IFRN

PROGRAMAÇÃO ORIENTADA A OBJETOS EM C#

Interfaces – Parte 01/02

Prof. Gilbert Azevedo

Objetivos

- Utilizar interfaces na escrita de programas orientados a objetos
- Programar a ordenação de vetores com as interfaces IComparable e IComparer
- Utilizar referências para interfaces
- Representar interfaces no diagrama de classes da UML

Motivação

- Padrões são largamente usados na indústria estabelecendo especificações para equipamentos e protocolos de comunicação
  - USB – Universal Serial Bus
  - SATA – Serial Advanced Technology Attachment
  - Display Port, HDMI, DVI, D-SUB
- Em hardwares, padrões estabelecem um contrato que deve ser seguido pelas partes interessadas
  - Tamanho e tipo de conectores, tamanho de equipamentos, furação, ...
- Como estabelecer um contrato em softwares?
  - Como definir que uma classe deve seguir uma especificação?

Ordenação de Vetores

- Ordenação de um vetor de inteiros usando Array.Sort

```
public static void Main (string[] args) {
  int[] v = { 2, 4, 6, 8, 1, 3, 5, 7 };
  Array.Sort(v);
  foreach(int i in v) Console.WriteLine(i);
}
```

Saída:
1
2
3
4
5
6
7
8

- Ordenação de um vetor de strings usando Array.Sort

```
public static void Main (string[] args) {
  string[] v = { "C#", "Java", "Python", "Basic" };
  Array.Sort(v);
  foreach(string i in v) Console.WriteLine(i);
}
```

Saída:
Basic
C#
Java
Python

Ordenação de Vetores

- Ordenação de um vetor de alunos usando Array.Sort

```
public static void Main () {
  Aluno a = new Aluno("Pedro", DateTime.Parse("01/01/1992"));
  Aluno b = new Aluno("Maria", DateTime.Parse("01/01/1993"));
  Aluno c = new Aluno("Paulo", DateTime.Parse("01/01/1991"));
  Aluno[] v = { a, b, c };
  Array.Sort(v);
  foreach(Aluno i in v)
    Console.WriteLine(i);
}
```

- Qual a saída?

```
class Aluno {
  private string nome;
  private DateTime nasc;
  public Aluno(string n, DateTime d) {
    this.nome = n;
    this.nasc = d;
  }
}
```

Contratos em Softwares

- A ordenação não funciona porque o método Sort estabelece que um contrato de comparação deve ser cumprido para ele funcionar
- A classe Aluno não cumpriu sua parte do contrato: definir como é feita a ordenação entre objetos da classe: nome e nascimento
- Em softwares, contratos são definidos por uma *Interface*

```
public static void Main () {
  ...
  Aluno[] v = { a, b, c };
  Array.Sort(v);
  ...
}
```

```
class Aluno {
  private string nome;
  private DateTime nasc;
  public Aluno(string n, DateTime d) {
    this.nome = n;
    this.nasc = d;
  }
}
```

Interface

- Interface é uma estrutura de software que define um comportamento requerido para um tipo (struct ou class), estabelecendo um “contrato”
- Exemplos de Comportamento
  - Comparação, Descarte, Iteração, Indexação, ...
- Operações dependentes de um comportamento específico
  - Ordenação de um vetor – método Sort
  - Liberação de recursos – instrução *using*
  - Iteração de coleções – instrução *foreach*
  - Indexação de coleções – operador de indexação [ ]

Interface IComparable

- De uma forma geral, interfaces definem métodos que um tipo deve possuir, ou seja, um comportamento esperado para o tipo
- A interface IComparable do *framework* define o método CompareTo como padrão de comparação entre objetos

```
interface IComparable {
    int CompareTo (object obj);
}
```

- Classes que implementem essa interface devem possuir o método CompareTo (público), que compara dois objetos da classe
- Definida a comparação, o método Array.Sort pode ser utilizado

