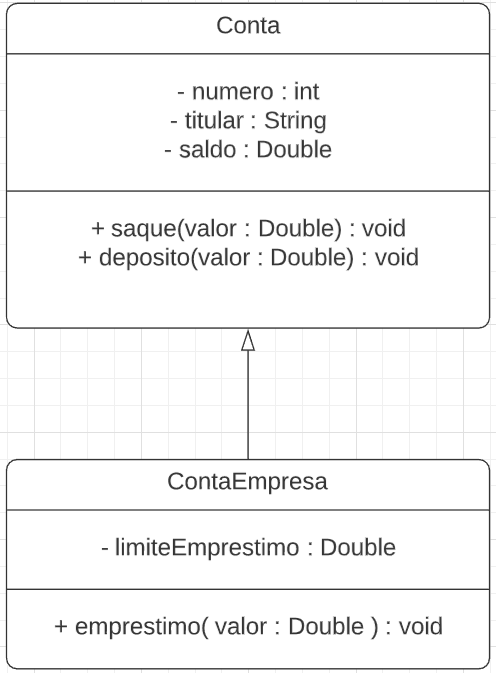
# Upcasting e Downcasting 15/12/2020

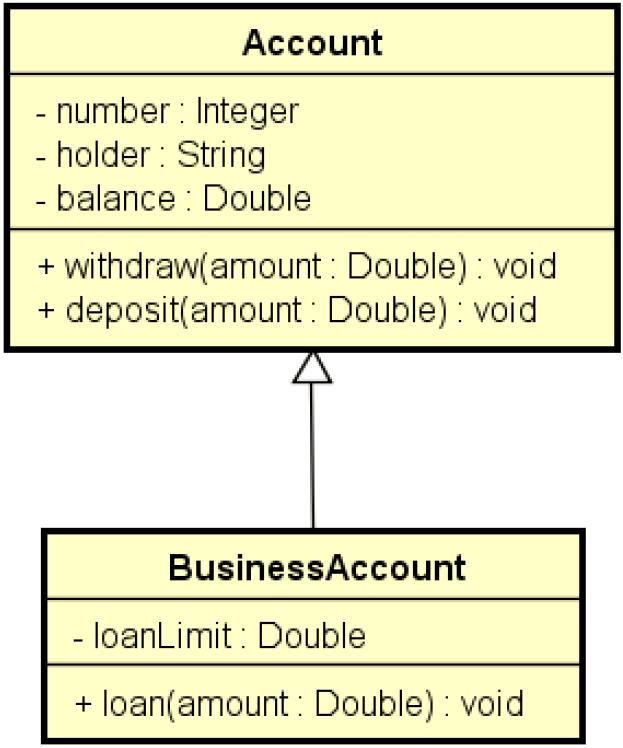
## Upcasting

* Casting da subclasse para superclasse (Casting = converter de um tipo para outro).
  + Seria como converter um objeto do tipo ContaEmpresa para uma variável do tipo Conta.
* Uso comum: polimorfismo (próximas aulas)

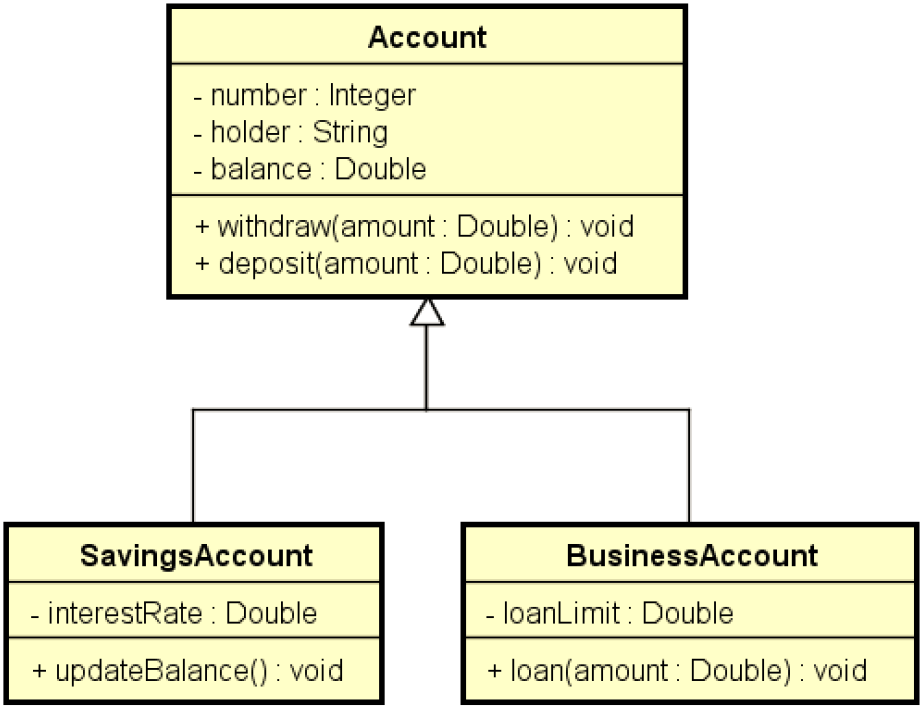
## Downcasting

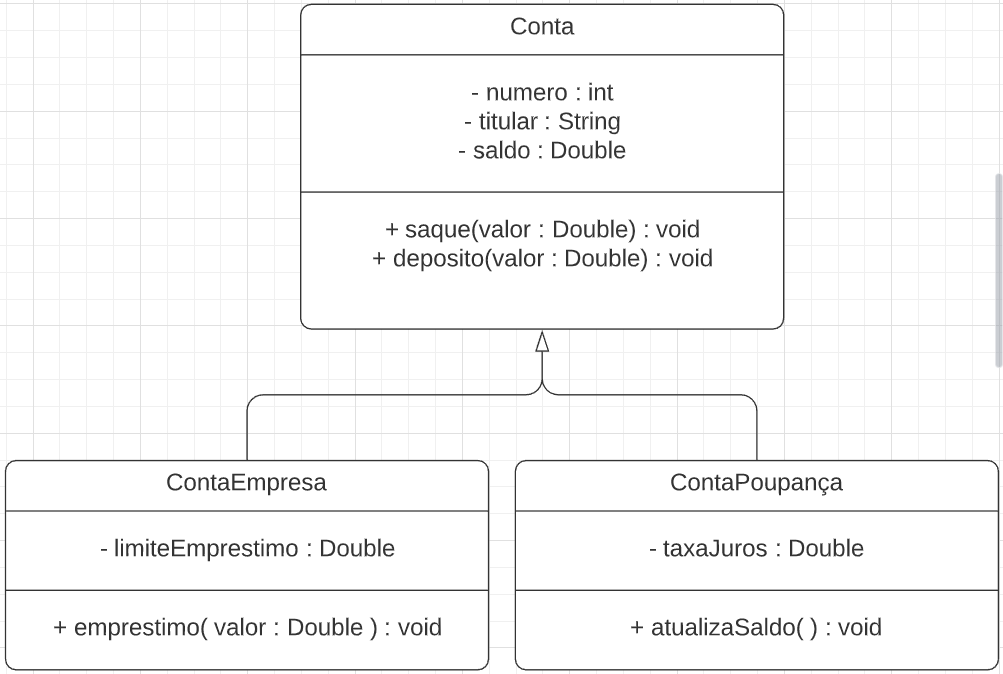
* Casting da superclasse para subclasse
  + Seria como converter um objeto da Conta para uma variável da ContaEmpresa.
* Palavra **as**
* Palavra **is**
* Uso comum: métodos que recebem parâmetros genéricos (ex: Equals – será visto posteriormente)

