



Interativa

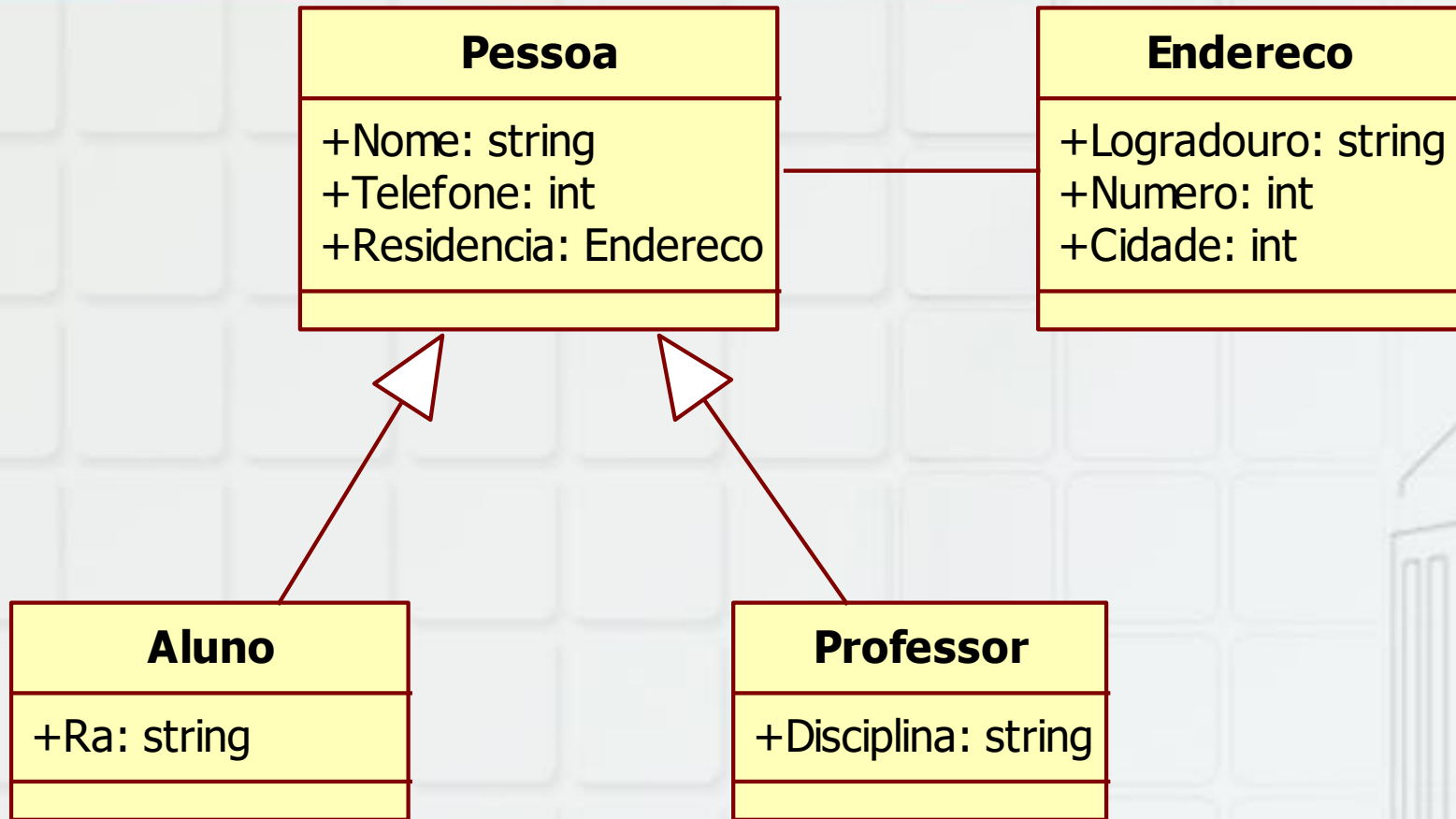
Unidade IV

PROGRAMAÇÃO ORIENTADA A OBJETOS I

Prof. Cassiano Gunji

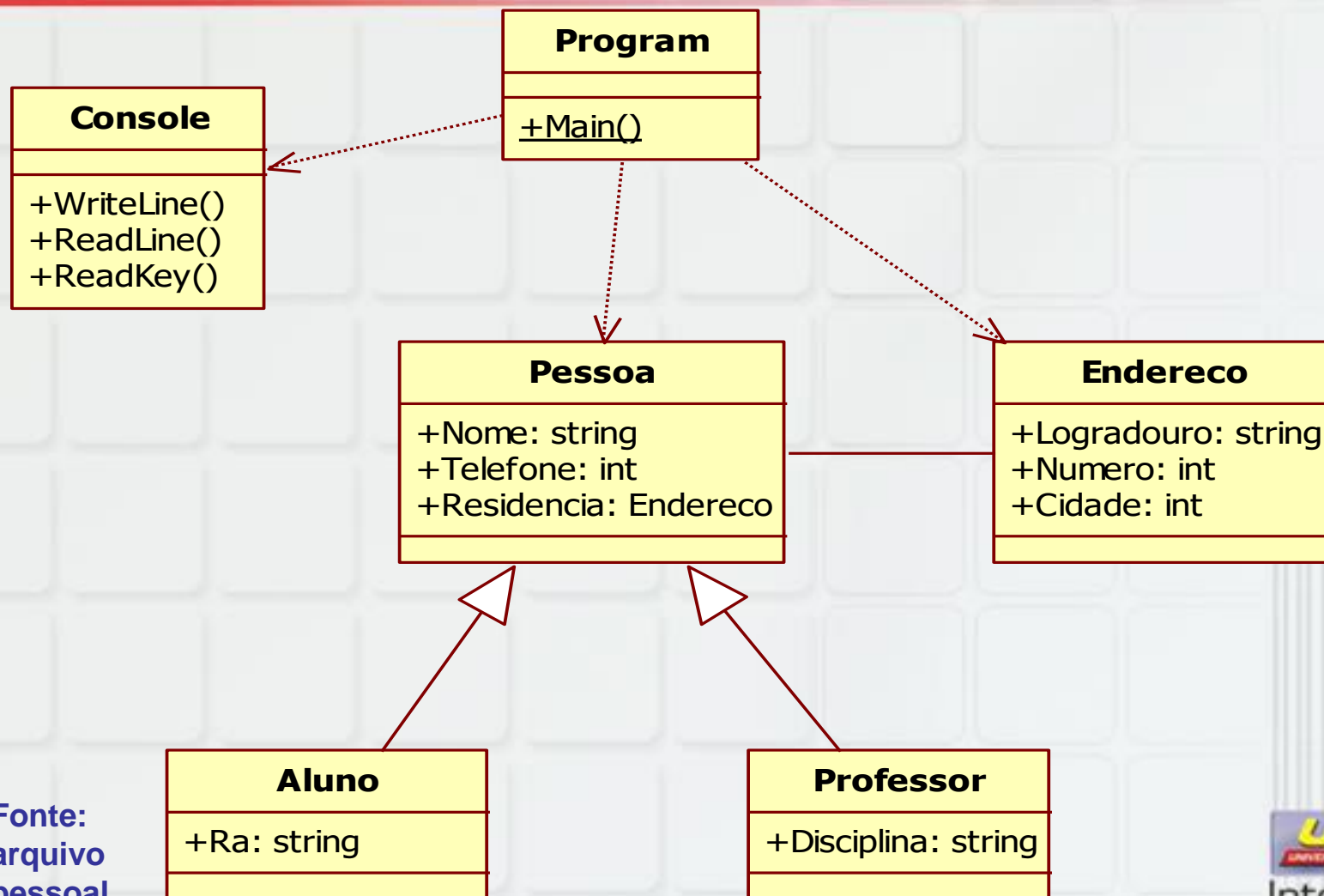


Relacionamentos entre classes – associação e generalização



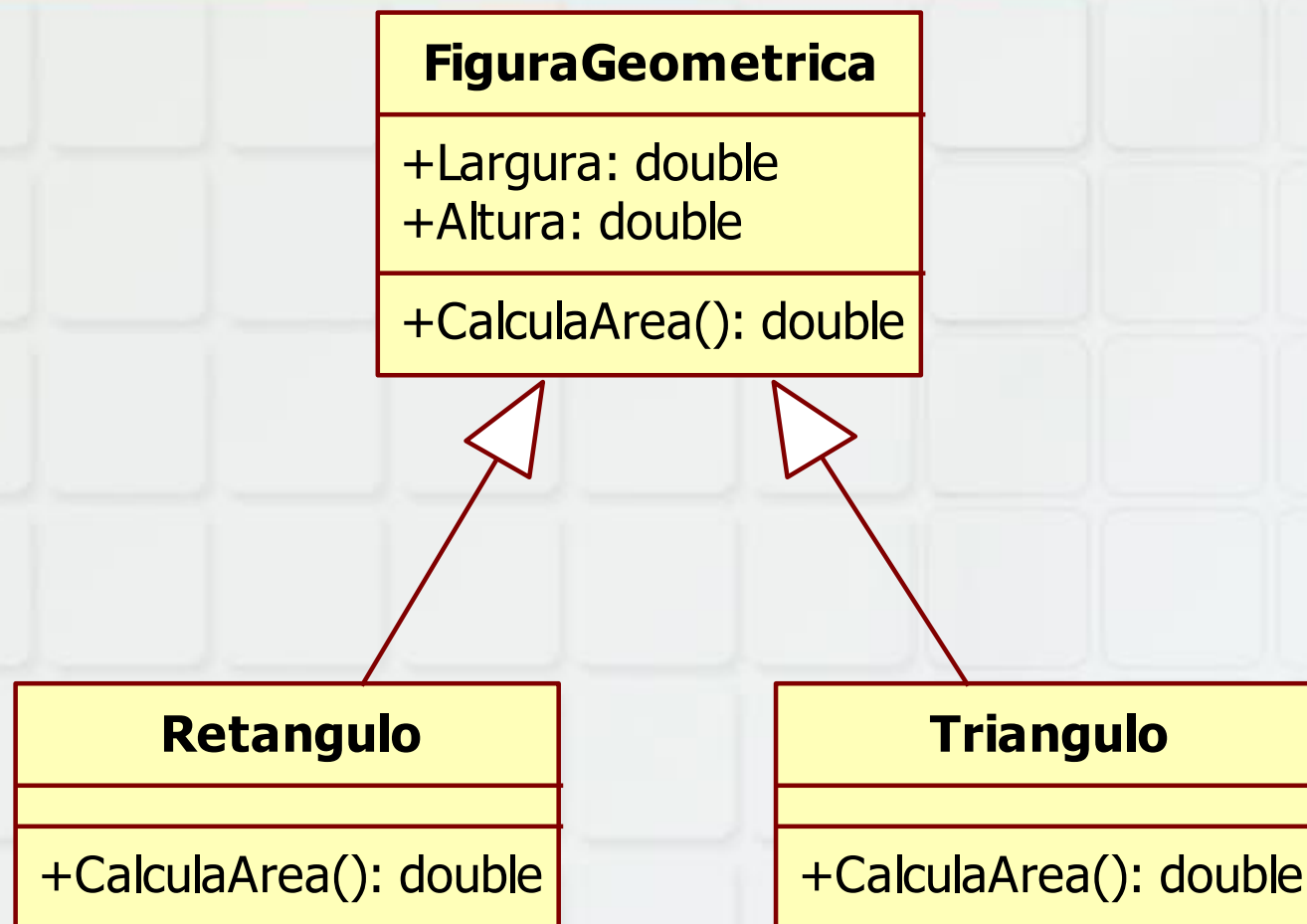
Fonte: arquivo pessoal.

Relacionamento entre classes – dependência



Fonte:
arquivo
pessoal.

Classes abstratas



Fonte: arquivo pessoal.

Classes abstratas

```
9  class FiguraGeometrica
10  {
11      public double Largura = 0;
12      public double Altura = 0;
13
14      -references
15      public virtual double CalculaArea()
16      {
17          return -1;
18      }
```

```
19  class Retangulo : FiguraGeometrica
20  {
21      -references
22      public override double CalculaArea()
23      {
24          return Largura * Altura;
25      }
26      -references
27      class Triangulo : FiguraGeometrica
28      {
29          -references
30          public override double CalculaArea()
31          {
32              return Largura * Altura / 2;
33          }
```

Fonte: arquivo pessoal.

Classes abstratas

```
33  class Program
34  {
    -references
35  static void Main(string[] args)
36  {
37      Retangulo r = new Retangulo();
38      r.Largura = 8.3;
39      r.Altura = 3.5;
40      Console.WriteLine("Área do retângulo: {0}", r.CalculaArea());
41
42      Triangulo t = new Triangulo();
43      t.Largura = 8.3;
44      t.Altura = 3.5;
45      Console.WriteLine("Área do triângulo: {0}", t.CalculaArea());
46
47      FiguraGeometrica fg = new FiguraGeometrica();
48      fg.Largura = 8.3;
49      fg.Altura = 3.5;
50      Console.WriteLine("Área da figura geométrica: {0}", fg.CalculaArea());
51
52      Console.ReadKey();
53  }
54  }
```