Método CompareTo

- Muitos tipos no C# implementam a interface IComparable
  - public struct Int32 : IComparable, ...
  - public sealed class String : ... IComparable, ...
- A implementação do método CompareTo segue o seguinte padrão
  - -1: quando o objeto é menor que o outro
  - 0: quando os objetos tem o mesmo valor
  - 1: quando o objeto é maior que o outro

```
int a = 10, b = 20;
Console.WriteLine(a.CompareTo(b)); // -1
Console.WriteLine("Java".CompareTo("C#")); // 1
```

Ordenação de Alunos

- A classe Aluno precisa implementar IComparable para o código abaixo, necessário à ordenação, funcionar
  - O método CompareTo é invocado pelo método Array.Sort

```
class MainClass {
  public static void Main (string[] args) {
    Aluno a = new Aluno("Pedro", DateTime.Parse("01/01/1992"));
    Aluno b = new Aluno("Maria", DateTime.Parse("01/01/1993"));
    Aluno c = new Aluno("Paulo", DateTime.Parse("01/01/1991"));
    Console.WriteLine(a.CompareTo(b));
    Console.WriteLine(a.CompareTo(c));
    Console.WriteLine(b.CompareTo(c));
  }
}
```

Implementação de IComparable

- A classe Aluno deve informar a implementação de IComparable e definir o método público CompareTo

```
class Aluno : IComparable {
  private string nome;
  private DateTime nasc;
  public Aluno(string n, DateTime d) {
    this.nome = n;
    this.nasc = d;
  }
  public int CompareTo(object obj) {
    // a.CompareTo(b)
    // Comparação entre os objetos this e obj
  }
}
```

CompareTo na Classe Aluno

- Ordenação usando o atributo nome da classe Aluno
  - CompareTo deve retornar -1, 0 ou 1.

```
class Aluno : IComparable {
  ...
  public int CompareTo(object obj) {
    // a.CompareTo(b). Comparação entre os objetos this e obj
    Aluno x = this;         // this é Aluno
    Aluno y = (Aluno) obj; // type-cast - obj é object
    if (x.nome == y.nome) return 0;
    if (x.nome.CompareTo(y.nome) == -1) return -1;
    if (x.nome.CompareTo(y.nome) == 1) return 1;
    return 0;
  }
}
```

Classe Aluno

- CompareTo retorna o resultado da comparação do atributo nome

```
class Aluno : IComparable {
  private string nome;
  private DateTime nasc;
  public Aluno(string n, DateTime d) {
    this.nome = n;
    this.nasc = d;
  }
  public int CompareTo(object obj) {
    return nome.CompareTo(((Aluno)obj).nome);
  }
  public override string ToString() {
    return $"{nome} - {nasc:dd/MM/yyyy}";
  }
}
```

Array.Sort e CompareTo

- O método Sort da classe Array utiliza o método CompareTo para ordenar os elementos do vetor.
- Como o Array referencia seus elementos? object ou Aluno

```
class MainClass {
  public static void Main (string[] args) {
    Aluno a = new Aluno("Pedro", DateTime.Parse("01/01/1992"));
    Aluno b = new Aluno("Maria", DateTime.Parse("01/01/1993"));
    Aluno c = new Aluno("Paulo", DateTime.Parse("01/01/1991"));
    Aluno[] v = { a, b, c };
    Array.Sort(v);
    foreach(Aluno i in v)
      Console.WriteLine(i);
  }
}
```

Saída:
Maria - 01/01/1993
Paulo - 01/01/1991
Pedro - 01/01/1992

Referência para Interface

- O tipo object pode referenciar quaisquer objetos no C#
  - Apenas os métodos de object podem ser invocados

```
object a = 10;
object b = "C#";
Console.WriteLine(a); // 10
Console.WriteLine(b); // C#
```

- Variáveis da interface podem referenciar objetos que a implementam
  - Apenas os métodos da interface podem ser invocados

```
IComparable x = "Java";
IComparable y = "C#";
Console.WriteLine(x.CompareTo(y)); // 1
```