****

****

**Exemplo:**

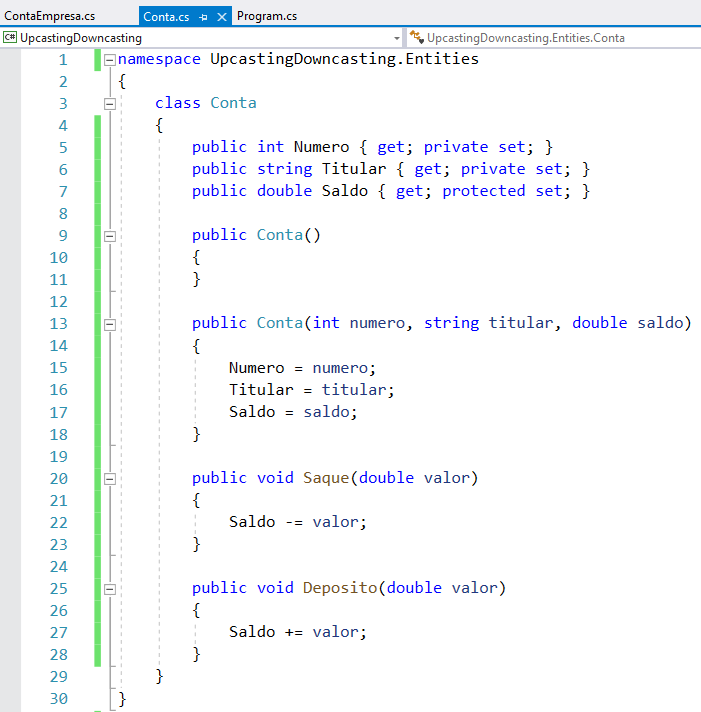
****

****

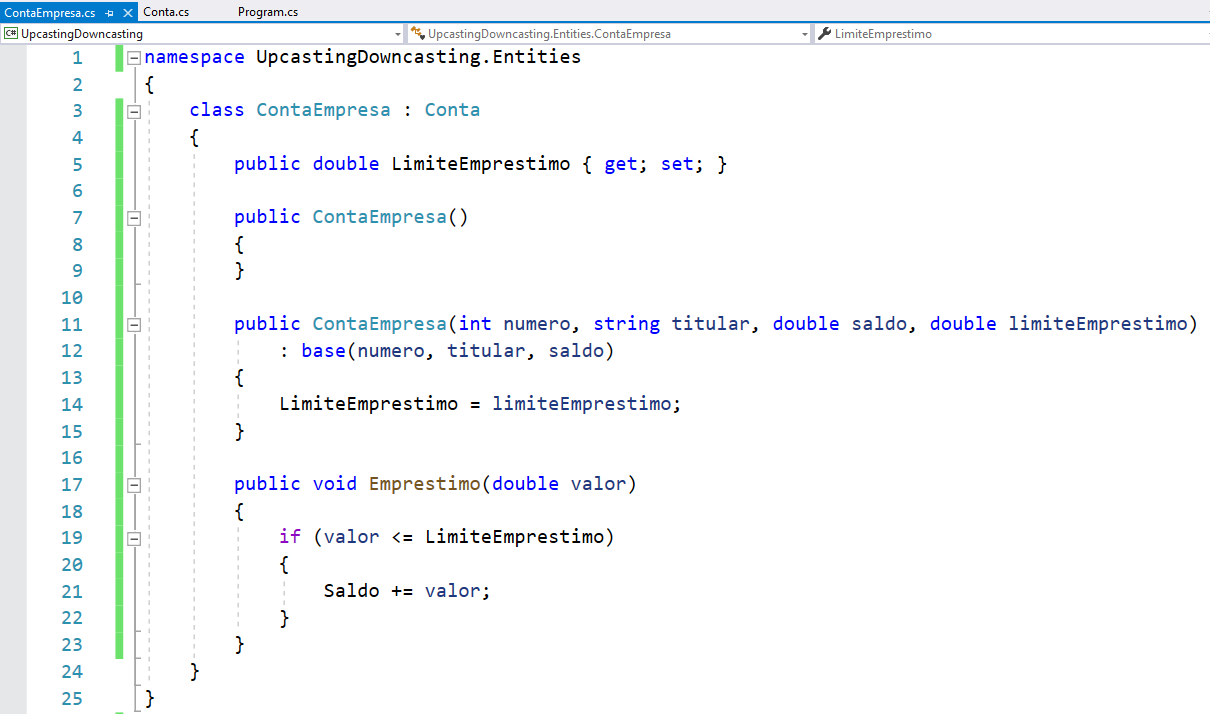
Vamos pegar o código da aula de Herança.

Criei um projeto com o nome UpcastingDowncasting, criei a pasta Entities. Dentro da pasta Entities criei as classes Conta e ContaEmpresa.

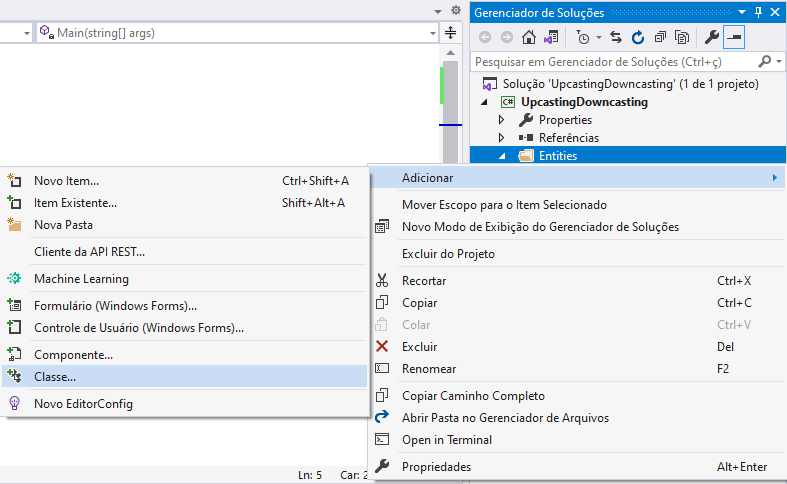
Classe Conta:



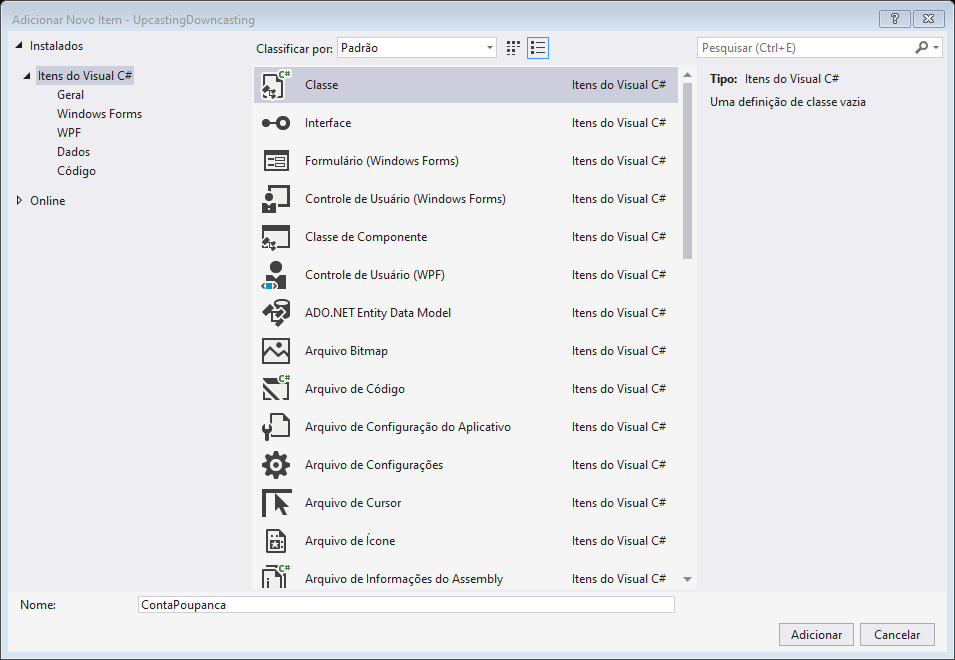
Classe ContaEmpresa:



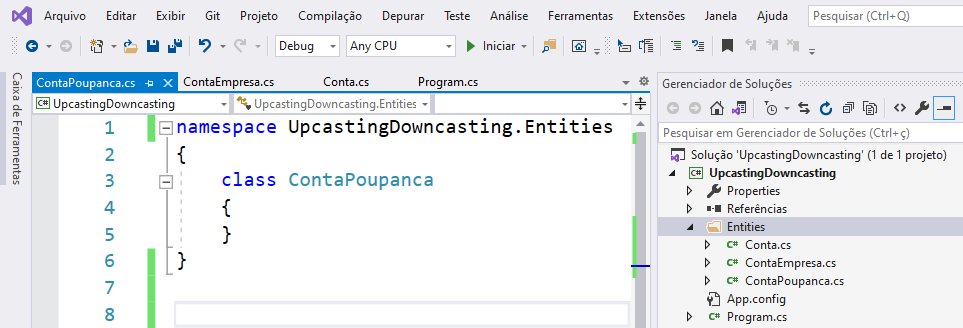
Criando a classe **ContaPoupanca**



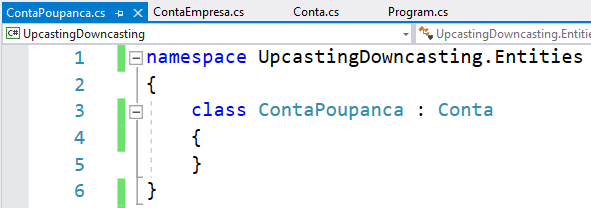
Nome ContaPoupanca



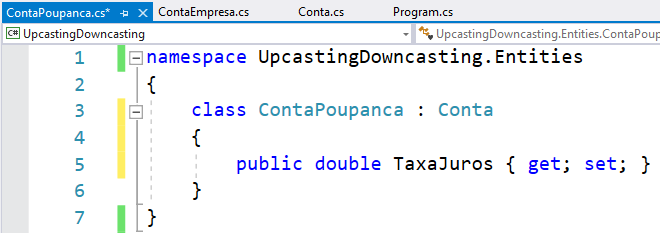
Classe criada.