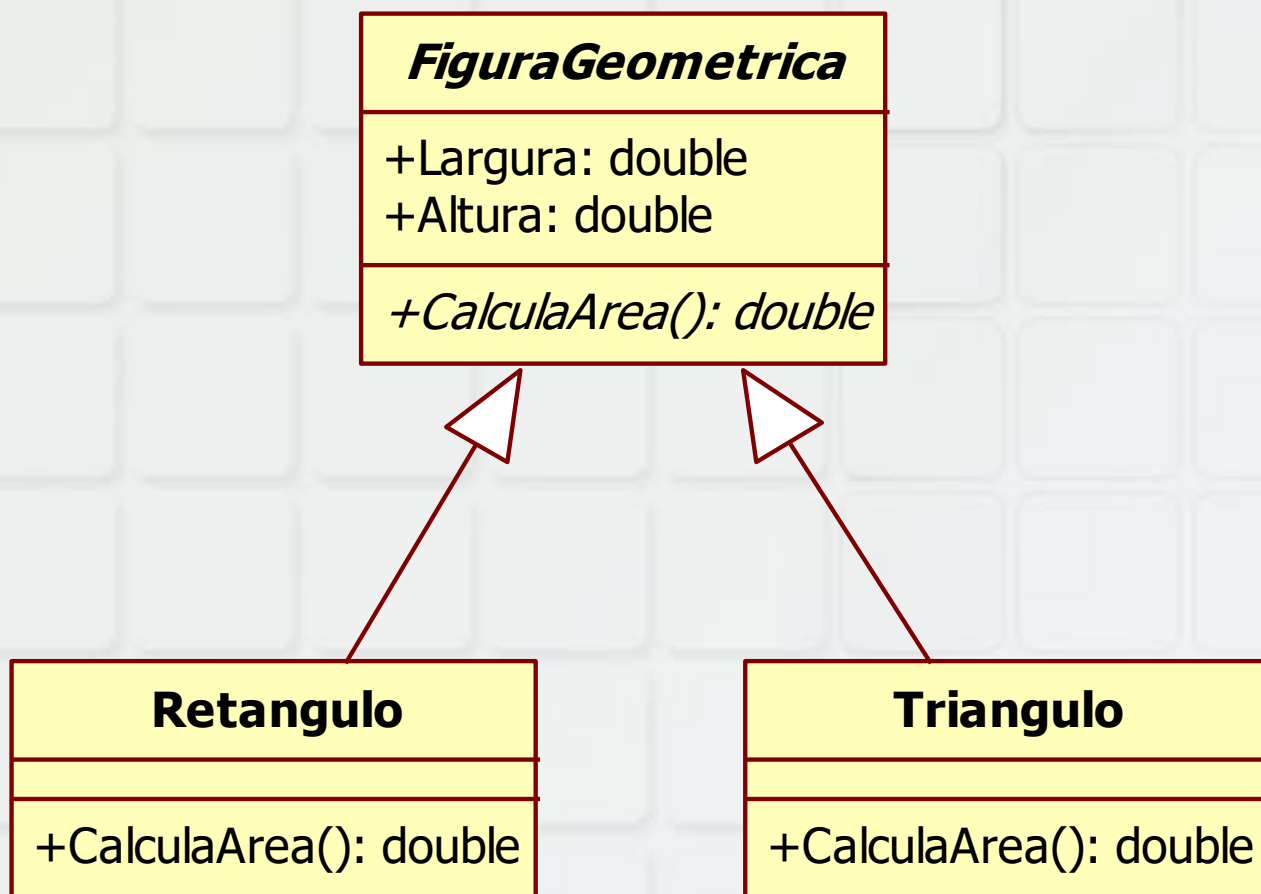
Classes abstratas

file:///C:/Users/Cassiano/Documents/Visual Stu...

```
Área do retângulo: 29,05  
Área do triângulo: 14,525  
Área da figura geométrica: -1
```

Fonte: arquivo pessoal.

Classes abstratas



Fonte: arquivo pessoal.

Classes abstratas

```
9  abstract class FiguraGeometrica
10 {
11     public double Largura = 0;
12     public double Altura = 0;
13
14     -references
15     public abstract double CalculaArea();
16 }
```

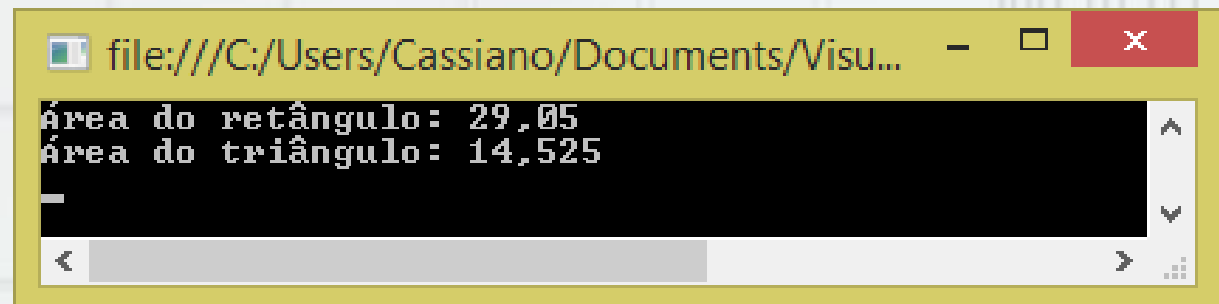
```
16 -references
17 class Retangulo : FiguraGeometrica
18 {
19     -references
20     public override double CalculaArea()
21     {
22         return Largura * Altura;
23     }
24
25     -references
26     class Triangulo : FiguraGeometrica
27     {
28         -references
29         public override double CalculaArea()
30         {
31             return Largura * Altura / 2;
32         }
33     }
34 }
```

Fonte: arquivo pessoal.

Classes abstratas

```
30 class Program
31 {
32     -references
33     static void Main(string[] args)
34     {
35         Retangulo r = new Retangulo();
36         r.Largura = 8.3;
37         r.Altura = 3.5;
38         Console.WriteLine("Área do retângulo: {0}", r.CalculaArea());
39
40         Triangulo t = new Triangulo();
41         t.Largura = 8.3;
42         t.Altura = 3.5;
43         Console.WriteLine("Área do triângulo: {0}", t.CalculaArea());
44
45         Console.ReadKey();
46     }
47 }
```

Fonte: arquivo pessoal.



The screenshot shows a Windows command prompt window with a yellow title bar. The title bar text is "file:///C:/Users/Cassiano/Documents/Visu...". The window contains the following text:

```
Área do retângulo: 29,05
Área do triângulo: 14,525
_
```

The text is displayed in a monospaced font on a black background. There are scrollbars on the right and bottom of the window.

Interatividade

Uma classe concreta (não abstrata) pode especializar uma classe abstrata ou uma classe concreta. Com isto em mente, assinale a alternativa correta.

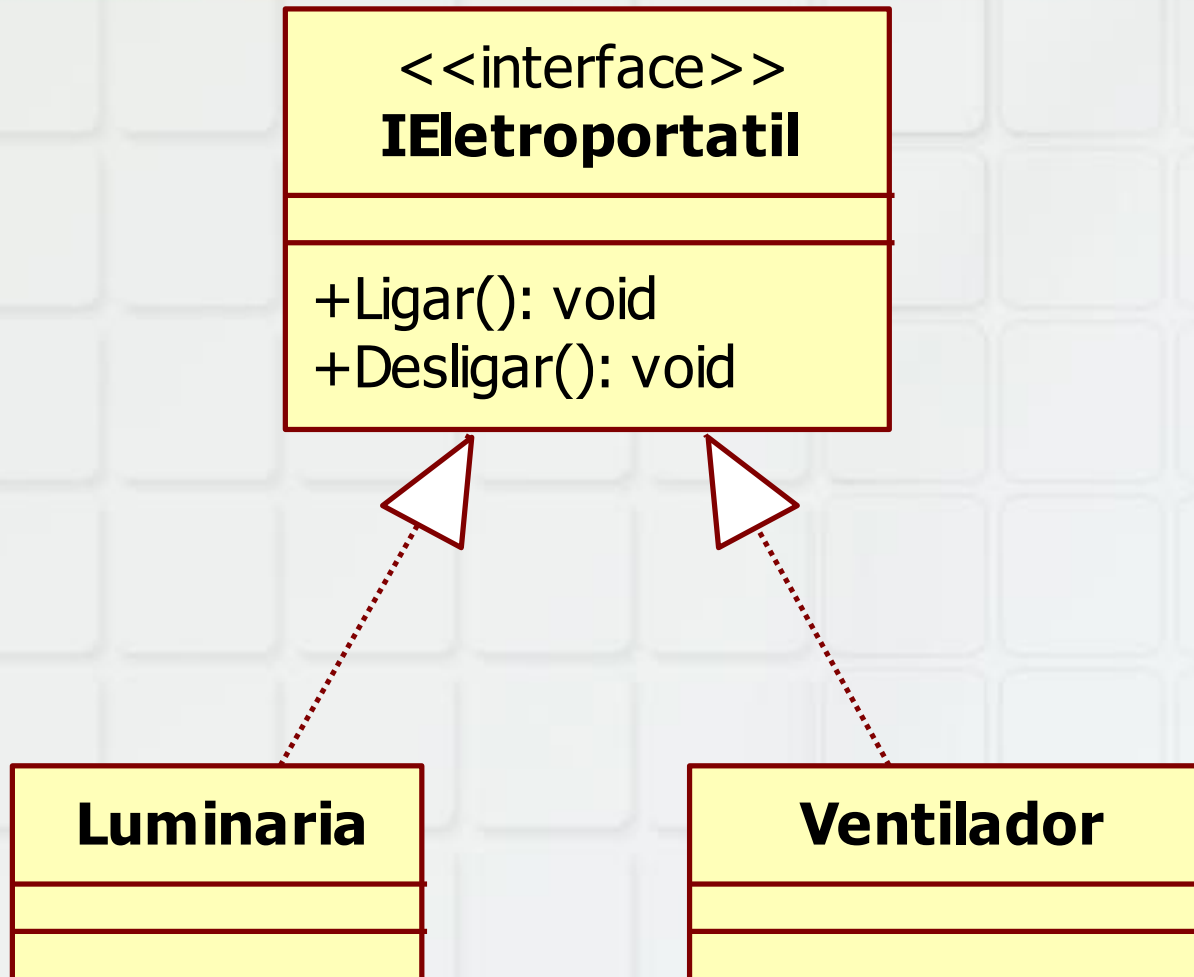
- a) Se a classe pai for abstrata, a classe filha também será.
- b) Se a classe pai define um método abstrato, a classe filha deverá implementar este método ou também será uma classe abstrata.
- c) Uma classe abstrata só pode declarar métodos abstratos.
- d) Se uma classe herda um método abstrato, ela não pode sobrescrever este método.
- e) Uma classe abstrata não pode ser subclasse de uma classe concreta.