Referência para Interface

- Usando IComparable para referenciar alunos
  - Apenas o método CompareTo pode ser invocado

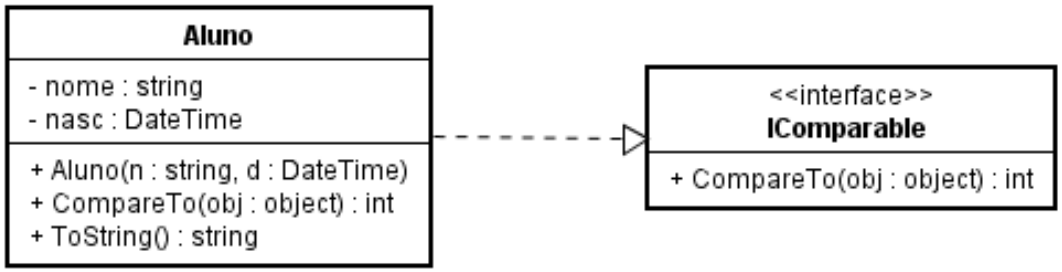
```
IComparable x = new Aluno("Pedro", DateTime.Parse("01/01/1992"));
IComparable y = new Aluno("Maria", DateTime.Parse("01/01/1993"));
IComparable z = new Aluno("Paulo", DateTime.Parse("01/01/1991"));
Console.WriteLine(x.CompareTo(y));
```

- O tipo do objeto é definido na sua criação

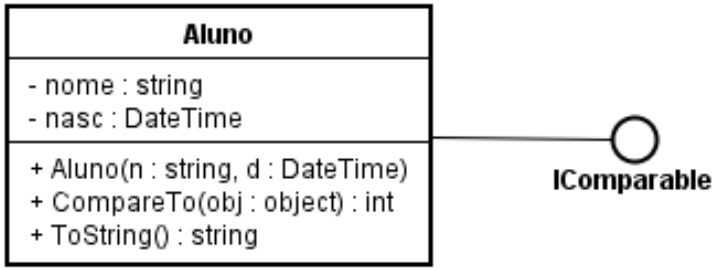
```
Aluno x = new Aluno("Pedro", DateTime.Parse("01/01/1992"));
IComparable y = new Aluno("Maria", DateTime.Parse("01/01/1993"));
object z = new Aluno("Paulo", DateTime.Parse("01/01/1991"));
Console.WriteLine(z.CompareTo(x)); // Erro
```

Interfaces e UML

- Realização: quando uma classe implementa uma interface
  - Representada por uma seta cheia e tracejada da classe para a interface



  - Representada na forma de “pirulito” (lollipop)



Interface IComparer

- Como ordenar utilizando outro critério de comparação?
- A interface IComparer do *framework* define o método Compare que também pode ser usado como padrão de comparação entre objetos

```
interface IComparer {
  int Compare(object x, object y);
}
```

- Compare permite definir vários critérios diferentes de comparação
- O método Sort da classe Array pode utilizar tanto IComparable quanto IComparer

Implementação de IComparer

- IComparer requer a escrita de uma classe que será responsável pela comparação dos objetos da classe alvo.
- Ordenação de Alunos pela data de nascimento

```
class AlunoNascComp : IComparer {
  public int Compare(object x, object y) {
    // Comparação entre os objetos x e y
  }
}
```

- Ordenando os alunos pela data de nascimento

```
Array.Sort(v, new AlunoNascComp());
```

Classes Aluno e AlunoNascComp

```
class Aluno : IComparable {
  private string nome;
  private DateTime nasc;
  public Aluno(string n, DateTime d) {
    this.nome = n;
    this.nasc = d;
  }
  public int CompareTo(object obj) {
    return nome.CompareTo(((Aluno)obj).nome);
  }
  public DateTime GetNascimento() {
    return nasc;
  }
  public override string ToString() {
    return $"{nome} - {nasc:dd/MM/yyyy}";
  }
}
```

```
class AlunoNascComp : IComparer {
  public int Compare(object x, object y) {
    Aluno a = (Aluno)x;
    Aluno b = (Aluno)y;
    return a.GetNascimento().CompareTo(
      b.GetNascimento());
  }
}
```