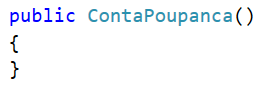
Criando Herança



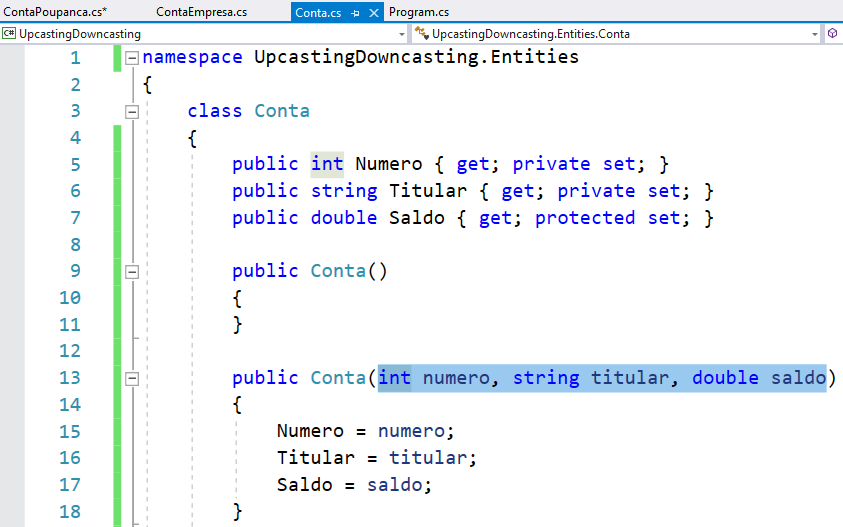
Incluindo a propriedade taxaJuros



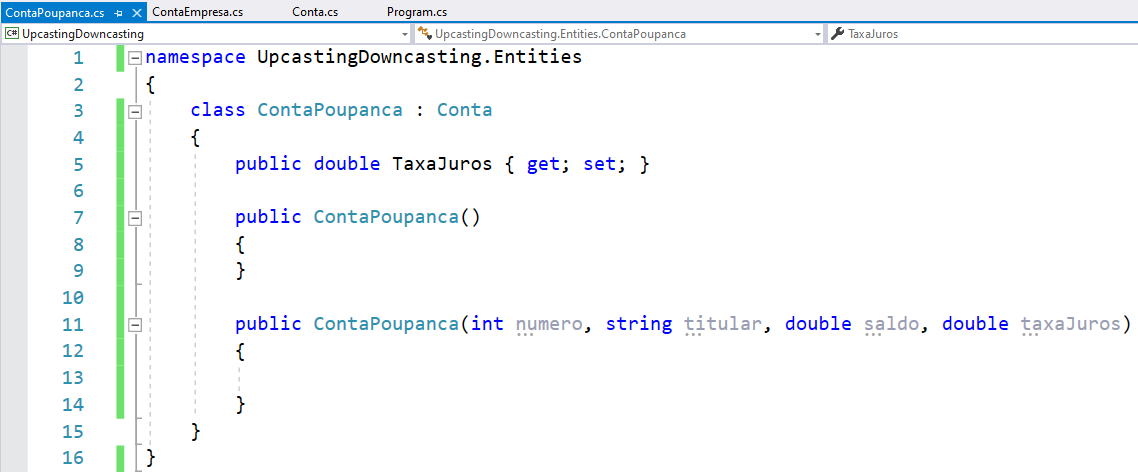
Construtor padrão



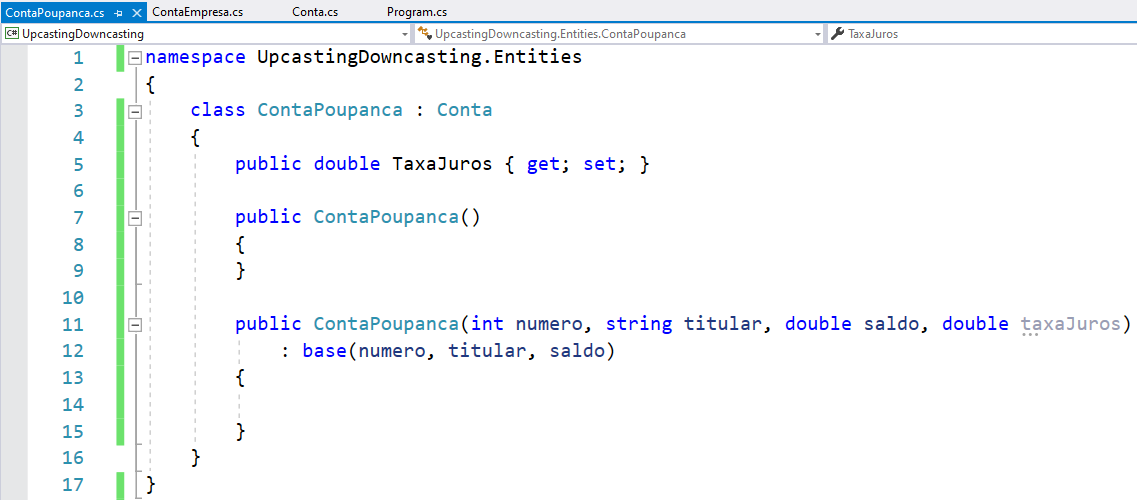
Construtor com argumentos. Para isso, podemos copiar os parâmetros do construtor da classe Conta.



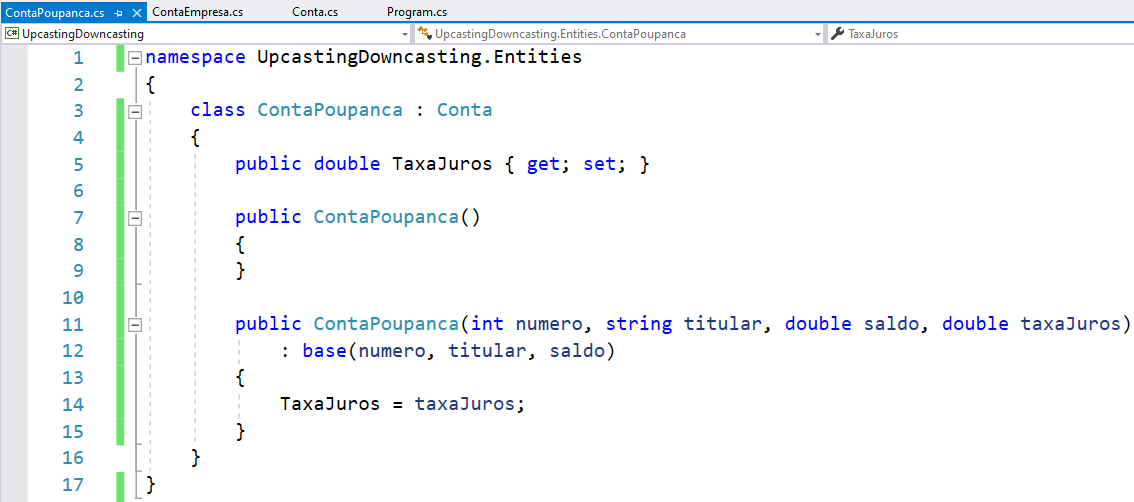
E colar no construtor da classe ContaPoupanca, e acrescentar o parâmetro próprio da classe.



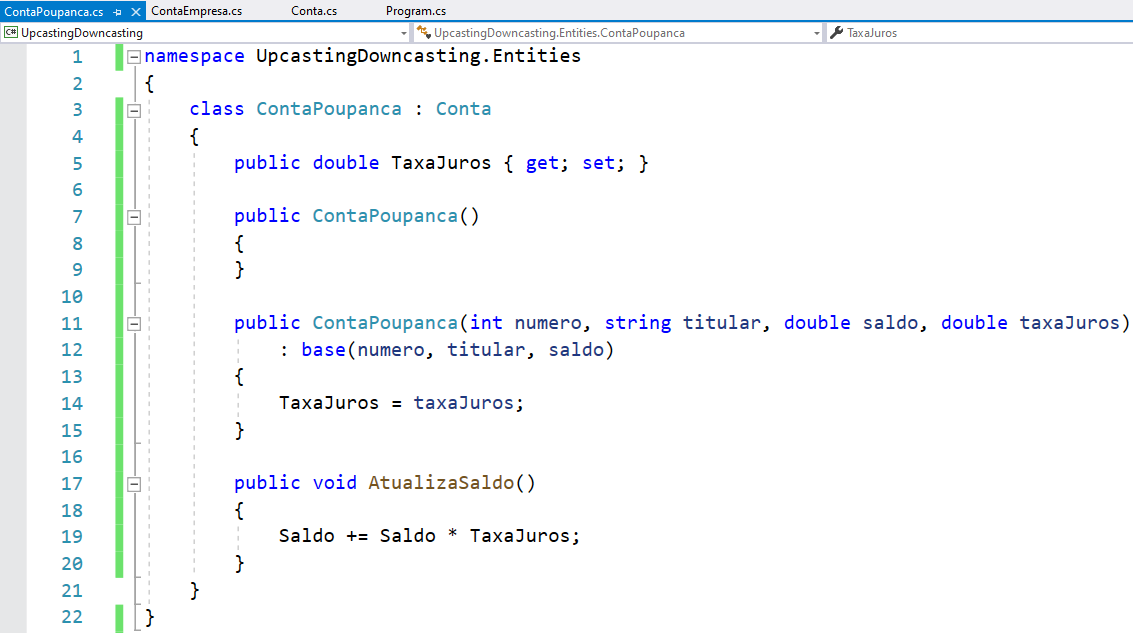
Agora vamos aproveitar o construtor da classe Conta.



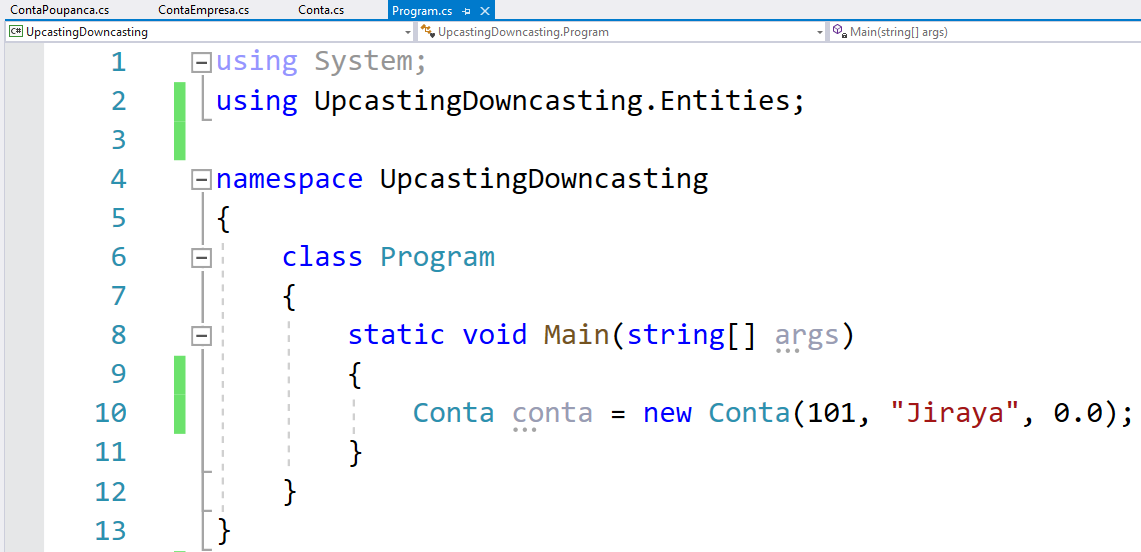
Agora, no corpo do construtor, vamos dizer que o atributo TaxaJuros irá receber o argumento taxaJuros que chegou como parâmetro de entrada.



