Linguagem de Programação II

Análise e Desenvolvimento de Sistemas

2º semestre de 2019

Prof. Me. Renato Carioca Duarte

Interfaces

- Interface é uma estrutura que permite ao desenvolvedor especificar todos os métodos e propriedades que ele deseja que as classes que a implementam disponibilizem.

- Utilizando uma interface, é possível separar completamente a definição (assinatura de métodos) de suas respectivas implementações, ou seja, de um lado (**interface**) temos a definição dos mesmos e do outro (**classe que implementa a interface**), temos a implementação dos mesmos.

- Podemos entender uma interface como sendo uma espécie de **contrato**, ou seja, se uma classe A implementa uma interface I, logo, I garante que A disponibilizará todos os métodos definidos em I. Caso este “contrato” seja quebrado, um erro será gerado.

Interfaces

- Interfaces são muito importantes pois, nos permitem separar o “o que” do “como”.
- A interface não se preocupa com a forma com a qual o método está sendo implementado e sim, que este método estará disponível a todos os objetos que a implementarem.
- A interface descreve a maneira como você deseja que o objeto seja utilizado ao invés da forma como ele foi implementado.

Exemplo

```
interface IAluno
{
    void SetMatriculaNome(int matr, string nome);
    String GetNome();
}

class Aluno : IAluno
{
    protected int Matricula;
    protected string Nome;

    public void SetMatriculaNome(int pID, string pName)
    {
        Matricula = pID;
        Nome = pName;
    }

    public String GetNome()  { return Nome; }

    static void Main(string[] args)
    {
        Aluno alu = new Aluno();
        alu.SetMatriculaNome(3456, "Ana Souza");
        Console.WriteLine(alu.GetNome());
        Console.ReadKey();
    }
}
```

Interfaces

- Uma interface é como uma classe base abstrata que contém apenas membros abstratos. Qualquer classe que implementa a interface deve implementar todos os seus membros.
- Uma interface não pode ser instanciada mas pode ser referenciada pelo objeto da classe que a implementa. Além disso, a referência da interface funciona como objeto de referência e se comporta como o objeto

```
static void Main(string[] args)
{
    IAluno alu = new Aluno();
    alu.SetMatriculaNome(3456, "Ana Souza");
    Console.WriteLine(alu.GetNome());
    Console.ReadKey();
}
```

- As interfaces podem conter propriedades, indexadores, métodos e eventos. As interfaces não têm implementações de métodos.
- Uma classe pode implementar várias interfaces. Uma classe pode herdar uma classe base e também implementar uma ou mais interfaces.

Interfaces – Herança Múltipla

- Outra importante justificativa para a utilização de interfaces em seus projetos .NET, é que, assim como o Java, o C# não trabalha com herança múltipla entre classes de forma nativa.
- Implementar este recurso somente é possível através da utilização de interfaces.
- Para superar esse problema, podemos usar interfaces onde uma classe pode implementar mais de uma interface ou de uma classe e de uma ou mais de uma interface.

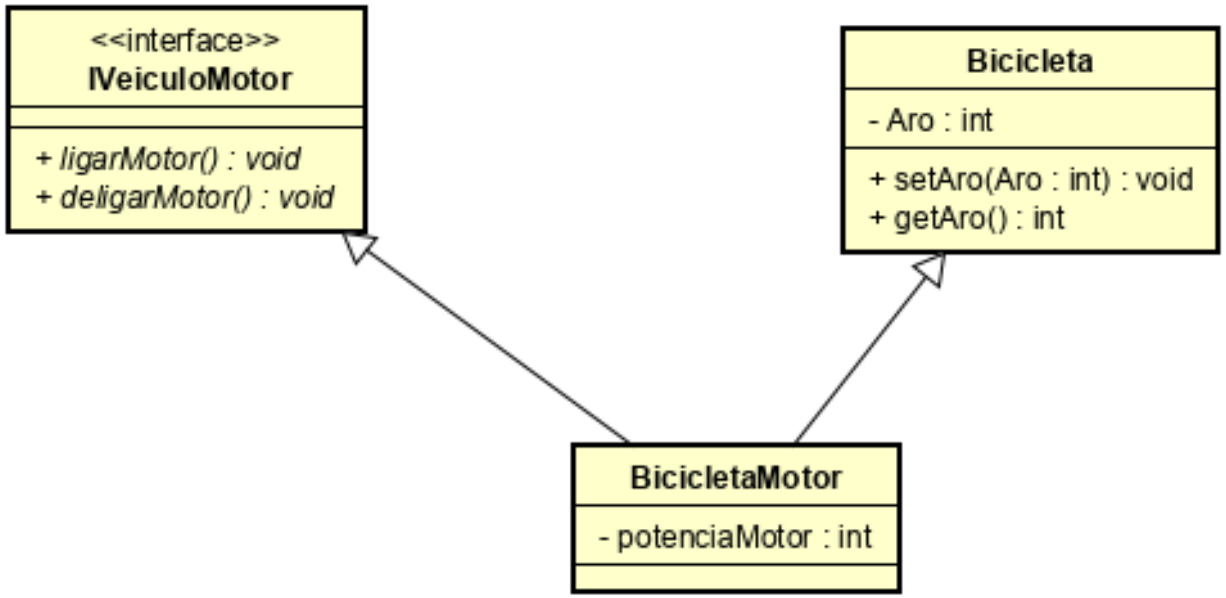
Interfaces – Herança Múltipla

```
public interface IMinhaInterface1
{
    void Metodo1();
}

public interface IMinhaInterface2
{
    void Metodo2();
}

public class MinhaClasse : IMinhaInterface1, IMinhaInterface2
{
    public void Metodo1() {  Console.WriteLine("Metodo1");  }
    public void Metodo2() {  Console.WriteLine("Metodo1");  }
}

class Program
{
    static void Main(string[] args)
    {
        MinhaClasse mc = new MinhaClasse();
        mc.Metodo1();
        mc.Metodo1();
    }
}
```

Exercício

Usar interface para implementar “herança múltipla” do diagrama abaixo:

1. Programar as 2 classes e a interface do desenho
2. Fazer um objeto da classe BicicletaMotor que recebe aro 26, que liga e depois desliga motor.



Exercício

- Implemente as classes e interfaces abaixo e crie um triatleta que faça todas 3 atividades.