Array.Sort e Compare

- O método Sort da classe Array utiliza um objeto da classe AlunoNascComp para ordenar o vetor de alunos

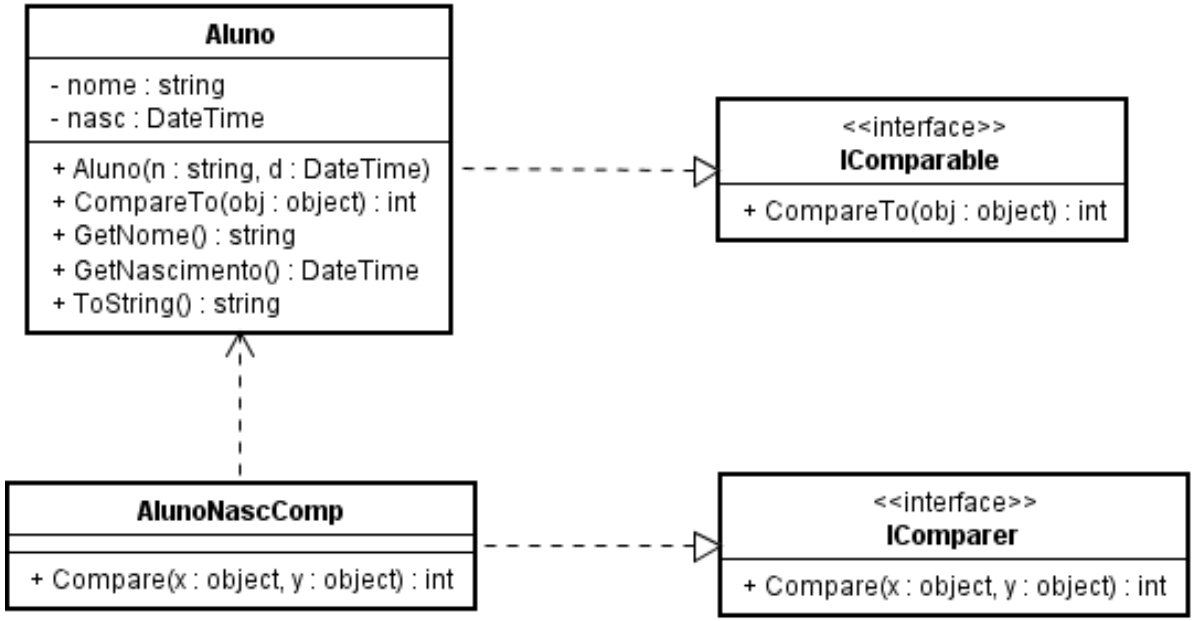
```
class MainClass {
  public static void Main (string[] args) {
    Aluno a = new Aluno("Pedro", DateTime.Parse("01/01/1992"));
    Aluno b = new Aluno("Maria", DateTime.Parse("01/01/1993"));
    Aluno c = new Aluno("Paulo", DateTime.Parse("01/01/1991"));
    Aluno[] v = { a, b, c };
    // Array.Sort(v);
    Array.Sort(v, new AlunoNascComp());
    foreach(Aluno i in v)
      Console.WriteLine(i);
  }
}
```

Saída anterior:
Maria - 01/01/1993
Paulo - 01/01/1991
Pedro - 01/01/1992

Saída atual:
Paulo - 01/01/1991
Pedro - 01/01/1992
Maria - 01/01/1993

Diagrama de Classes

- Diagrama de classes final



Referências

- Microsoft Visual C# 2010 – Passo a passo, John Sharp, Bookman, 2010
- UML – Uma Abordagem Prática, Gilleanes T. A. Guedes, Novatec, 2004
- Interfaces (Guia de Programação em C#)
  - https://docs.microsoft.com/pt-br/dotnet/csharp/programming-guide/interfaces/index