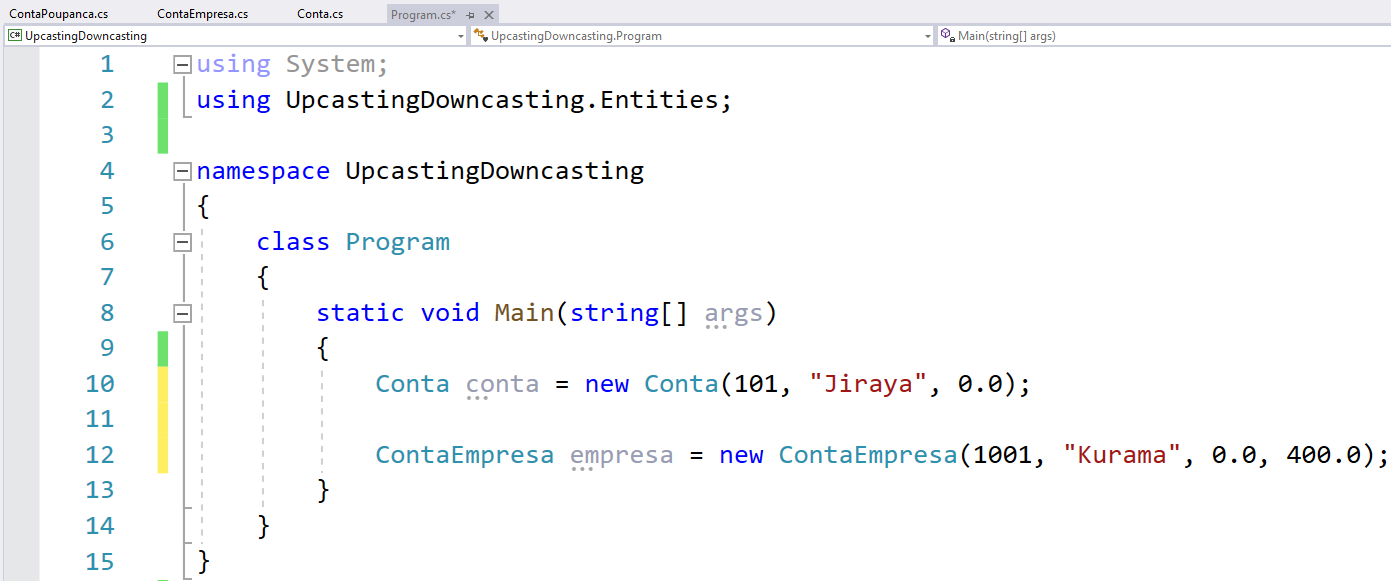
Vamos criar o método AtualizaSaldo. Esse método atualiza o saldo da conta de acordo com a taxa de juros.



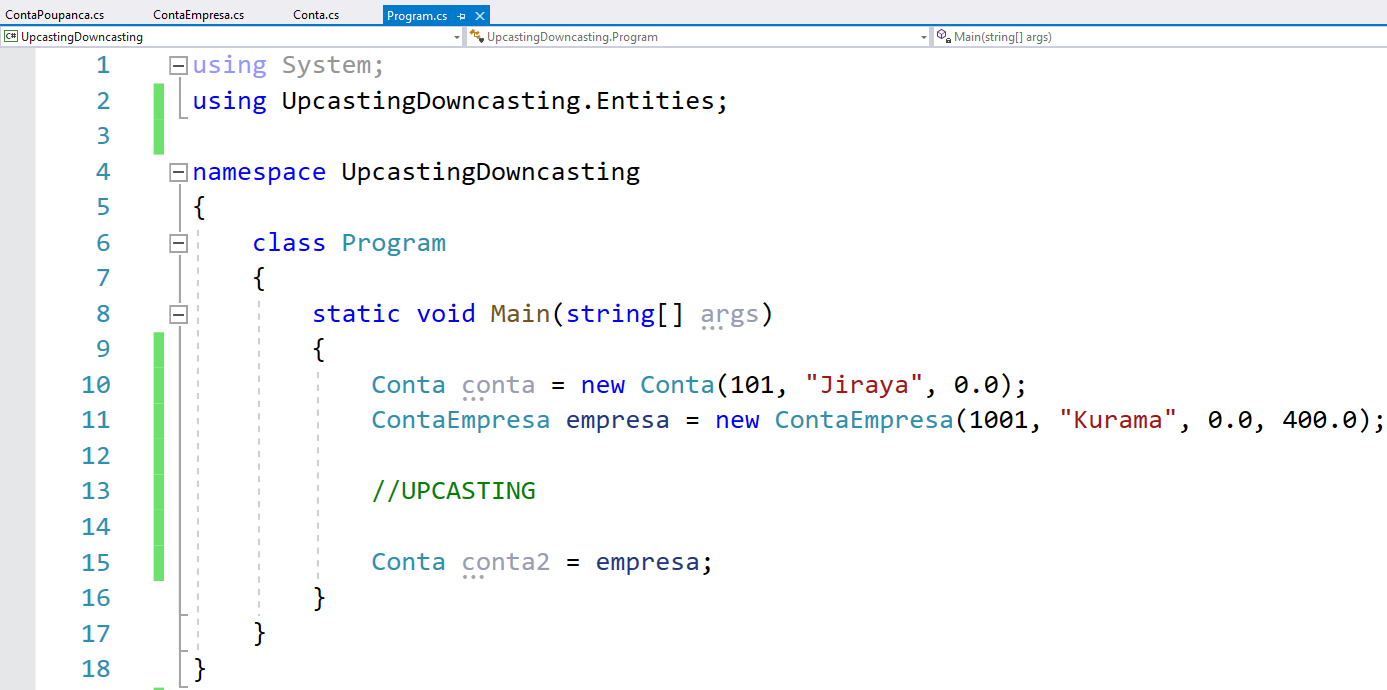
Vamos testar agora no programa principal. Vamos instanciar um objeto do tipo Conta.

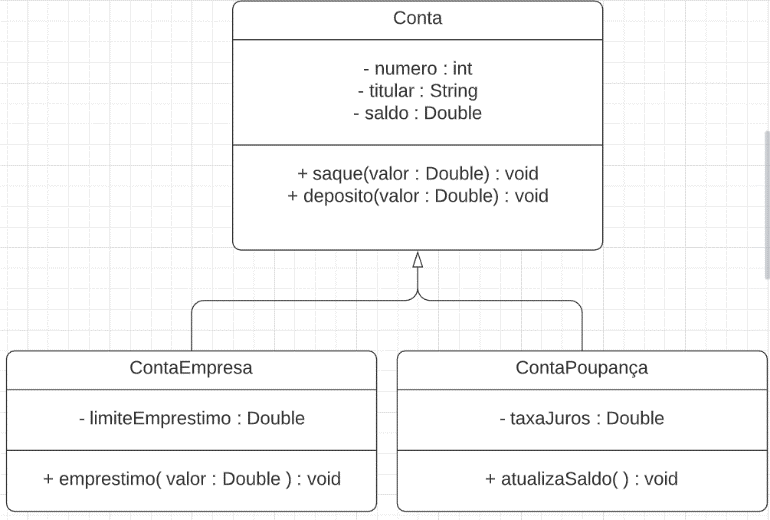


Vamos instanciar um objeto do tipo ContaEmpresa.



Agora sim. Vamos fazer um teste de UpCasting – Uma conversão da **subclasse** ContaEmpresa para uma **superclasse** Conta.



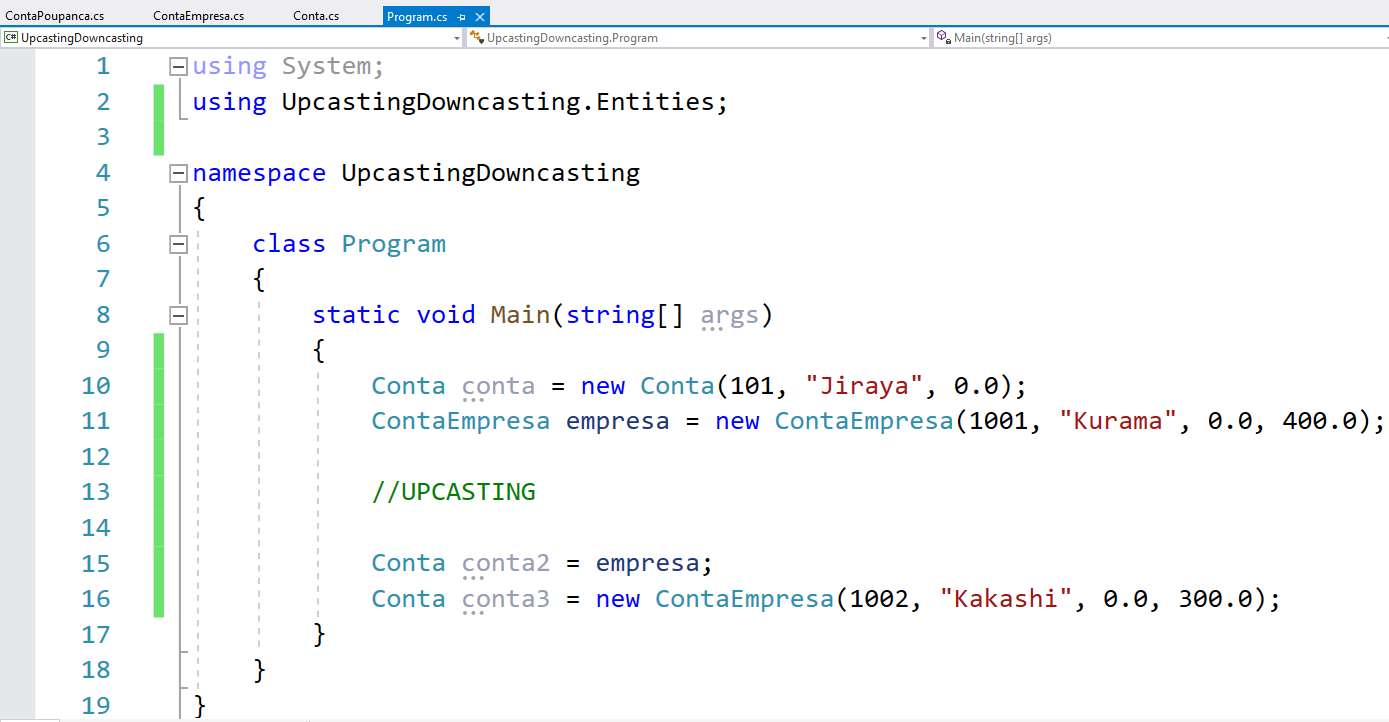
Por conta da condição “**é um**” e a **herança**.

