

# Curso C# Completo Programação Orientada a Objetos + Projetos

**Capítulo: Interfaces**

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

## Interfaces

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

## Interface

Interface é um tipo que define um conjunto de operações que uma classe (ou struct) deve implementar.

A interface estabelece um **contrato** que a classe (ou struct) deve cumprir.

```
interface IShape {  
  
    double Area();  
    double Perimeter();  
}
```

Pra quê interfaces?

- Para criar sistemas com **baixo acoplamento** e **flexíveis**.

## Problema exemplo

Uma locadora brasileira de carros cobra um valor por hora para locações de até 12 horas. Porém, se a duração da locação ultrapassar 12 horas, a locação será cobrada com base em um valor diário. Além do valor da locação, é acrescido no preço o valor do imposto conforme regras do país que, no caso do Brasil, é 20% para valores até 100.00, ou 15% para valores acima de 100.00. Fazer um programa que lê os dados da locação (modelo do carro, instante inicial e final da locação), bem como o valor por hora e o valor diário de locação. O programa deve então gerar a nota de pagamento (contendo valor da locação, valor do imposto e valor total do pagamento) e informar os dados na tela. Veja os exemplos.

**Example 1:**

Enter rental data  
 Car model: **Civic**  
 Pickup (dd/MM/yyyy hh:mm): **25/06/2018 10:30**  
 Return (dd/MM/yyyy hh:mm): **25/06/2018 14:40**  
 Enter price per hour: **10.00**  
 Enter price per day: **130.00**  
 INVOICE:  
 Basic payment: 50.00  
 Tax: 10.00  
 Total payment: 60.00

*Calculations:*

*Duration = (25/06/2018 14:40) - (25/06/2018 10:30) = 4:10 = 5 hours*

*Basic payment = 5 \* 10 = 50*

*Tax = 50 \* 20% = 50 \* 0.2 = 10*

**Example 2:**

Enter rental data  
 Car model: **Civic**  
 Pickup (dd/MM/yyyy hh:mm): **25/06/2018 10:30**  
 Return (dd/MM/yyyy hh:mm): **27/06/2018 11:40**  
 Enter price per hour: **10.00**  
 Enter price per day: **130.00**  
 INVOICE:  
 Basic payment: 390.00  
 Tax: 58.50  
 Total payment: 448.50

*Calculations:*

*Duration = (27/06/2018 11:40) - (25/06/2018 10:30) = 2 days + 1:10 = 3 days*

*Basic payment = 3 \* 130 = 390*

*Tax = 390 \* 15% = 390 \* 0.15 = 58.50*

# Solução do problema

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

(recordando - cap. 6)