Resposta

Uma classe concreta (não abstrata) pode especializar uma classe abstrata ou uma classe concreta. Com isto em mente, assinale a alternativa correta.

- a) Se a classe pai for abstrata, a classe filha também será.
- b) Se a classe pai define um método abstrato, a classe filha deverá implementar este método ou também será uma classe abstrata.**
- c) Uma classe abstrata só pode declarar métodos abstratos.
- d) Se uma classe herda um método abstrato, ela não pode sobrescrever este método.
- e) Uma classe abstrata não pode ser subclasse de uma classe concreta.

Interfaces



Fonte: arquivo pessoal.

Interfaces

2 references

```
interface IEletroportatil
```

```
{
```

4 references

```
void Ligar();
```

4 references

```
void Desligar();
```

```
}
```

2 references

```
class Luminaria : IEletroportatil
```

```
{
```

4 references

```
public void Ligar()
```

```
{
```

```
    Console.WriteLine("A luminária foi ligada.");
```

```
}
```

4 references

```
public void Desligar()
```

```
{
```

```
    Console.WriteLine("A luminária foi desligada");
```

```
}
```

```
}
```

Fonte: arquivo pessoal.



Interativa

Interfaces

2 references

```
25 class Ventilador : IEletoportatil
```

```
26 {
```

4 references

```
27     public void Ligar()
```

```
28     {
```

```
29         Console.WriteLine("O ventilador foi ligado.");
```

```
30     }
```

4 references

```
31     public void Desligar()
```

```
32     {
```

```
33         Console.WriteLine("O ventilador foi desligado");
```

```
34     }
```

```
35 }
```

Interfaces

0 references

```
class Program
```

```
{
```

0 references

```
static void Main(string[] args)
```

```
{
```

```
    Luminaria l = new Luminaria();
```

```
    Ventilador v = new Ventilador();
```

```
    l.Ligar();
```

```
    v.Ligar();
```

```
    l.Desligar();
```

```
    v.Desligar();
```

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

```
}
```

```
}
```

file:///C:/Users/Cassiano/Docum...

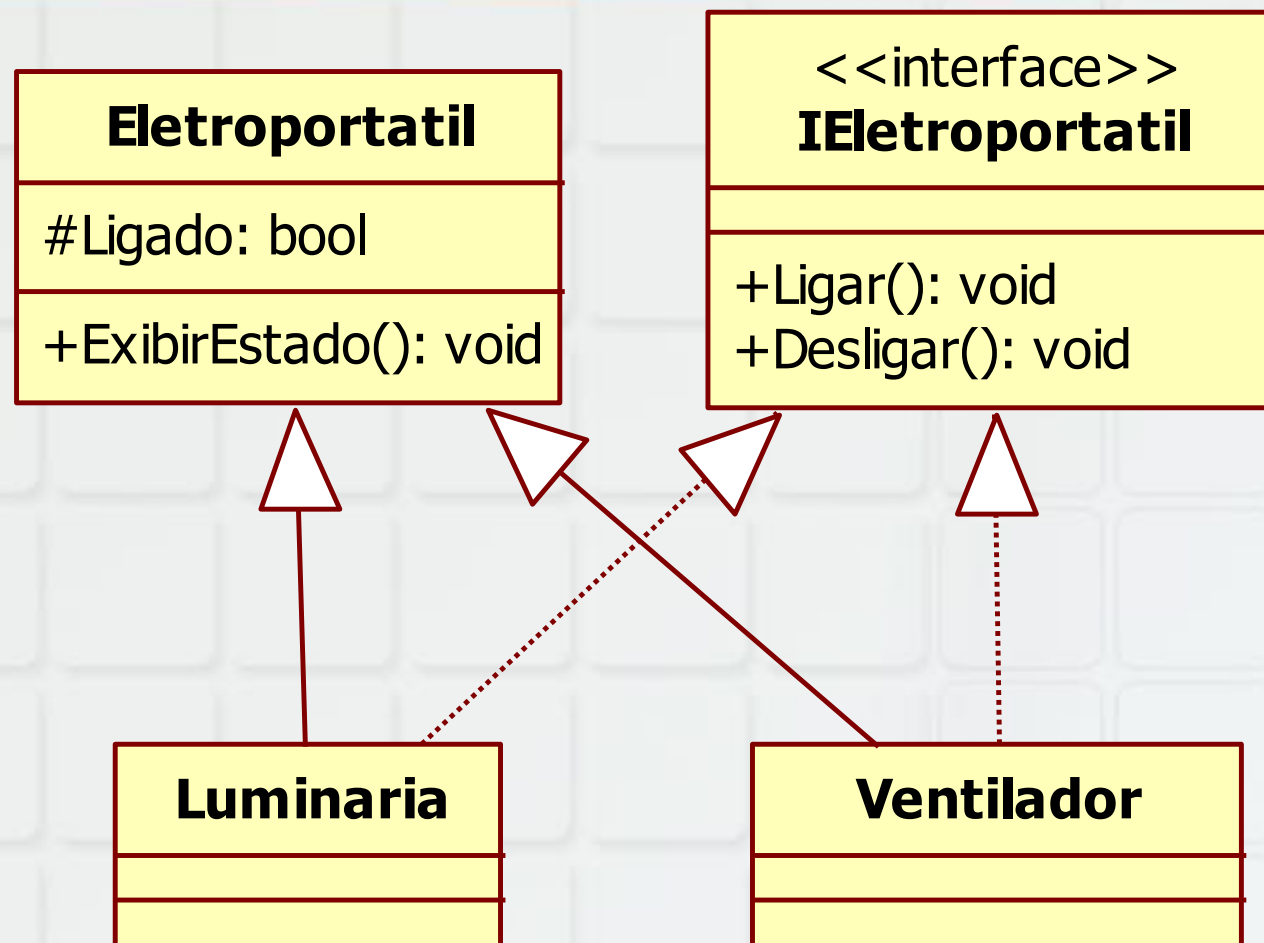
```
A luminária foi ligada.  
O ventilador foi ligado.  
A luminária foi desligada  
O ventilador foi desligado
```

Fonte: arquivo pessoal.



UNIP
UNIVERSIDADE PAULISTA
Interativa

Combinando herança e realização



Fonte: arquivo pessoal.

Combinando herança e realização

2 references

```
class Eletroportatil
```

```
{
```

```
    protected bool ligado;
```

2 references

```
    public void ExibirEstado()
```

```
{
```

```
    if (ligado)
```

```
{
```

```
        Console.WriteLine("O aparelho está ligado");
```

```
}
```

```
    else
```

```
{
```

```
        Console.WriteLine("O aparelho está desligado");
```

```
}
```

```
}
```

```
}
```

Combinando herança e realização

2 references

```
29 class Luminaria : Eletroportatil, IEletroportatil  
30 {
```

3 references

```
31 public void Ligar()  
32 {
```

```
33     ligado = true;
```

```
34     Console.WriteLine("A luminária foi ligada.");
```

```
35 }
```

2 references

```
36 public void Desligar()  
37 {
```

```
38     ligado = false;
```

```
39     Console.WriteLine("A luminária foi desligada");
```

```
40 }
```

```
41 }
```

Fonte: arquivo pessoal.

Combinando herança e realização

```
55  class Program
56  {
    // references
57  static void Main(string[] args)
58  {
59      Luminaria l = new Luminaria();
60      l.ExibirEstado();
61      l.Ligar();
62      l.ExibirEstado();
63
64      Console.ReadKey();
65  }
66 }
```

file:///C:/Users/Cassiano/Documen...

```
0 aparelho está desligado
A luminária foi ligada.
0 aparelho está ligado
```

Interatividade

Sobre a especialização de classes e a realização de interfaces, assinale a alternativa incorreta.

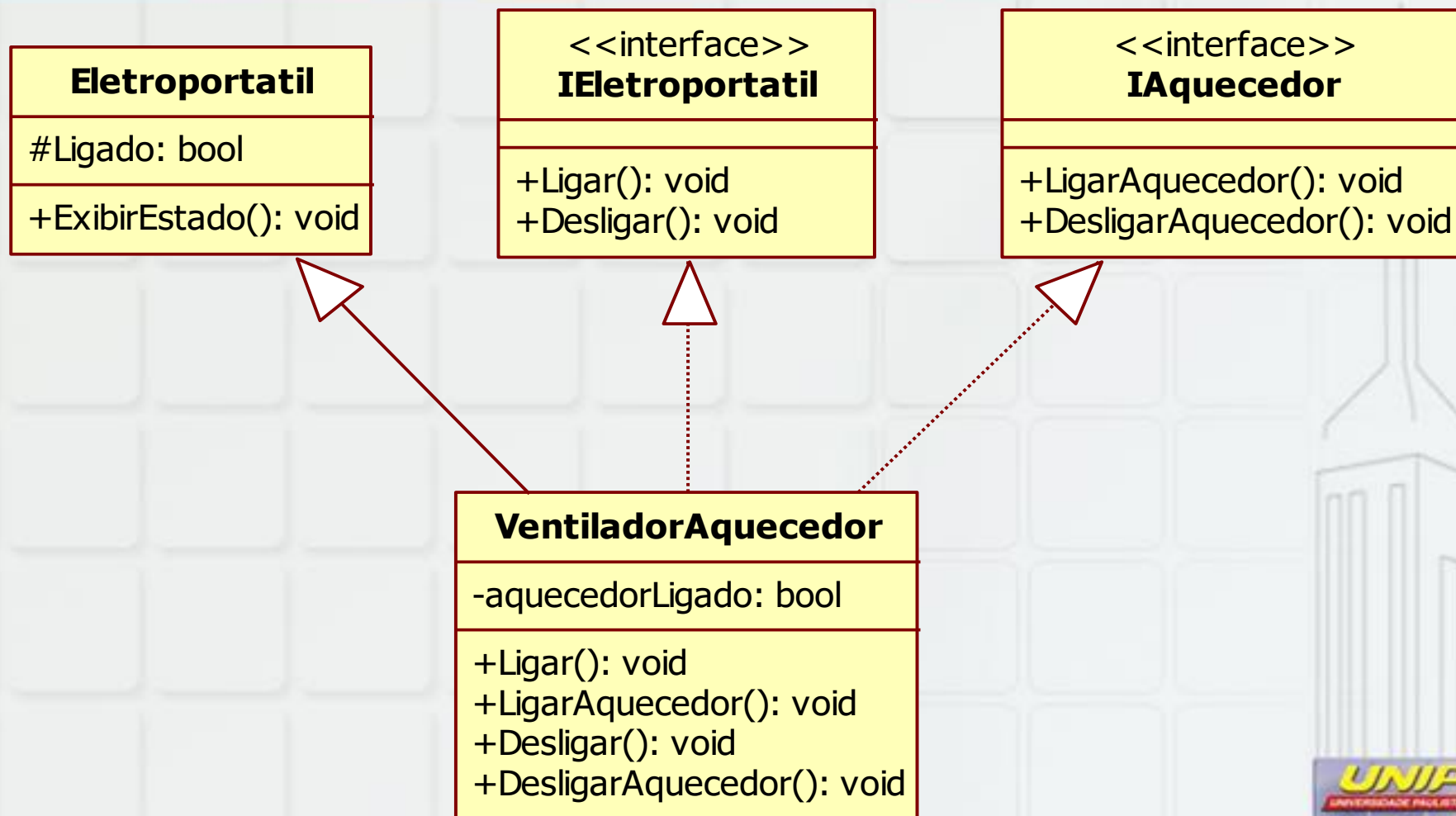
- a) Classes abstratas podem declarar atributos estáticos e não estáticos.**
- b) Interfaces só podem declarar atributos estáticos e constantes.**
- c) Ao se sobrescrever um método abstrato, deve-se usar a palavra-chave “override”.**
- d) Ao se implementar um método definido em uma interface não é necessário utilizar nenhuma palavra-chave.**
- e) Uma classe abstrata não precisa implementar métodos abstratos de uma classe pai.**

Resposta

Sobre a especialização de classes e a realização de interfaces, assinale a alternativa incorreta.

- a) Classes abstratas podem declarar atributos estáticos e não estáticos.
- b) Interfaces só podem declarar atributos estáticos e constantes.**
- c) Ao se sobrescrever um método abstrato, deve-se usar a palavra-chave “override”.
- d) Ao se implementar um método definido em uma interface não é necessário utilizar nenhuma palavra-chave.
- e) Uma classe abstrata não precisa implementar métodos abstratos de uma classe pai.

Realizando mais de uma interface



Realizando mais de uma interface

```
9  interface IEletoportatil
10 {
11     4 references
12     void Ligar();
13     3 references
14     void Desligar();
15 }
16 1 reference
17 interface IAquecedor
18 {
19     2 references
20     void LigarAquecedor();
21     1 reference
22     void DesligarAquecedor();
23 }
```



Realizando mais de uma interface

```
60 class VentiladorAquecedor: Eletroportatil, IEletroportatil, IAquecedor
61 {
62     private bool aquecedorLigado;
63     public void Ligar()...
68     public void Desligar()...
73     public void LigarAquecedor()
74     {
75         aquecedorLigado = true;
76         Console.WriteLine("O ar quente foi ligado.");
77     }
78     public void DesligarAquecedor()
79     {
80         aquecedorLigado = false;
81         Console.WriteLine("O ar quente foi desligado");
82     }
83 }
```

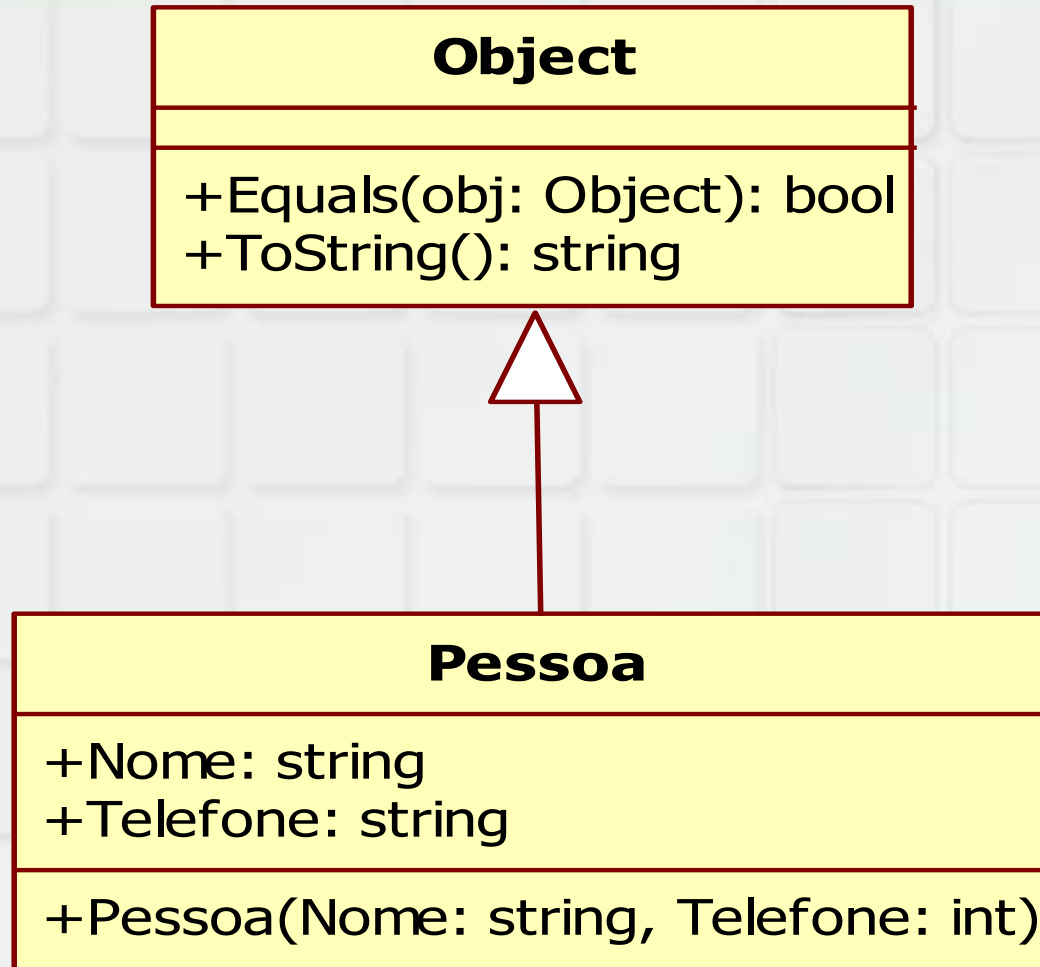
Realizando mais de uma interface

```
84 class Program
85 {
    0 references
86 static void Main(string[] args)
87 {
88     VentiladorAquecedor va = new VentiladorAquecedor();
89     va.ExibirEstado();
90     va.Ligar();
91     va.LigarAquecedor();
92     va.ExibirEstado();
93
94     Console.ReadKey();
95 }
96 }
```

file:///C:/Users/Cassiano/Docu...

```
0 aparelho está desligado
0 ventilador foi ligado.
0 ar quente foi ligado.
0 aparelho está ligado
```

A classe Object



Fonte: arquivo pessoal.

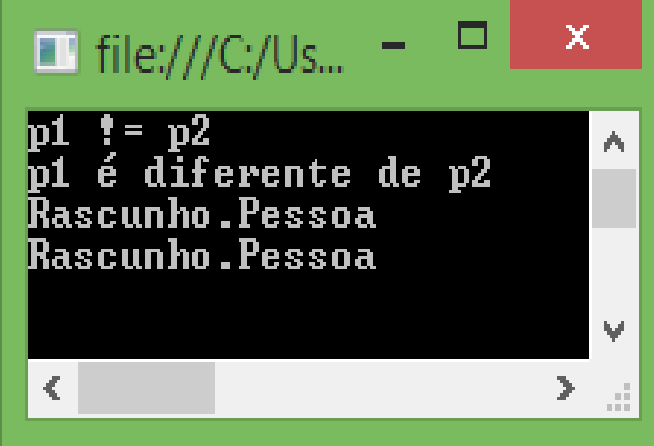
A classe Object

```
9  class Pessoa
10 {
11     public string Nome;
12     public int Telefone;
13
14     2 references
15     public Pessoa(string Nome, int Telefone)
16     {
17         this.Nome = Nome;
18         this.Telefone = Telefone;
19     }
19 }
```

Fonte: arquivo pessoal.

A classe Object

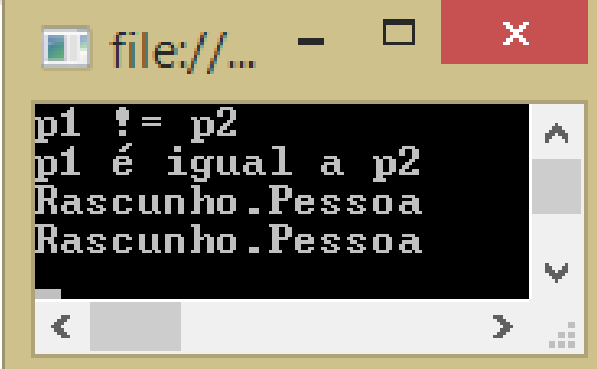
```
20 class Program
21 {
22     // references
23     static void Main(string[] args)
24     {
25         Pessoa p1 = new Pessoa("João", 123);
26         Pessoa p2 = new Pessoa("João", 123);
27
28         if (p1 == p2)
29             Console.WriteLine("p1 == p2");
30         else
31             Console.WriteLine("p1 != p2");
32
33         if (p1.Equals(p2))
34             Console.WriteLine("p1 é igual a p2");
35         else
36             Console.WriteLine("p1 é diferente de p2");
37
38         Console.WriteLine(p1);
39         Console.WriteLine(p2);
40         Console.ReadKey();
41     }
}
```



```
p1 != p2
p1 é diferente de p2
Rascunho.Pessoa
Rascunho.Pessoa
```

A classe Object

```
9  class Pessoa
10 {
11     public string Nome;
12     public int Telefone;
13
14     2 references
15     public Pessoa(string Nome, int Telefone) ...
16     1 reference
17     public override bool Equals(object obj)
18     {
19         bool igual = false;
20         Pessoa pessoa;
21         if (obj is Pessoa)
22         {
23             pessoa = (Pessoa)obj;
24             if (this.Nome == pessoa.Nome && this.Telefone == pessoa.Telefone)
25                 igual = true;
26         }
27         return igual;
28     }
29 }
30
31 }
```



```
p1 != p2
p1 é igual a p2
Rascunho.Pessoa
Rascunho.Pessoa
```

Fonte: arquivo pessoal.

A classe Object

```
9  class Pessoa
10  {
11      public string Nome;
12      public int Telefone;
13
14      2 references
15      public Pessoa(string Nome, int Telefone) ...
16      1 reference
17      public override bool Equals(object obj) ...
18      0 references
19      public override string ToString()
20      {
21          return String.Format("Nome: {0}, Telefone: {1}", Nome, Telefone);
22      }
23  }
```

file:///C:/Users/Cassiano/...

```
p1 != p2
p1 é igual a p2
Nome: João, Telefone: 123
Nome: João, Telefone: 123
```

Fonte: arquivo pessoal.

Interatividade

Sobre a especialização de classes e a realização de interfaces, assinale a alternativa incorreta.

- a) Uma classe pode especializar uma classe concreta ou uma classe abstrata.**
- b) Uma classe pode especializar uma classe concreta e realizar uma interface.**
- c) Uma classe pode especializar uma classe abstrata e realizar três interfaces.**
- d) Uma classe pode realizar qualquer número de interfaces.**
- e) Uma classe pode especializar uma classe concreta e uma classe abstrata.**

Resposta

Sobre a especialização de classes e a realização de interfaces, assinale a alternativa incorreta.

- a) Uma classe pode especializar uma classe concreta ou uma classe abstrata.
- b) Uma classe pode especializar uma classe concreta e realizar uma interface.
- c) Uma classe pode especializar uma classe abstrata e realizar três interfaces.
- d) Uma classe pode realizar qualquer número de interfaces.
- e) **Uma classe pode especializar uma classe concreta e uma classe abstrata.**

Tratamento de exceções

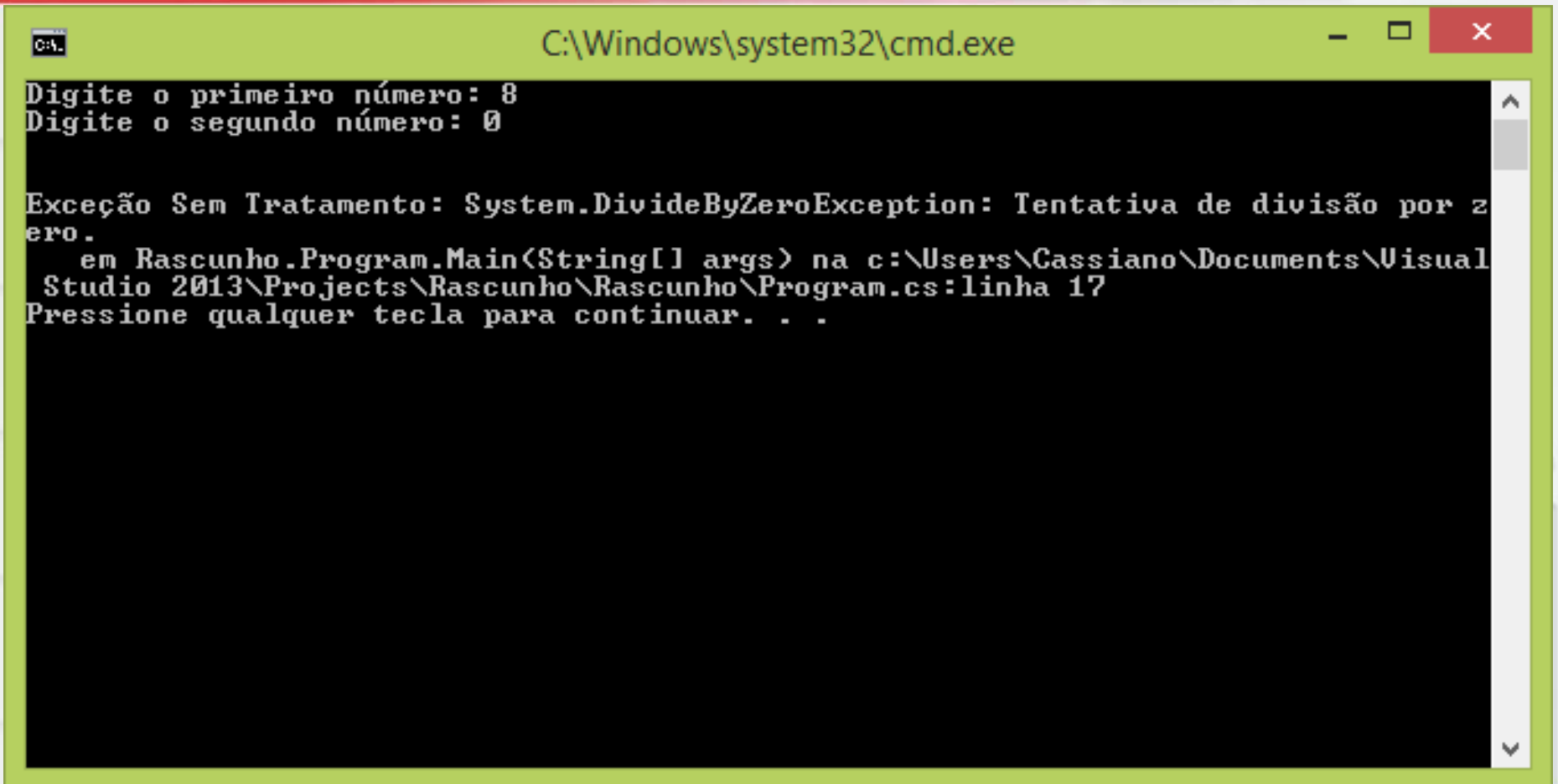
```
9  class Program
10 {
    0 references
11     static void Main(string[] args)
12     {
13         int n1 = LeiaNumero("Digite o primeiro número: ");
14         int n2 = LeiaNumero("Digite o segundo número: ");
15
16         Console.WriteLine();
17         Console.WriteLine("{0} / {1} = {2}", n1, n2, n1 / n2);
18         Console.ReadKey();
19     }
    2 references
20     static int LeiaNumero(string mensagem)
21     {
22         int numero;
23
24         Console.Write(mensagem);
25         numero = Convert.ToInt32(Console.ReadLine());
26
27         return numero;
28     }
29 }
```

Fonte: arquivo pessoal.



Interativa

Tratamento de exceções



A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window has a green title bar and standard Windows window controls (minimize, maximize, close). The command prompt shows the following text:

```
C:\> Digite o primeiro número: 8
C:\> Digite o segundo número: 0

Exceção Sem Tratamento: System.DivideByZeroException: Tentativa de divisão por zero.
    em Rascunho.Program.Main(String[] args) na c:\Users\Cassiano\Documents\Visual Studio 2013\Projects\Rascunho\Rascunho\Program.cs:linha 17
Pressione qualquer tecla para continuar. . .
```

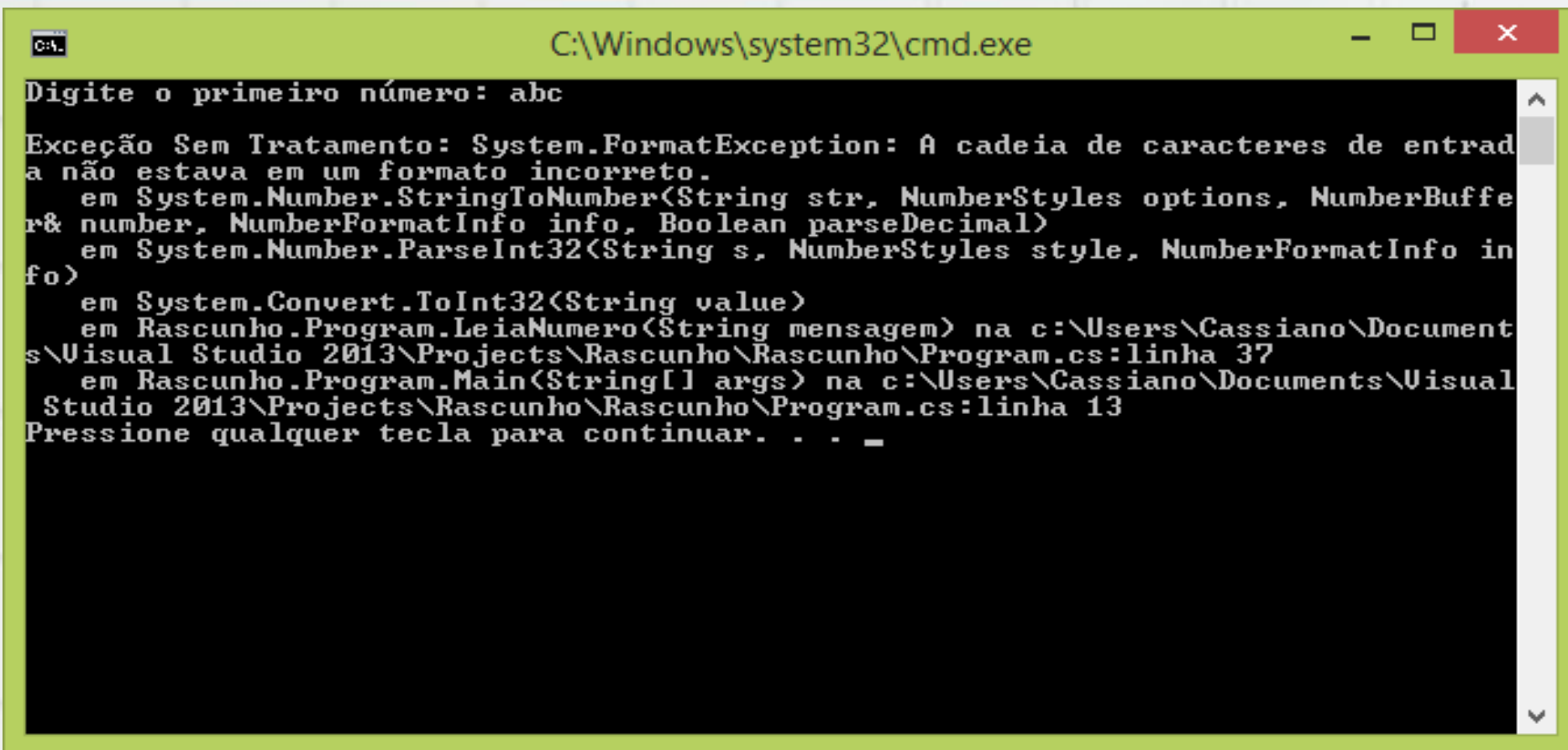
Fonte: arquivo pessoal.

Tratamento de exceções

```
9  class Program
10 {
    0 references
11     static void Main(string[] args)
12     {
13         int n1 = LeiaNumero("Digite o primeiro número: ");
14         int n2 = LeiaNumero("Digite o segundo número: ");
15
16         Console.WriteLine();
17         try
18         {
19             Console.WriteLine("{0} / {1} = {2}", n1, n2, n1 / n2);
20         }
21         catch (DivideByZeroException)
22         {
23             Console.WriteLine("Não é possível fazer uma divisão por zero.");
24         }
25         Console.ReadKey();
26     }
```

Fonte: arquivo pessoal.

Tratamento de exceções



The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The prompt asks to enter a number, and the user enters "abc". This triggers an unhandled exception. The error message is displayed in Portuguese, indicating a "System.FormatException" because the input string "abc" is not in a valid numeric format. The stack trace shows the exception was thrown in the "StringToNumber" method, then in "ParseInt32", and finally in the "Convert.ToInt32" method. The application is identified as "Rascunho" (Draft) running in Visual Studio 2013. The user is prompted to press any key to continue.

```
C:\Windows\system32\cmd.exe
Digite o primeiro número: abc
Exceção Sem Tratamento: System.FormatException: A cadeia de caracteres de entrada não estava em um formato incorreto.
   em System.Number.StringToNumber(String str, NumberStyles options, NumberBuffer& number, NumberFormatInfo info, Boolean parseDecimal)
   em System.Number.ParseInt32(String s, NumberStyles style, NumberFormatInfo info)
   em System.Convert.ToInt32(String value)
   em Rascunho.Program.LeiaNumero(String mensagem) na c:\Users\Cassiano\Documents\Visual Studio 2013\Projects\Rascunho\Rascunho\Program.cs:linha 37
   em Rascunho.Program.Main(String[] args) na c:\Users\Cassiano\Documents\Visual Studio 2013\Projects\Rascunho\Rascunho\Program.cs:linha 13
Pressione qualquer tecla para continuar. . . _
```

Tratamento de exceções

```
27 static int LeiaNumero(string mensagem)
28 {
29     int numero = 0;
30     bool correto = false;
31
32     do
33     {
34         try
35         {
36             Console.Write(mensagem);
37             numero = Convert.ToInt32(Console.ReadLine());
38             correto = true;
39         }
40         catch (FormatException)
41         {
42             Console.WriteLine("Você não digitou um número inteiro. Tente de novo.");
43         }
44     } while (!correto);
45
46     return numero;
47 }
```

Tratamento de exceções

file:///C:/Users/Cassiano/Documents/Visual Studio 2013/Projects/Rascunho/Ras...

```
Digite o primeiro número: a
Você não digitou um número inteiro. Tente de novo.
Digite o primeiro número: b
Você não digitou um número inteiro. Tente de novo.
Digite o primeiro número: c
Você não digitou um número inteiro. Tente de novo.
Digite o primeiro número: d
Você não digitou um número inteiro. Tente de novo.
Digite o primeiro número: e
Você não digitou um número inteiro. Tente de novo.
Digite o primeiro número: f
Você não digitou um número inteiro. Tente de novo.
Digite o primeiro número: g
Você não digitou um número inteiro. Tente de novo.
Digite o primeiro número: 8
Digite o segundo número: 2
```

```
8 / 2 = 4
```

Interatividade

Qual das alternativas abaixo não é uma característica do tratamento de exceções?

- a) O tratamento de exceções aumenta a estabilidade do sistema.
- b) O tratamento de exceções deve ser feito sempre que há a possibilidade de ocorrência de um problema com a execução do programa.
- c) O tratamento de uma exceção deve sempre permitir que o programa se recupere do problema que causou a exceção.
- d) Um programa que é interrompido por uma exceção não tratada é um programa de baixa qualidade.
- e) O tratamento de todas as exceções pode tornar a manutenção do programa mais difícil.

Resposta

Qual das alternativas abaixo não é uma característica do tratamento de exceções?

- a) O tratamento de exceções aumenta a estabilidade do sistema.
- b) O tratamento de exceções deve ser feito sempre que há a possibilidade de ocorrência de um problema com a execução do programa.
- c) O tratamento de uma exceção deve sempre permitir que o programa se recupere do problema que causou a exceção.
- d) Um programa que é interrompido por uma exceção não tratada é um programa de baixa qualidade.
- e) O tratamento de todas as exceções pode tornar a manutenção do programa mais difícil.

ATÉ A PRÓXIMA!



Interativa