Podemos dizer também que a ContaPoupanca **é uma** Conta.

Podemos dizer que a ContaEmpresa **é uma** Conta.

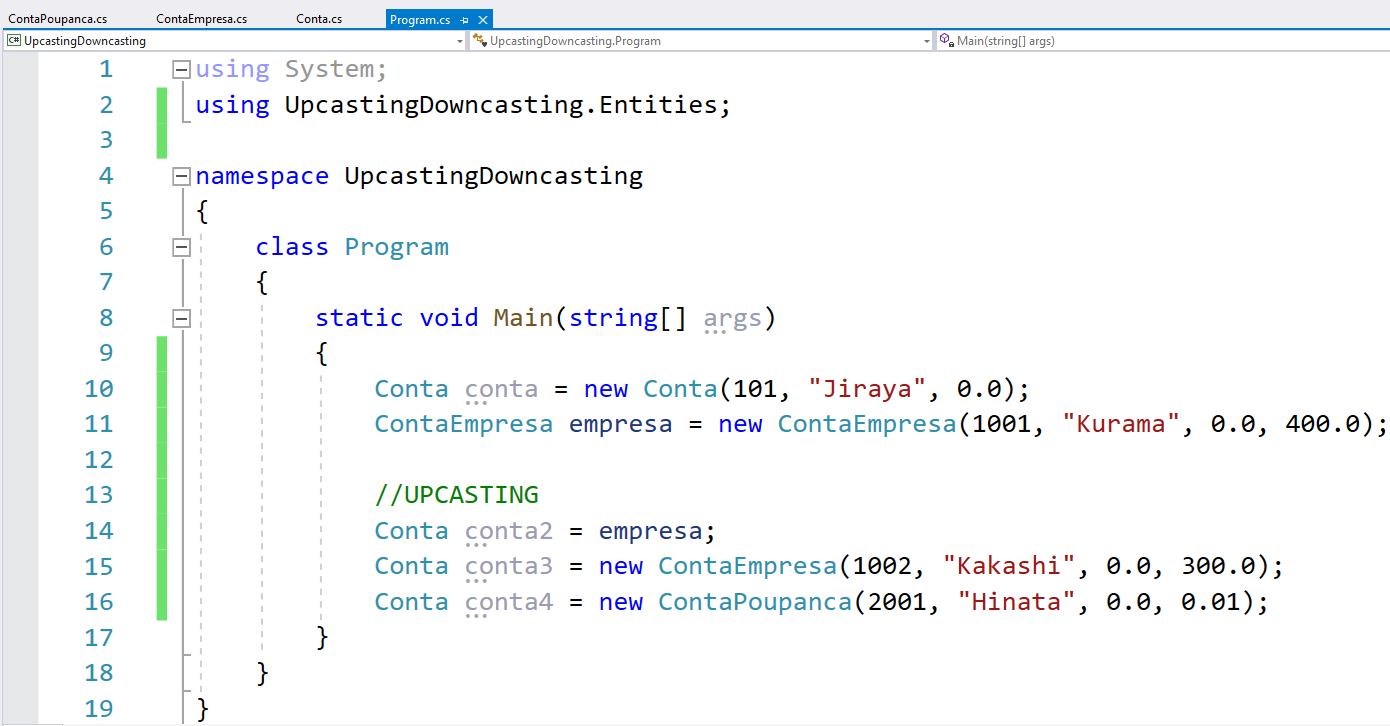
Por isso, podemos fazer a atribuição da linha 15 perfeitamente. Ou seja, uma variável de uma superclasse, aceita qualquer outra de suas subclasses.

Continuando.



A variável conta3 do tipo Conta está recebendo uma nova conta do tipo ContaEmpresa.

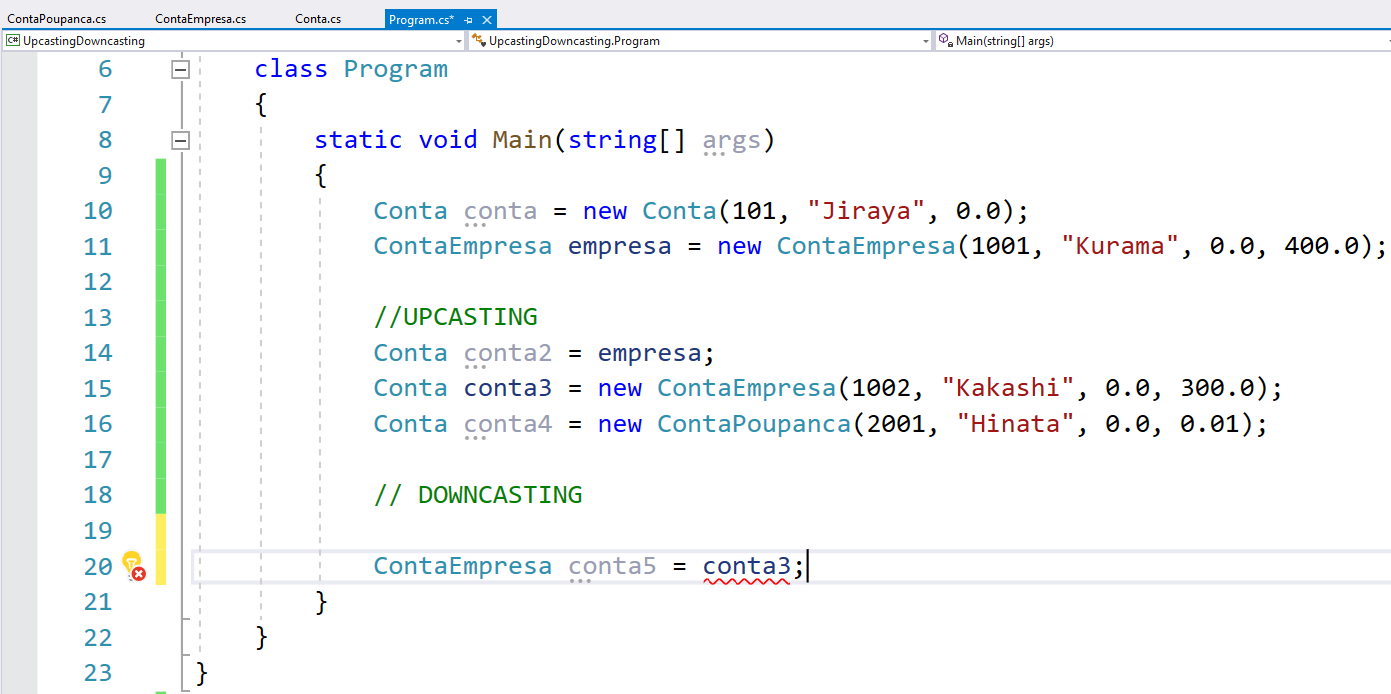
Outro exemplo.



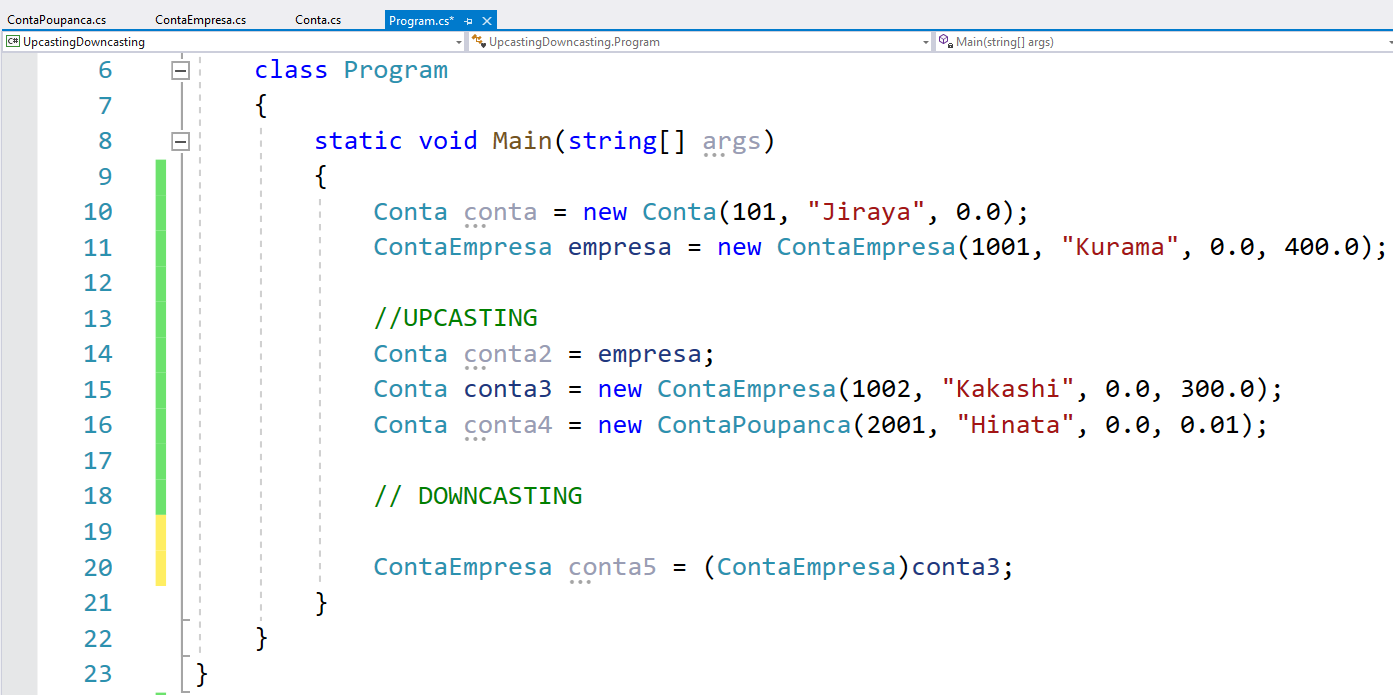
# DownCasting

Converter uma **superclasse** em uma **subclasse**.

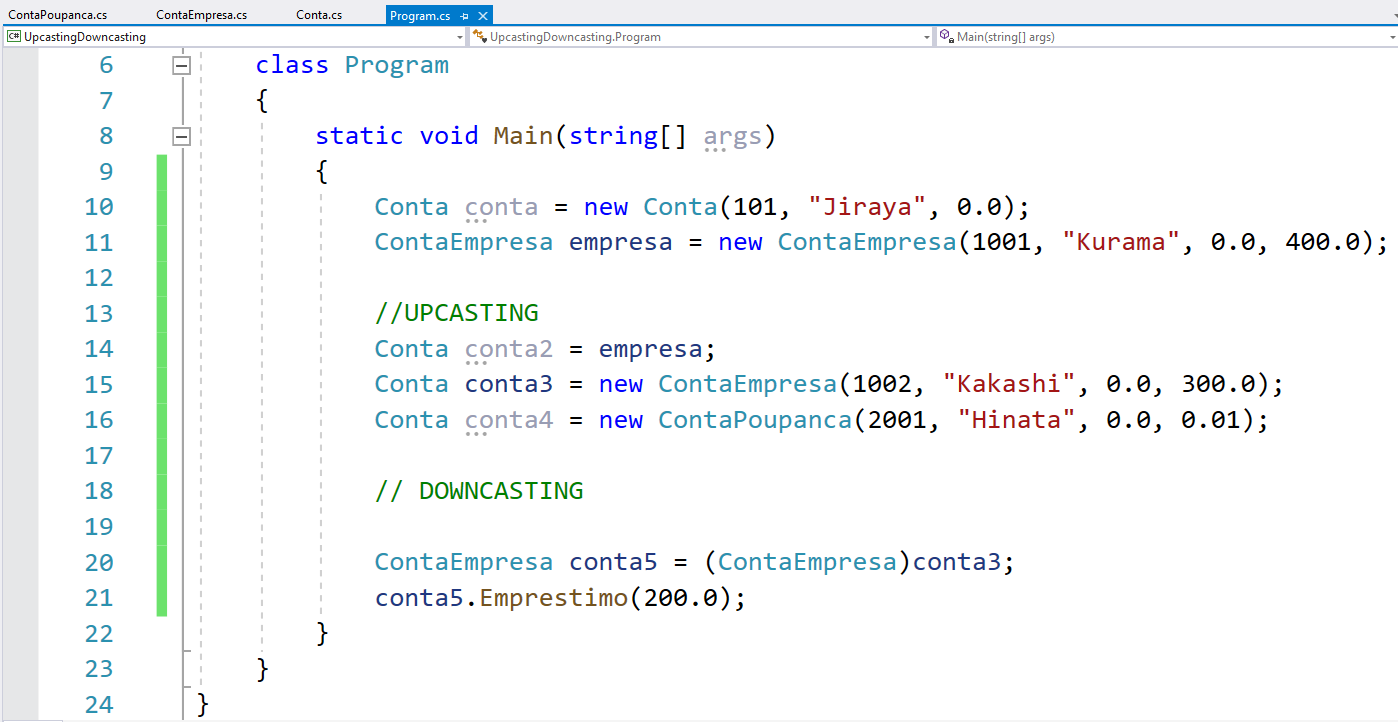
Se tentarmos fazer uma subclasse receber uma superclasse, o compilador não permite.



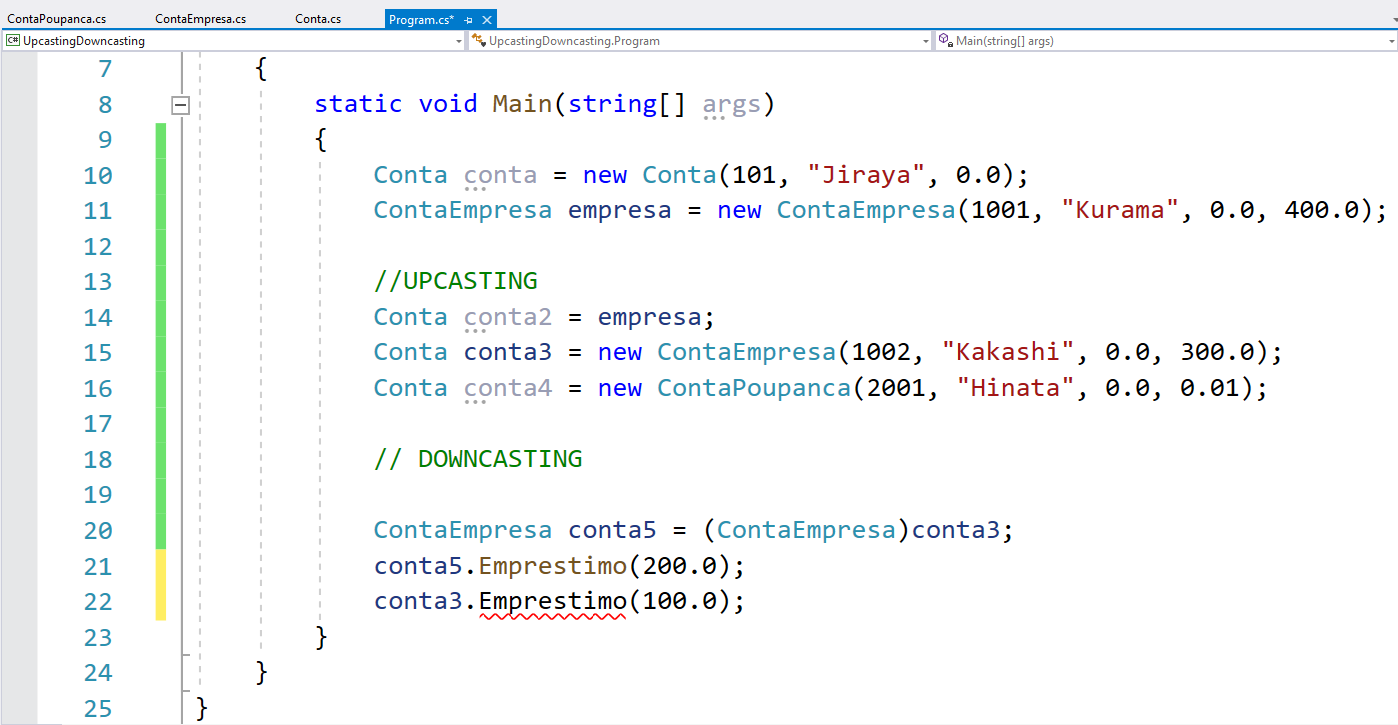
Para realizar essa conversão, precisamos fazer um casting (conversão).



Podemos solicitar um empréstimo.



Observe, que não conseguimos realizar um empréstimo para a conta3, pois ela não foi convertida.



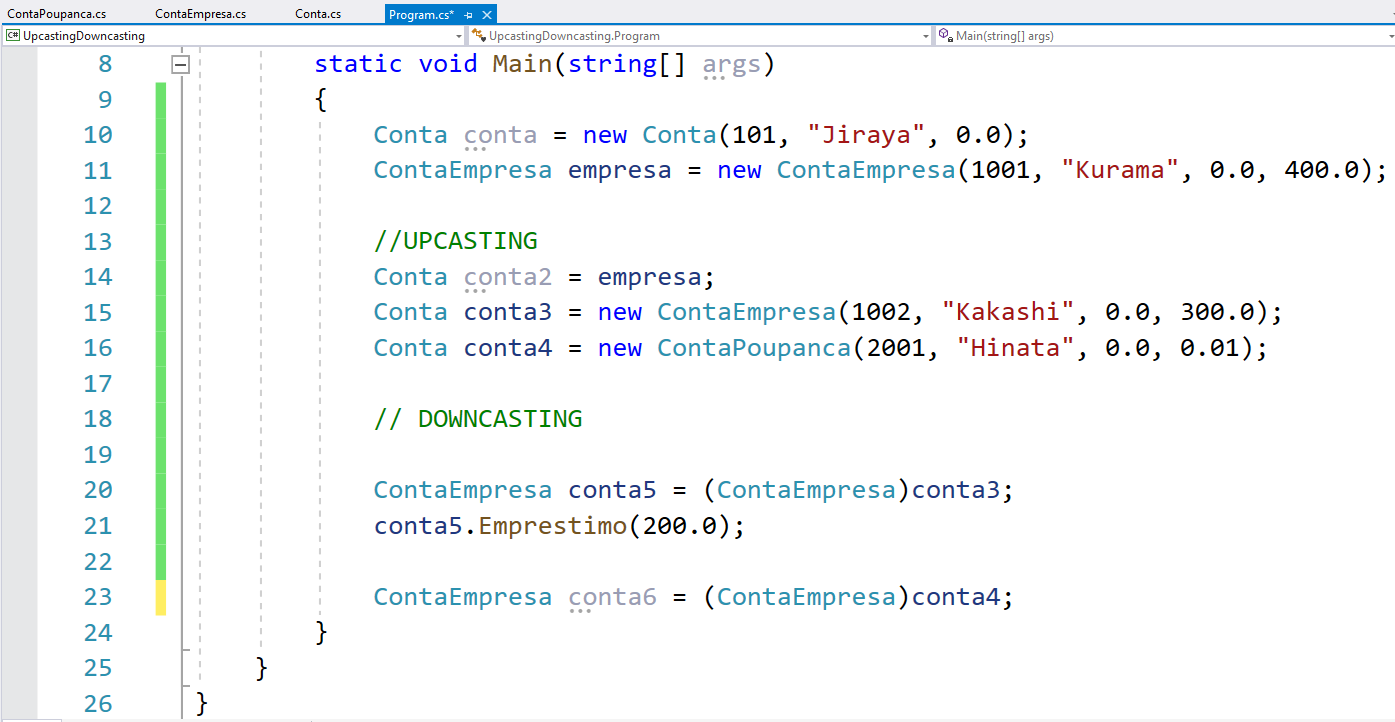
Ou seja, para poder fazer um empréstimo, precisamos fazer um Dowcasting primeiro.

Outro teste. Vamos tentar atribuir uma variável do tipo ContaPoupanca em uma ContaEmpresa.

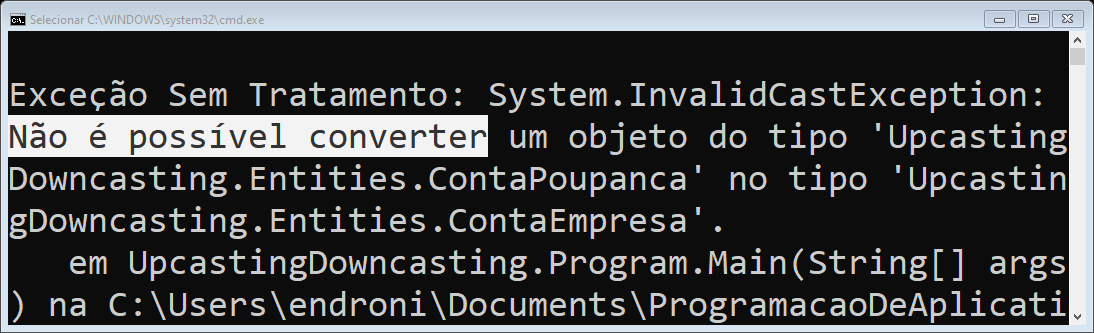


**Atenção!**

Convertendo, aceita, porém...

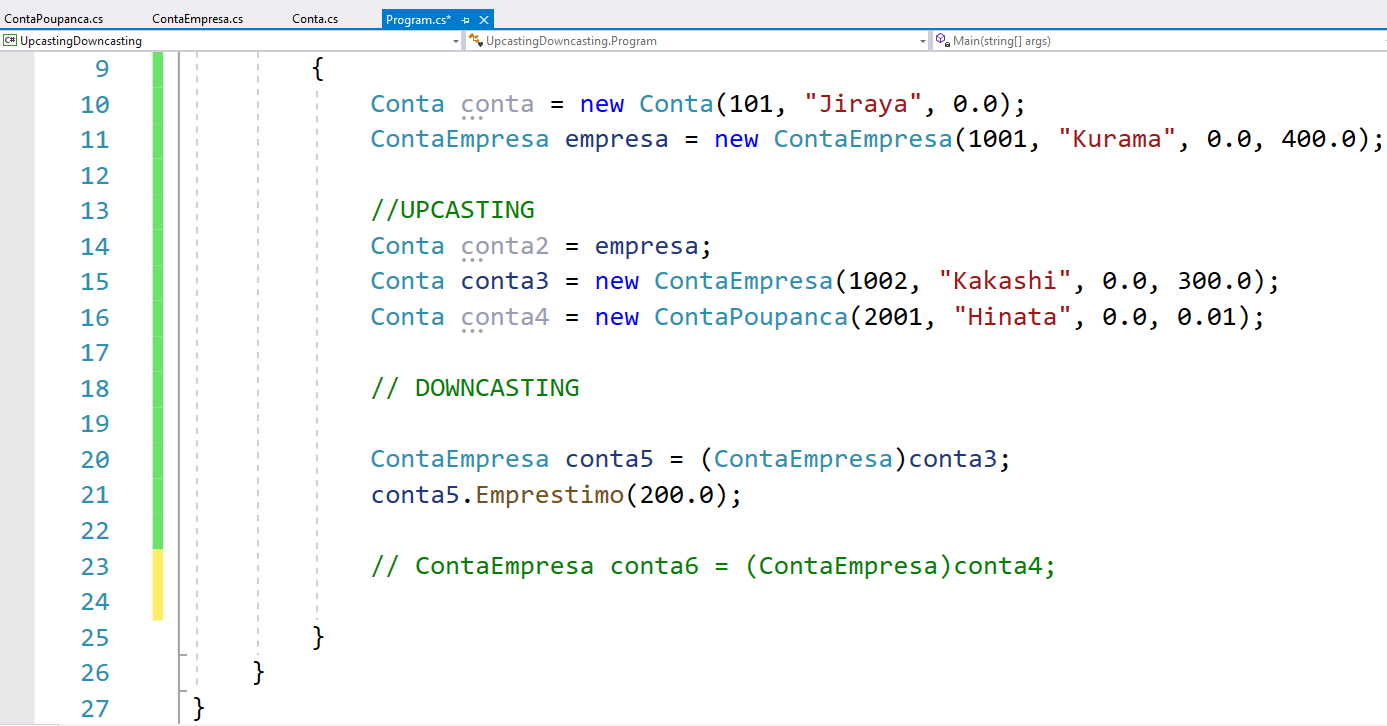


Porém essas classes não são compatíveis. Por isso, o compilador não acusa erro, mas ao compilar ele informa erro.



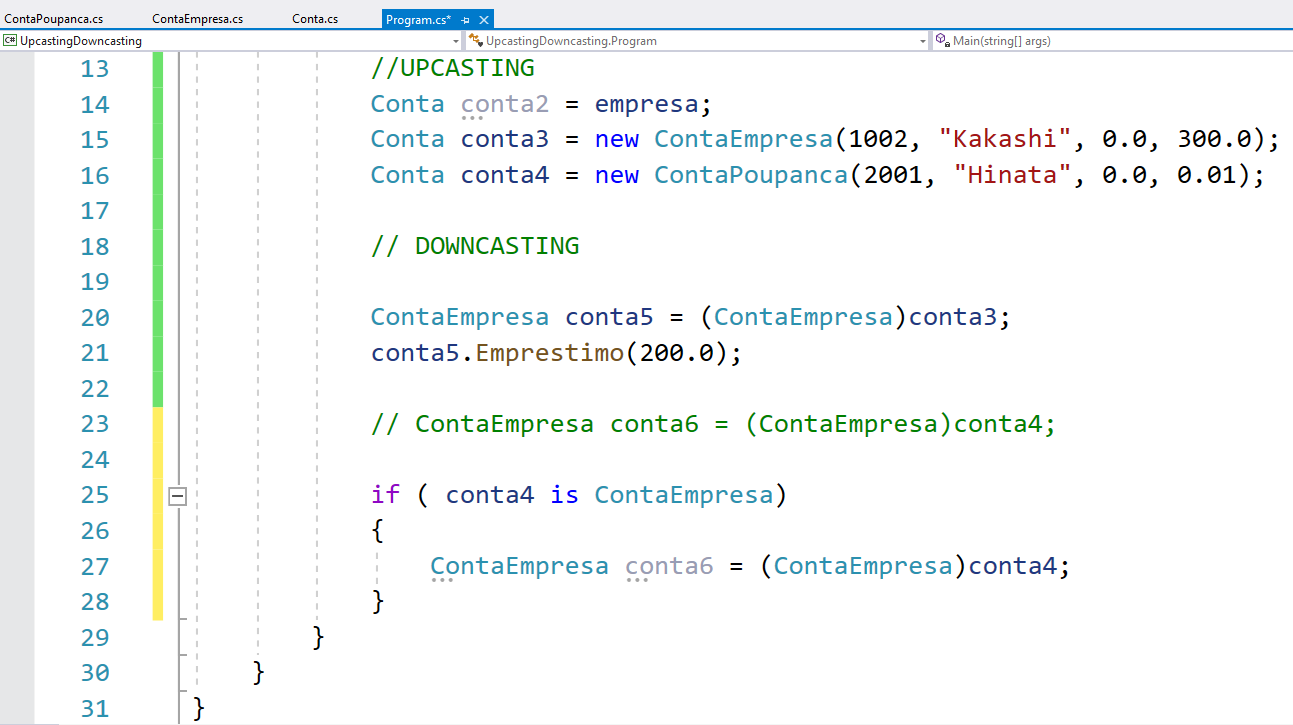
O Downcasting é uma operação insegura. Só deve ser utilizado em caso de total certeza.