Views

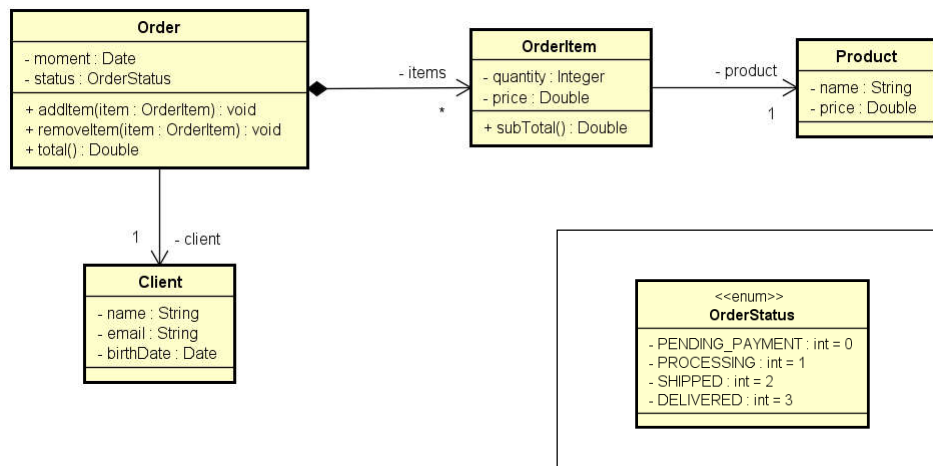
Controllers

Entities

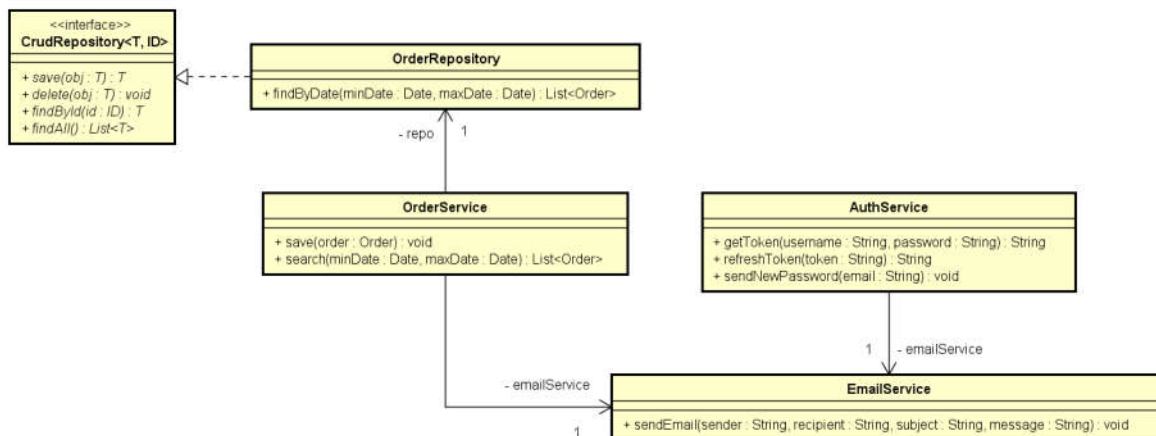
Services

Repositories

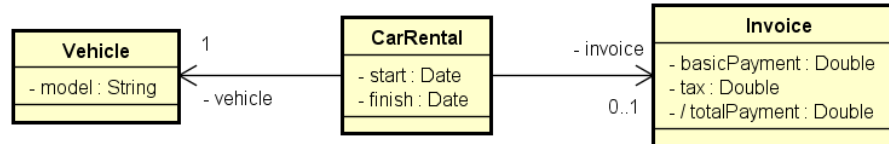
## Entities



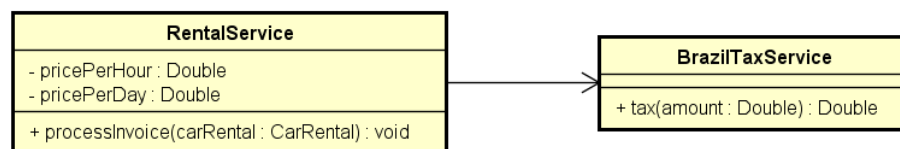
## Services



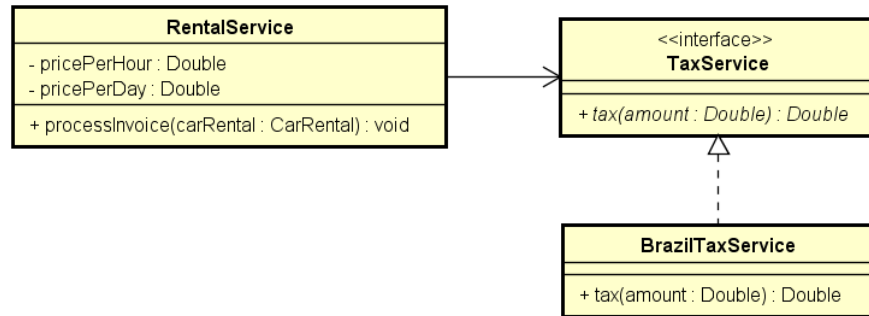
## Domain layer design



## Service layer design (no interface)



## Service layer design



## Projeto no Github

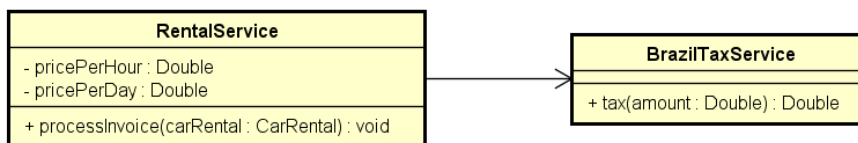
<https://github.com/acenelio/interfaces1-csharp>

