + p.endereco);
```

Criar um objeto significa criar a instância de uma classe, mas em termos simples, significa criar uma variáveis que é do tipo que você criou, por exemplo, Professor. A orientação a objetos nos permite trabalhar com nossos próprios "tipos". Não precisamos mais nos limitar aos tipos primitivos: int, double, *string** etc, podemos ter o tipo Aluno! E quem define como esse tipo vai ser, é você!

**String não é um tipo primitivo, mas sim uma Classe.*

A sintaxe em C# para criar um objeto é:

```
NomeDaClasse nomeDoObjeto = new NomeDaClasse( );

Aluno aluno = new Aluno( );
```

Além de atributos, nossa classe também pode ter métodos. Por exemplo, vamos criar um método para mostrar os dados de Aluno na tela.

```
public class Aluno { // classe
    public string nome;  // atributo
    public int idade;
    public string matricula;
    public string endereco;

    //método que recebe um aluno e exibe seus dados na tela
    public void MostraDadosDoAlunoNaTela() {
        Console.WriteLine ("Dados do Aluno: ");
        Console.WriteLine ("Nome: " + nome);
        Console.WriteLine ("Idade: " + idade);
        Console.WriteLine ("Turma " + turma);
        Console.WriteLine ("Matrícula " + matricula);
        Console.WriteLine ("Endereço: " + endereco);
    }
}
```

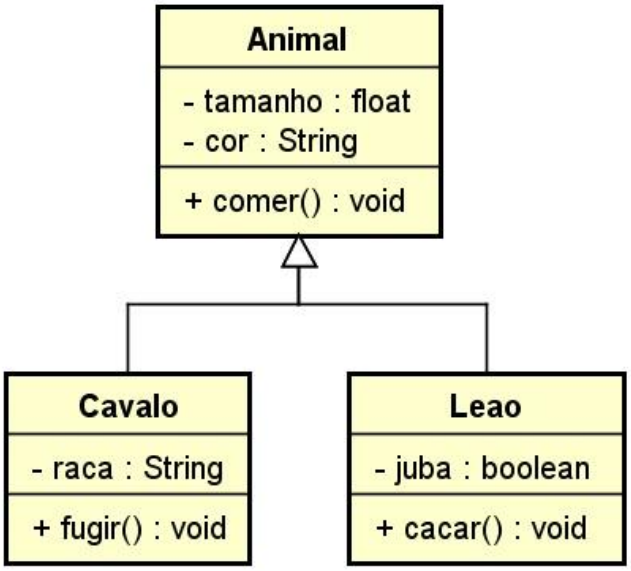
E então agora podemos chamar o método **MostraDadosDoAlunoNaTela()**, e tudo funcionará magicamente.

```
Aluno aluno = new Aluno( );
aluno. MostraDadosDoAlunoNaTela(); → Chamada do método
```

EXERCÍCIOS

Questão 1

Em programação, um **diagrama de classes** é uma representação da estrutura e relações das classes que servem de modelo para objetos. É uma modelagem muito útil para o desenvolvimento de sistemas, pois define as classes que o sistema necessita e é a base para a construção dos diagramas de comunicação, sequência e estados. Veja o exemplo a seguir:



Baseado na explicação, construa uma aplicação que implemente o diagrama de classe a seguir: