



Interativa

Unidade III

PROGRAMAÇÃO ORIENTADA A OBJETOS I

Prof. Cassiano Gunji



Abstração de classes

Pessoa

+Nome: string
+Email: string
+Telefone: int

```
class Pessoa
{
    public string Nome;
    public string Email;
    public int Telefone;
}
```

Fonte: arquivo pessoal.

Instanciando uma classe em objetos

```
class Program
{
    -references
    public static void Main(string[] args)
    {
        Pessoa p1 = new Pessoa();
        p1.Nome = "Joaquim";
        p1.Email = "joaquim@provedor.com";
        p1.Telefone = 1234565789;

        Pessoa p2 = new Pessoa();
        p2.Nome = "Manoel";
        p2.Email = "manoel@provedor.com";
        p2.Telefone = 987654321;
    }
}
```

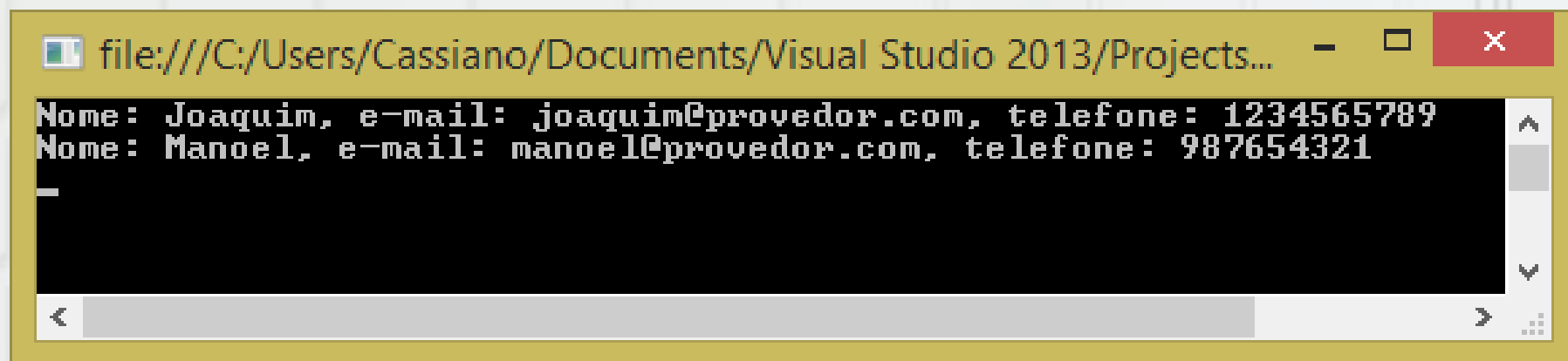
<u>p1 : Pessoa</u>
Nome = "Joaquim"
Email = "joaquim@provedor.com"
Telefone = 123456789

<u>p2 : Pessoa</u>
Nome = "Manoel"
Email = "manoel@provedor.com"
Telefone = 987654321

Instanciando uma classe em objetos

```
Console.WriteLine("Nome: {0}, e-mail: {1}, telefone: {2}",  
    p1.Nome, p1.Email, p1.Telefone);  
Console.WriteLine("Nome: {0}, e-mail: {1}, telefone: {2}",  
    p2.Nome, p2.Email, p2.Telefone);
```

Fonte: Arquivo pessoal



```
file:///C:/Users/Cassiano/Documents/Visual Studio 2013/Projects...  
Nome: Joaquim, e-mail: joaquim@provedor.com, telefone: 1234565789  
Nome: Manoel, e-mail: manoel@provedor.com, telefone: 987654321
```

Fonte: arquivo pessoal.

Referência a objetos

```
public static void Main(string[] args)
{
    Pessoa p1 = new Pessoa();
    p1.Nome = "Joaquim";
    p1.Email = "joaquim@provedor.com";
    p1.Telefone = 1234565789;

    Pessoa p2 = p1;
    p2.Nome = "Manoel";
    p2.Email = "manoel@provedor.com";
    p2.Telefone = 987654321;

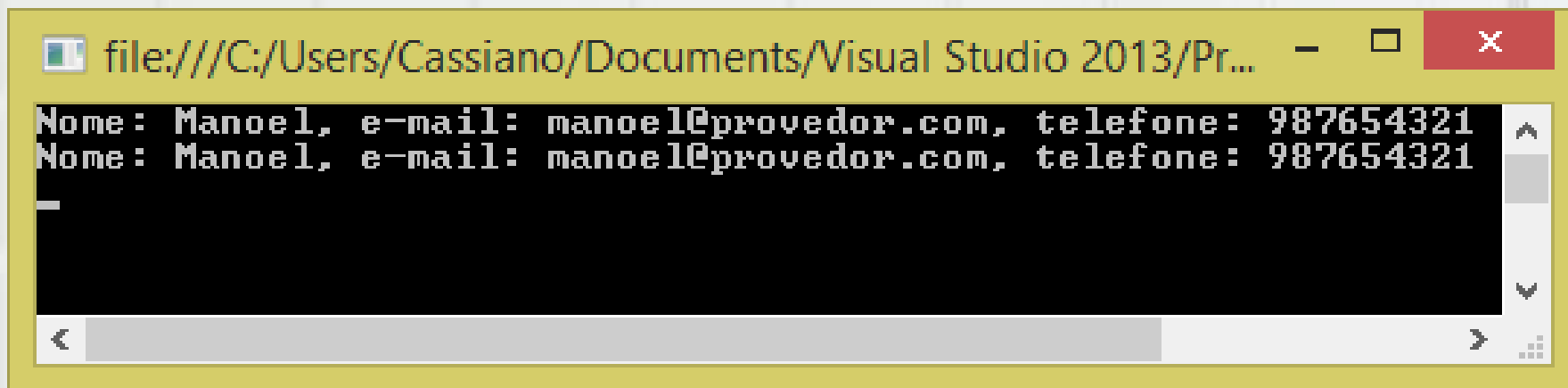
    Console.WriteLine("Nome: {0}, e-mail: {1}, telefone: {2}",
        p1.Nome, p1.Email, p1.Telefone);
    Console.WriteLine("Nome: {0}, e-mail: {1}, telefone: {2}",
        p2.Nome, p2.Email, p2.Telefone);

    Console.ReadKey();
}
```

p1 : Pessoa

Nome = "Joaquim"
Email = "joaquim@provedor.com"
Telefone = 123456789

Referência a objetos



file:///C:/Users/Cassiano/Documents/Visual Studio 2013/Pr... - □ ×

```
Nome: Manoel, e-mail: manoel@provedor.com, telefone: 987654321
Nome: Manoel, e-mail: manoel@provedor.com, telefone: 987654321
```

Fonte: arquivo pessoal.

Métodos

Retangulo

+Largura: double

+Altura: double

+CalculaArea(): double

+AumentarLargura(l: double): void

+AumentarAltura(a: double): void

Fonte: arquivo pessoal.

Métodos

```
class Retangulo
{
    public double Largura = 0;
    public double Altura = 0;

    -references
    public double CalculaArea()
    {
        return Largura * Altura;
    }
    -references
    public void AumentaLargura(double l)
    {
        Largura += l;
    }
    -references
    public void AumentaAltura(double a)
    {
        Altura += a;
    }
}
```

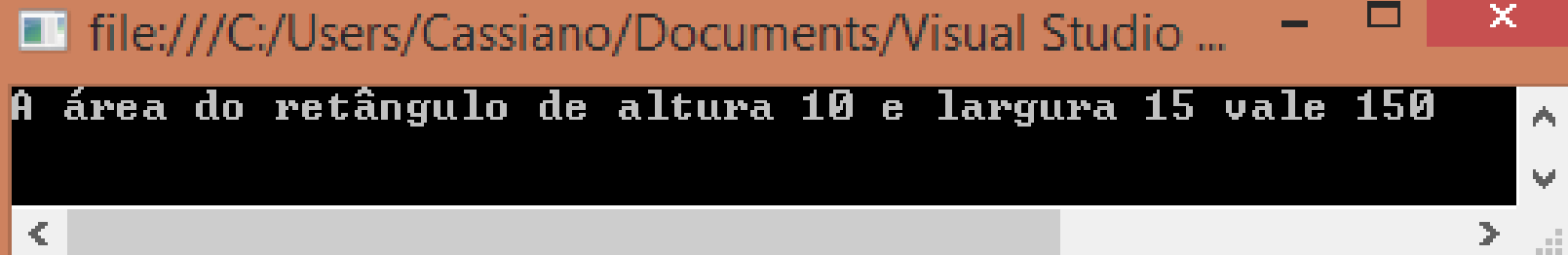


Métodos

```
class Program
{
    -references
    public static void Main(string[] args)
    {
        Retangulo r = new Retangulo();
        r.AumentaAltura(10);
        r.AumentaLargura(15);
        double area = r.CalculaArea();

        Console.WriteLine("A área do retângulo de altura {0} e largura {1} vale {2}",
            r.Altura, r.Largura, area);
        Console.ReadKey();
    }
}
```

Fonte: Arquivo pessoal



The screenshot shows a console window titled "file:///C:/Users/Cassiano/Documents/Visual Studio ...". The output text is "A área do retângulo de altura 10 e largura 15 vale 150". The window has a standard Windows title bar with minimize, maximize, and close buttons. The console text is in a monospaced font on a black background.

Fonte: arquivo pessoal.

Interatividade

Sobre os atributos e métodos de uma classe, podemos afirmar que:

- a) atributos não apresentam parêntesis;**
- b) métodos só apresentam parêntesis se possuírem parâmetros;**
- c) atributos devem possuir uma implementação declarada na classe que será executada no objeto;**
- d) métodos sem parâmetros comportam-se como atributos;**
- e) métodos com retorno void devem possuir o comando return em sua implementação.**

Resposta

Sobre os atributos e métodos de uma classe, podemos afirmar que:

- a) atributos não apresentam parêntesis;**
- b) métodos só apresentam parêntesis se possuírem parâmetros;**
- c) atributos devem possuir uma implementação declarada na classe que será executada no objeto;**
- d) métodos sem parâmetros comportam-se como atributos;**
- e) métodos com retorno void devem possuir o comando return em sua implementação.**

Método construtor

- É um método que é executado no momento em que o objeto está sendo instanciado.
- Deve ter exatamente o mesmo nome da classe, incluindo as letras maiúsculas e minúsculas.
- Não deve ter tipo de retorno, nem mesmo void.



Método construtor

```
class Data
{
    public int Dia;
    public int Mes;
    public int Ano;

    -references
    public Data(int dia, int mes, int ano)
    {
        Dia = dia;
        Mes = mes;
        Ano = ano;
    }
}
```



Método construtor

```
class Program
{
    -references
    public static void Main(string[] args)
    {
        Data d1 = new Data(25, 12, 2015);
    }
}
```

Fonte: arquivo pessoal.

Referência this

```
class Data
{
    public int Dia;
    public int Mes;
    public int Ano;

    -references
    public Data(int Dia, int Mes, int Ano)
    {
        Dia = Dia;
        Mes = Mes;
        Ano = Ano;
    }
}
```

Referência this

```
class Data
{
    public int Dia;
    public int Mes;
    public int Ano;

    -references
    public Data(int Dia, int Mes, int Ano)
    {
        this.Dia = Dia;
        this.Mes = Mes;
        this.Ano = Ano;
    }
}
```

Fonte: arquivo pessoal.



Encapsulamento de atributos

```
class Data
{
    private int Dia;
    private int Mes;
    private int Ano;

    -references
    public int GetDia()
    {
        return Dia;
    }
    -references
    public void SetDia(int Dia)
    {
        if (Dia > 0 && Dia <= 31)
        {
            this.Dia = Dia;
        }
    }
}
```

```
public int GetMes()
{
    return Mes;
}
- references
public void SetMes(int Mes)
{
    if (Mes > 0 && Mes <= 12)
    {
        this.Mes = Mes;
    }
}
- references
public int GetAno()
{
    return Ano;
}
- references
public void SetAno(int Ano)
{
    this.Ano = Ano;
}
```

Encapsulamento de atributos

```
public Data(int Dia, int Mes, int Ano)
{
    SetDia(Dia);
    SetMes(Mes);
    SetAno(Ano);
}
}
```

Fonte: arquivo pessoal.

Interatividade

Qual das alternativas abaixo pode ser afirmada com relação aos métodos construtores?

- a) Métodos construtores não possuem valor de retorno, por isso são sempre void.**
- b) Métodos construtores não podem receber parâmetros.**
- c) Métodos construtores podem ser executados a qualquer momento.**
- d) Métodos construtores podem ter qualquer nome.**
- e) Uma classe pode ter mais de um método construtor.**



Resposta

Qual das alternativas abaixo pode ser afirmada com relação aos métodos construtores?

- a) Métodos construtores não possuem valor de retorno, por isso são sempre void.
- b) Métodos construtores não podem receber parâmetros.
- c) Métodos construtores podem ser executados a qualquer momento.
- d) Métodos construtores podem ter qualquer nome.
- e) **Uma classe pode ter mais de um método construtor.**



Sobrecarga de construtores

```
public Data(int Dia, int Mes, int Ano)
{
    SetDia(Dia);
    SetMes(Mes);
    SetAno(Ano);
}
```

—references

```
public Data()
{
    SetDia(1);
    SetMes(1);
    SetAno(1);
}
```



Sobrecarga de construtores

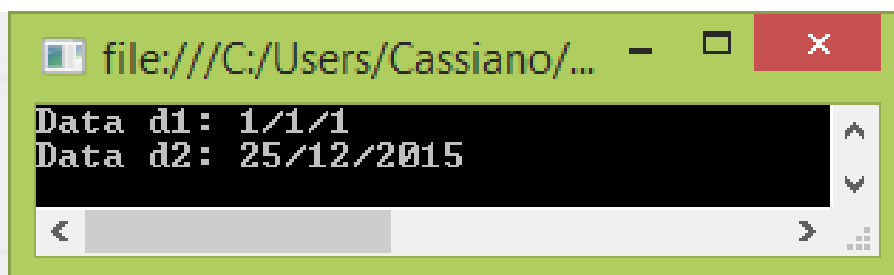
```
public Data(int Dia, int Mes, int Ano)
{
    SetDia(Dia);
    SetMes(Mes);
    SetAno(Ano);
}
- references
public Data()
    : this(1, 1, 1)
{
}
```

Sobrecarga de construtores

```
class Program
{
    -references
    public static void Main(string[] args)
    {
        Data d1 = new Data();
        Data d2 = new Data(25, 12, 2015);

        Console.WriteLine("Data d1: {0}/{1}/{2}",
            d1.GetDia(), d1.GetMes(), d1.GetAno());
        Console.WriteLine("Data d2: {0}/{1}/{2}",
            d2.GetDia(), d2.GetMes(), d2.GetAno());

        Console.ReadKey();
    }
}
```



```
file:///C:/Users/Cassiano/...
Data d1: 1/1/1
Data d2: 25/12/2015
```

Fonte: arquivo pessoal.

O modificador static

- O modificador static, quando aplicado a métodos ou atributos, indica que este método ou atributo pertence à classe, não aos objetos desta classe.



O modificador static

```
9  class Classe
10 {
11     -references
12     public double Multiplique(double a, double b)
13     {
14         return a * b;
15     }
16
17     -references
18     class Program
19     {
20         -references
21         public static void Main(string[] args)
22         {
23             Classe objeto = new Classe();
24             Console.WriteLine("2 x 3 = {0}", objeto.Multiplique(2, 3));
25
26             Console.ReadKey();
27         }
28     }
29 }
```

O modificador static

```
9  class Classe
10 {
    -references
11     public static double Multiplique(double a, double b)
12     {
13         return a * b;
14     }
15 }
16
    -references
17 class Program
18 {
    -references
19     public static void Main(string[] args)
20     {
21         Console.WriteLine("2 x 3 = {0}", Classe.Multiplique(2, 3));
22
23         Console.ReadKey();
24     }
25 }
```

O modificador static

```
class Classe
{
    private static int contador;
    -references
    public Classe()
    {
        contador++;
    }
    -references
    public int GetContador()
    {
        return contador;
    }
}
```

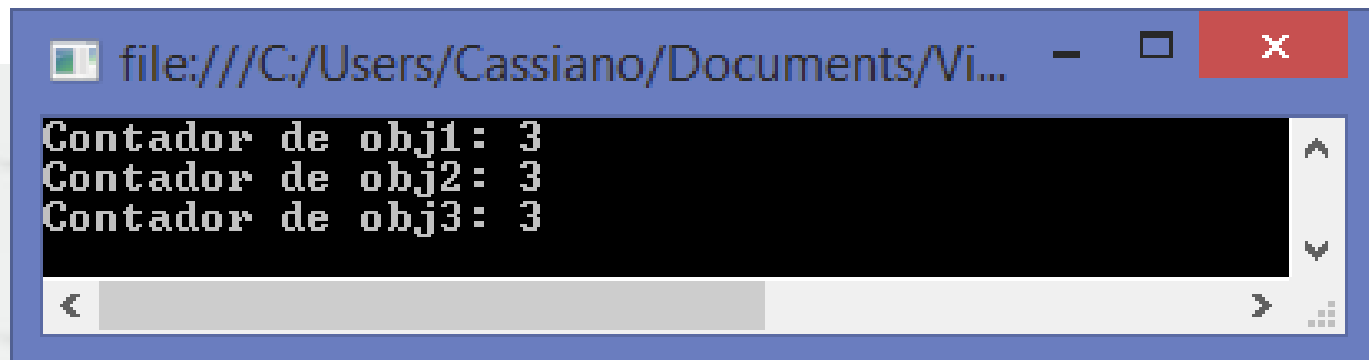
Fonte: arquivo pessoal.



O modificador static

```
class Program
{
    -references
    public static void Main(string[] args)
    {
        Classe obj1 = new Classe();
        Classe obj2 = new Classe();
        Classe obj3 = new Classe();

        Console.WriteLine("Contador de obj1: {0}", obj1.GetContador());
        Console.WriteLine("Contador de obj2: {0}", obj2.GetContador());
        Console.WriteLine("Contador de obj3: {0}", obj3.GetContador());
        Console.ReadKey();
    }
}
```



The screenshot shows a Windows command prompt window with a blue title bar. The title bar text is "file:///C:/Users/Cassiano/Documents/Vi...". The window contains three lines of text: "Contador de obj1: 3", "Contador de obj2: 3", and "Contador de obj3: 3". The text is displayed in a monospaced font on a black background. The window has standard Windows window controls (minimize, maximize, close) in the top right corner.

Fonte:
arquivo
pessoal.

Interatividade

Assinale a alternativa incorreta.

- a) Uma classe pode ter mais de um construtor.**
- b) O compilador decide qual sobrecarga de método utilizar a partir dos parâmetros passados para o método.**
- c) Um atributo static tem seu valor constante, ou seja, não pode ser alterado.**
- d) Um método static pode ser executado diretamente da classe, não necessitando de uma instância desta classe.**
- e) Um método static pode ser sobrecarregado.**

Resposta

Assinale a alternativa incorreta.

- a) Uma classe pode ter mais de um construtor.
- b) O compilador decide qual sobrecarga de método utilizar a partir dos parâmetros passados para o método.
- c) Um atributo static tem seu valor constante, ou seja, não pode ser alterado.
- d) Um método static pode ser executado diretamente da classe, não necessitando de uma instância desta classe.
- e) Um método static pode ser sobrecarregado.

Associação

-references

```
class Endereco
{
    public string Logradouro;
    public int Numero;
    public string Cidade;
}
```

-references

```
class Pessoa
{
    public string Nome;
    public Endereco Residencia;
}
```

```
class Program
```

```
{
    -references
    public static void Main(string[] args)
    {
        Pessoa cliente = new Pessoa();
        cliente.Nome = "Joaquim";

        Endereco casa = new Endereco();
        casa.Logradouro = "Rua das Flores";
        casa.Numero = 42;
        casa.Cidade = "Trás dos Montes";

        cliente.Residencia = casa;
    }
}
```

Fonte: arquivo pessoal.