# Inversão de controle, Injeção de dependência

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

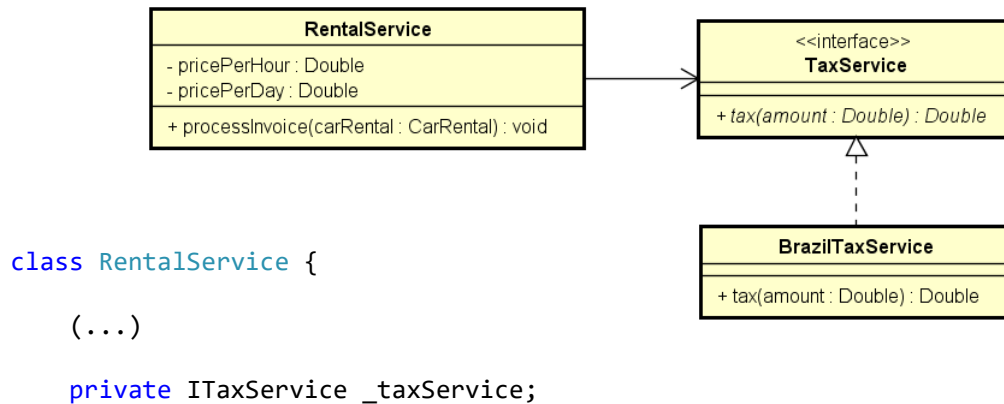
- Acoplamento forte
- A classe RentalService conhece a dependência concreta
- Se a classe concreta mudar, é preciso mudar a classe RentalService



```
class RentalService {  
    (...)  
    private BrazilTaxService _brazilTaxService = new BrazilTaxService();  
}
```



- Acoplamento fraco
- A classe RentalService não conhece a dependência concreta
- Se a classe concreta mudar, a classe RentalService não muda nada



## Injeção de dependência por meio de construtor

```

class Program {
    static void Main(string[] args) {
        (...)
        RentalService rentalService = new RentalService(hour, day, new BrazilTaxService());
    }
}
  
```

```

class RentalService {
    private ITaxService _taxService;

    public RentalService(double pricePerHour, double pricePerDay, ITaxService taxService) {
        PricePerHour = pricePerHour;
        PricePerDay = pricePerDay;
        _taxService = taxService;
    }
}
  
```

A red arrow points from the new BrazilTaxService() expression in the Program class to the ITaxService taxService parameter in the RentalService constructor, illustrating the injection of the concrete dependency.

## Inversão de controle

- **Inversão de controle**

Padrão de desenvolvimento que consiste em retirar da classe a responsabilidade de instanciar suas dependências.

- **Injeção de dependência**

É uma forma de realizar a inversão de controle: um componente externo instancia a dependência, que é então injetada no objeto "pai". Pode ser implementada de várias formas:

- Construtor
- Objeto de instanciação (builder / factory)
- Container / framework

## Exercício de fixação

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Uma empresa deseja automatizar o processamento de seus contratos. O processamento de um contrato consiste em gerar as parcelas a serem pagas para aquele contrato, com base no número de meses desejado.

A empresa utiliza um serviço de pagamento online para realizar o pagamento das parcelas. Os serviços de pagamento online tipicamente cobram um juro mensal, bem como uma taxa por pagamento. Por enquanto, o serviço contratado pela empresa é o do Paypal, que aplica **juros simples** de 1% a cada parcela, mais uma **taxa** de pagamento de 2%.

Fazer um programa para ler os dados de um contrato (número do contrato, data do contrato, e valor total do contrato). Em seguida, o programa deve ler o número de meses para parcelamento do contrato, e daí gerar os registros de parcelas a serem pagas (data e valor), sendo a primeira parcela a ser paga um mês após a data do contrato, a segunda parcela dois meses após o contrato e assim por diante. Mostrar os dados das parcelas na tela.

Veja exemplo na próxima página.

### Example:

```
Enter contract data
Number: 8028
Date (dd/MM/yyyy): 25/06/2018
Contract value: 600.00
Enter number of installments: 3
Installments:
25/07/2018 - 206.04
25/08/2018 - 208.08
25/09/2018 - 210.12
```

### Calculations (1% monthly simple interest + 2% payment fee):

#### Quota #1:

$200 + 1\% * 1 = 202$   
 $202 + 2\% = 206.04$

#### Quota #2:

$200 + 1\% * 2 = 204$   
 $204 + 2\% = 208.08$

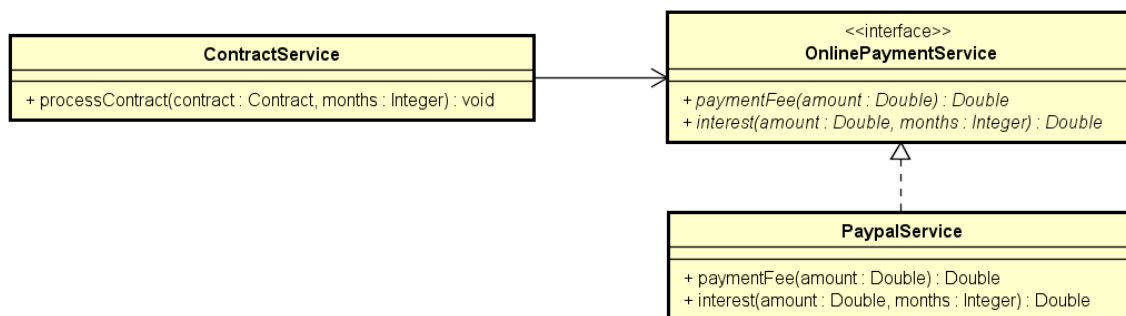
#### Quota #3:

$200 + 1\% * 3 = 206$   
 $206 + 2\% = 210.12$

## Domain layer design (entities)



## Service layer design



## Repositório Github

<https://github.com/acenelio/interfaces4-csharp>

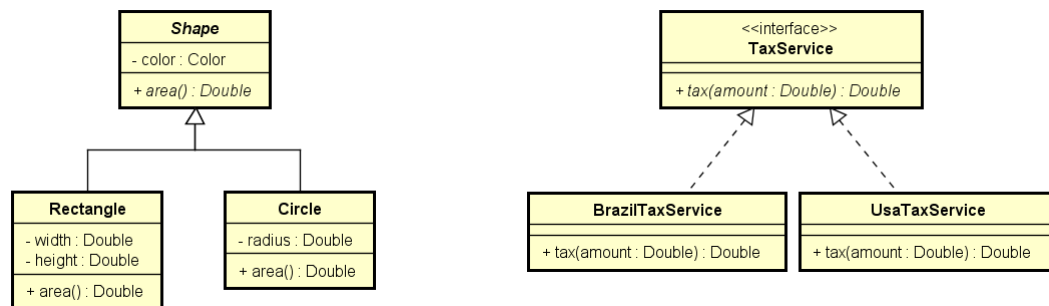
## Herdar vs. cumprir contrato

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

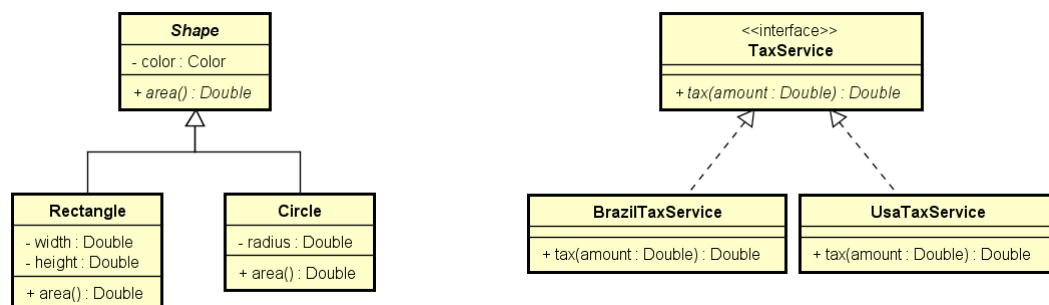
## Aspectos em comum entre herança e interfaces

- Relação é-um
- Generalização/especialização
- Polimorfismo

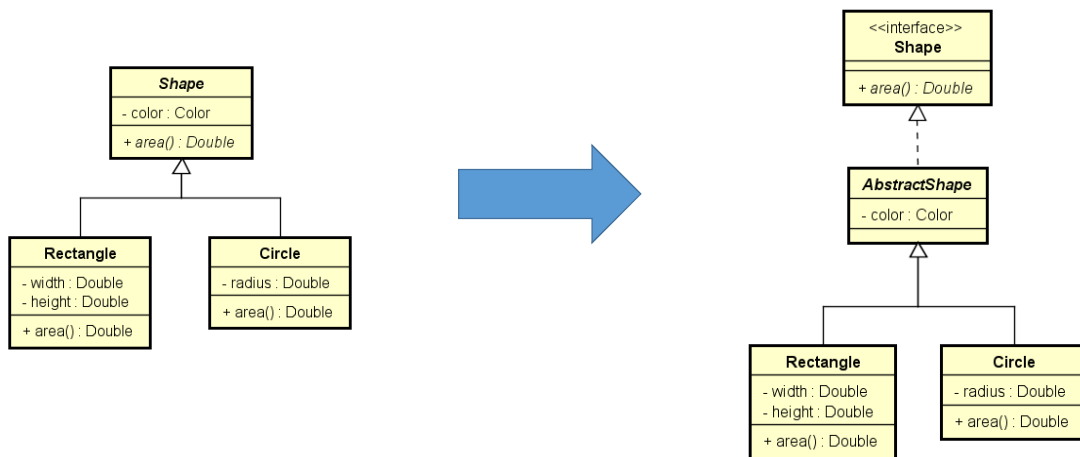


## Diferença fundamental

- Herança => reuso
- Interface => contrato a ser cumprido



E se eu precisar implementar Shape como interface, porém também quiser definir uma estrutura comum reutilizável para todas figuras?



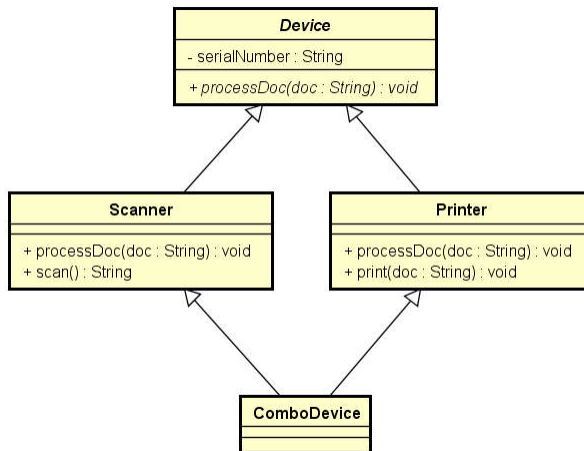
<https://github.com/acenelio/interfaces2-csharp>

## Herança múltipla e o problema do diamante

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

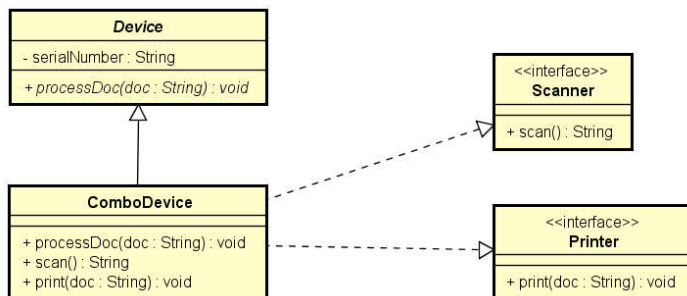
## Problema do diamante



A herança múltipla pode gerar o problema do diamante: uma ambiguidade causada pela existência do mesmo método em mais de uma superclasse.

Herança múltipla não é permitida na maioria das linguagens!

## Porém, uma classe (ou struct) pode implementar mais de uma interface



### ATENÇÃO:

Isso NÃO é herança múltipla, pois NÃO HÁ REUSO na relação entre ComboDevice e as interfaces Scanner e Printer.

ComboDevice não herda, mas sim implementa as interfaces (cumpre o contrato).

<https://github.com/acenelio/interfaces3-csharp>



# Interface IComparable

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

# Interface Comparable

[https://msdn.microsoft.com/en-us/library/system.icomparable\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/system.icomparable(v=vs.110).aspx)

```
public interface IComparable {  
    int CompareTo(object other);  
}
```

## Problema motivador

Faça um programa para ler um arquivo contendo nomes de pessoas (um nome por linha), armazenando-os em uma lista. Depois, ordenar os dados dessa lista e mostra-los ordenadamente na tela. Nota: o caminho do arquivo pode ser informado "*hardcode*".

Maria Brown  
Alex Green  
Bob Grey  
Anna White  
Alex Black  
Eduardo Rose  
Willian Red  
Marta Blue  
Alex Brown

```
using System;
using System.IO;
using System.Collections.Generic;

namespace Course {
    class Program {
        static void Main(string[] args) {

            string path = @"c:\temp\in.txt";

            try {
                using (StreamReader sr = File.OpenText(path)) {
                    List<string> list = new List<string>();
                    while (!sr.EndOfStream) {
                        list.Add(sr.ReadLine());
                    }
                    list.Sort();
                    foreach (string str in list) {
                        Console.WriteLine(str);
                    }
                }
            }
            catch (IOException e) {
                Console.WriteLine("An error occurred");
                Console.WriteLine(e.Message);
            }
        }
    }
}
```

## Outro problema

Faça um programa para ler um arquivo contendo funcionários (nome e salário) no formato .csv, armazenando-os em uma lista. Depois, ordenar a lista por nome e mostrar o resultado na tela. Nota: o caminho do arquivo pode ser informado "*hardcode*".

```
Maria Brown,4300.00
Alex Green,3100.00
Bob Grey,3100.00
Anna White,3500.00
Alex Black,2450.00
Eduardo Rose,4390.00
Willian Red,2900.00
Marta Blue,6100.00
Alex Brown,5000.00
```

```
using System.Globalization;

namespace Course {
    class Employee {

        public string Name { get; set; }
        public double Salary { get; set; }

        public Employee(string csvEmployee) {
            string[] vect = csvEmployee.Split(',');
            Name = vect[0];
            Salary = double.Parse(vect[1], CultureInfo.InvariantCulture);
        }

        public override string ToString() {
            return Name + ", " + Salary.ToString("F2", CultureInfo.InvariantCulture);
        }
    }
}
```

# Interface IComparable

```
namespace System {
    public interface IComparable {

        int CompareTo(object obj);

    }
}
```

```
Console.WriteLine("maria".CompareTo("alex"));
Console.WriteLine("alex".CompareTo("maria"));
Console.WriteLine("maria".CompareTo("maria"));
```

## Output:

```
1
-1
0
```

Value	Meaning
Less than zero	The current instance precedes the object specified by the <a href="#">CompareTo</a> method in the sort order.
Zero	This current instance occurs in the same position in the sort order as the object specified by the <a href="#">CompareTo</a> method.
Greater than zero	This current instance follows the object specified by the <a href="#">CompareTo</a> method in the sort order.

```
using System;
using System.Globalization;

namespace Course {
    class Employee : IComparable {

        public string Name { get; set; }
        public double Salary { get; set; }

        public Employee(string csvEmployee) {
            string[] vect = csvEmployee.Split(',');
            Name = vect[0];
            Salary = double.Parse(vect[1], CultureInfo.InvariantCulture);
        }

        public override string ToString() {
            return Name + ", " + Salary.ToString("F2", CultureInfo.InvariantCulture);
        }

        public int CompareTo(object obj) {
            if (!(obj is Employee)) {
                throw new ArgumentException("Comparing error: argument is not an Employee");
            }
            Employee other = obj as Employee;
            return Name.CompareTo(other.Name);
        }
    }
}
```