Reutilização de classes – delegação

```
class Data
{
    public int dia;
    public int mes;
    public int ano;

    -references
    public Data(int dia, int mes, int ano)
    {
        this.dia = dia;
        this.mes = mes;
        this.ano = ano;
    }
}
```

```
class Hora
{
    public int hora;
    public int minuto;
    public int segundo;

    -references
    public Hora(int hora, int minuto, int segundo)
    {
        this.hora = hora;
        this.minuto = minuto;
        this.segundo = segundo;
    }
}
```

Fonte: arquivo pessoal.

Reutilização de classes – delegação

```
class DataHora
{
    public Data EstaData;
    public Hora EstaHora;

    -references
    public DataHora(int Dia, int Mes, int Ano, int Hora, int Minuto, int Segundo)
    {
        EstaData = new Data(Dia, Mes, Ano);
        EstaHora = new Hora(Hora, Minuto, Segundo);
    }
}
```

Fonte: arquivo pessoal.

Reutilização de classes – herança

```
class Pessoa
{
    public string Nome;
    public int Telefone;
    -references
    public Pessoa(string Nome, int Telefone)
    {
        this.Nome = Nome;
        this.Telefone = Telefone;
    }
}
```

Fonte:
arquivo
pessoal.

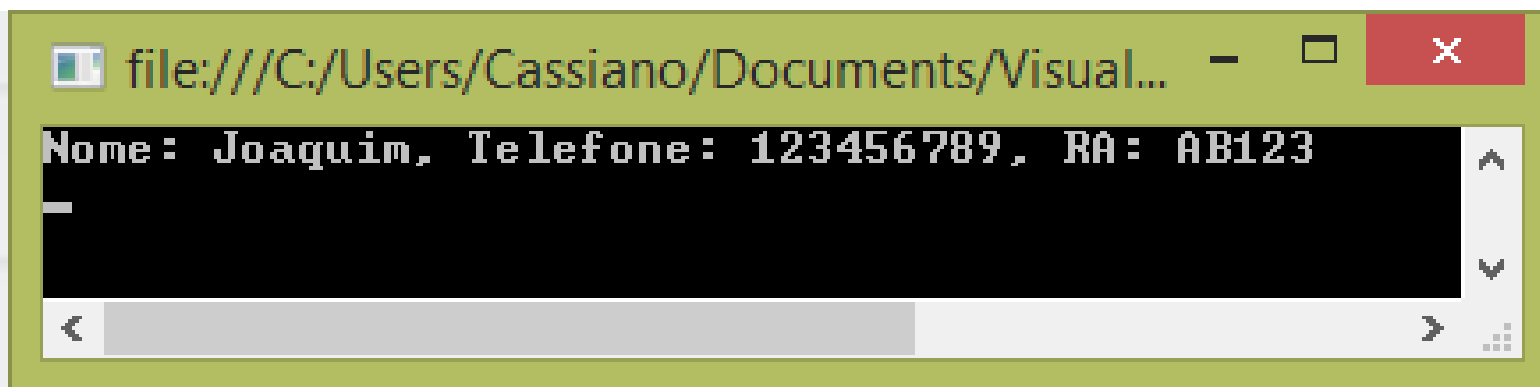
```
class Aluno : Pessoa
{
    public string Ra;
    -references
    public Aluno(string Nome, int Telefone, string Ra)
        : base(Nome, Telefone)
    {
        this.Ra = Ra;
    }
}
```



Reutilização de classes – herança

```
class Program
{
    -references
    public static void Main(string[] args)
    {
        Aluno a = new Aluno("Joaquim", 123456789, "AB123");
        Console.WriteLine("Nome: {0}, Telefone: {1}, RA: {2}",
            a.Nome, a.Telefone, a.Ra);

        Console.ReadKey();
    }
}
```

A screenshot of a Windows command prompt window. The title bar shows the file path "file:///C:/Users/Cassiano/Documents/Visual...". The command prompt displays the output of a C# program: "Nome: Joaquim, Telefone: 123456789, RA: AB123". Below the output, there is a horizontal line and a cursor. The window has a green border and standard Windows window controls (minimize, maximize, close) in the top right corner.

```
file:///C:/Users/Cassiano/Documents/Visual...
Nome: Joaquim, Telefone: 123456789, RA: AB123
_
```

Polimorfismo

```
class Program
{
    -references
    public static void Escola(Aluno aluno)
    {
        Console.WriteLine("O aluno {0} entrou na escola.", aluno.Nome);
    }
    -references
    public static void Lanchonete(Pessoa pessoa)
    {
        Console.WriteLine("A pessoa {0} entrou na lanchonete.", pessoa.Nome);
    }
    -references
    public static void Main(string[] args)
    {
        Aluno a = new Aluno("Joaquim", 123456789, "AB123");
        Pessoa p = new Pessoa("Maria", 987654321);

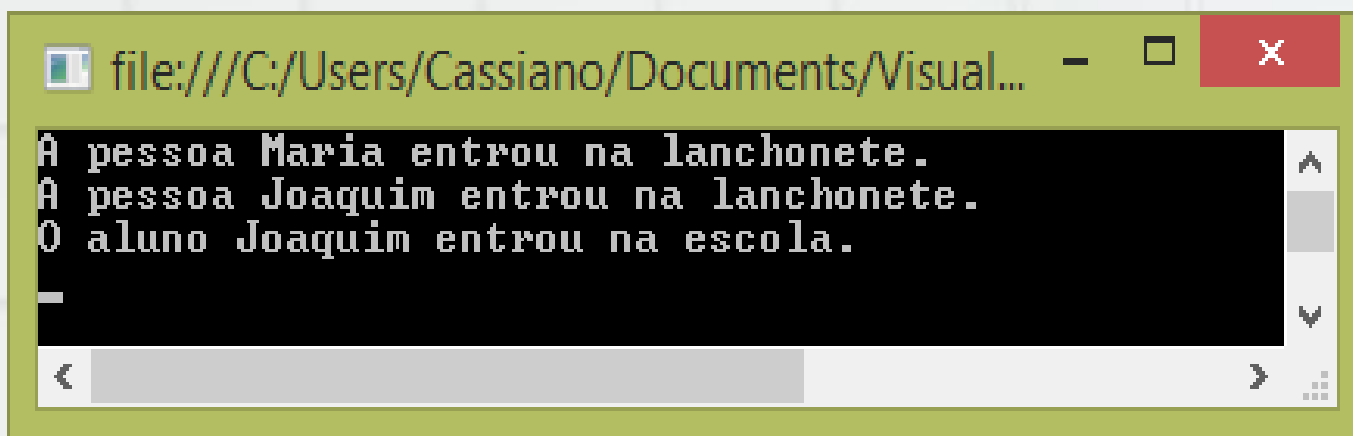
        Lanchonete(p);
        Lanchonete(a);
        Escola(a);
        Escola(p);
    }
}
```

Polimorfismo

```
public static void Main(string[] args)
{
    Aluno a = new Aluno("Joaquim", 123456789, "AB123");
    Pessoa p = new Pessoa("Maria", 987654321);

    Lanchonete(p);
    Lanchonete(a);
    Escola(a);

    Console.ReadKey();
}
```



Fonte: arquivo pessoal.



Interatividade

O paradigma orientado a objetos incentiva a prática da reutilização de código. Qual das alternativas abaixo não se aplica a este conceito?

- a) Com o polimorfismo entre classes, economizamos a escrita de código
- b) A delegação permite reutilizar classes em outras classes.
- c) A herança entre classes faz com que definições de atributos e métodos sejam passados de uma classe a outra.
- d) Podemos reutilizar um método construtor de uma superclasse invocando-o do construtor de uma subclasse.
- e) A reutilização de código torna a manutenção do sistema mais rápida e barata.

Resposta

O paradigma orientado a objetos incentiva a prática da reutilização de código. Qual das alternativas abaixo não se aplica a este conceito?

- a) **Com o polimorfismo entre classes, economizamos a escrita de código**
- b) A delegação permite reutilizar classes em outras classes.
- c) A herança entre classes faz com que definições de atributos e métodos sejam passados de uma classe a outra.
- d) Podemos reutilizar um método construtor de uma superclasse invocando-o do construtor de uma subclasse.
- e) A reutilização de código torna a manutenção do sistema mais rápida e barata.

ATÉ A PRÓXIMA!