Vamos comentar a linha 23 para ela não dar mais erro.

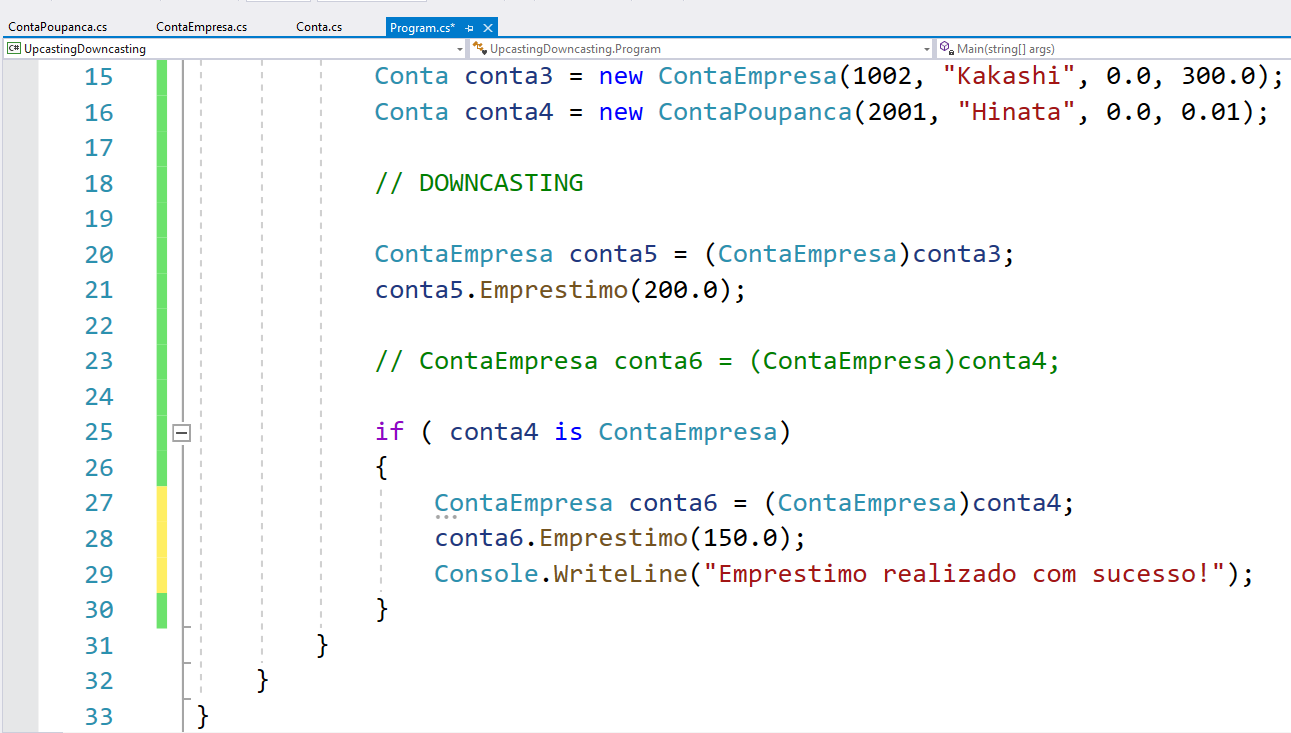


# is

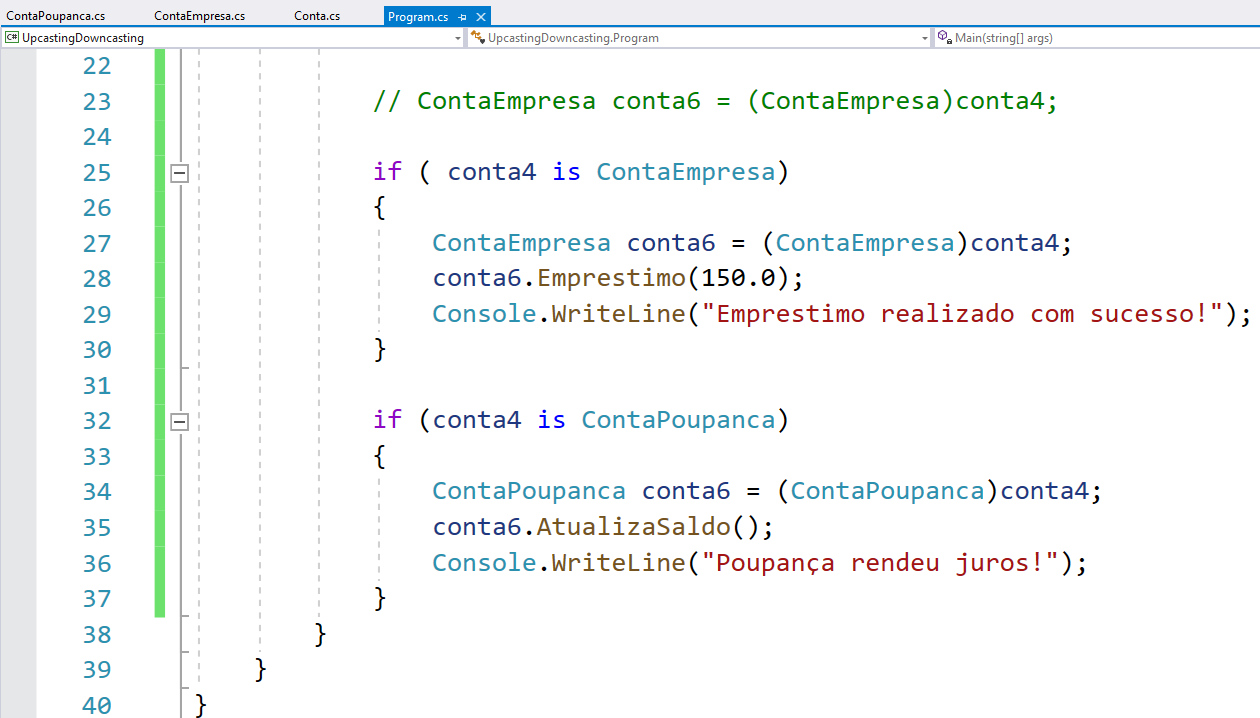
Para correção, precisamos utilizar a palavra **is** para testar a condição antes de realizar a conversão.



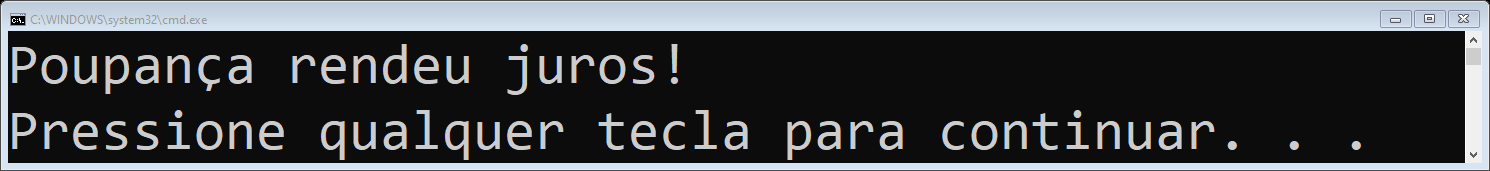
Vamos implementar esse empréstimo. Vamos imprimir uma mensagem de empréstimo realizado com sucesso, caso as contas sejam compatíveis.



Agora vamos fazer um outro teste para saber se a conta4 é uma instancia de uma ContaPoupanca. Se for verdadeiro, ela irá render juros e atualizar o sado da conta.



Compilando com Ctrl + F5



Concluindo. O primeiro if falhou, porque ele não é uma instancia de ContaEmpresa, mas o segundo if passou, porque ele realmente é uma instancia de ContaPoupanca e portanto foi feito um casting(conversão) de tipo. Para ser mais preciso, realizou um Downcasting.

# as

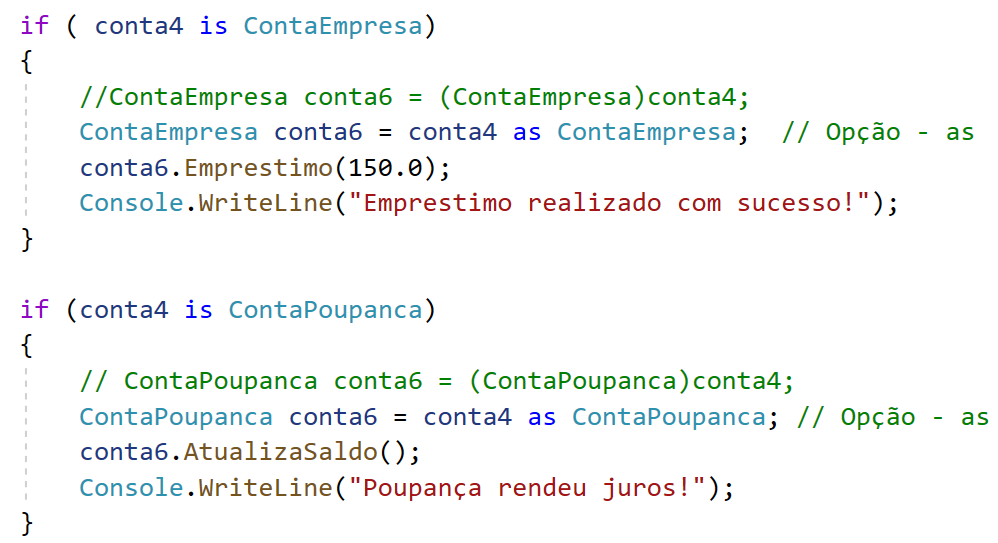
Uma outra forma de se fazer o Downcasting é utilizando a palavra chave **as**.



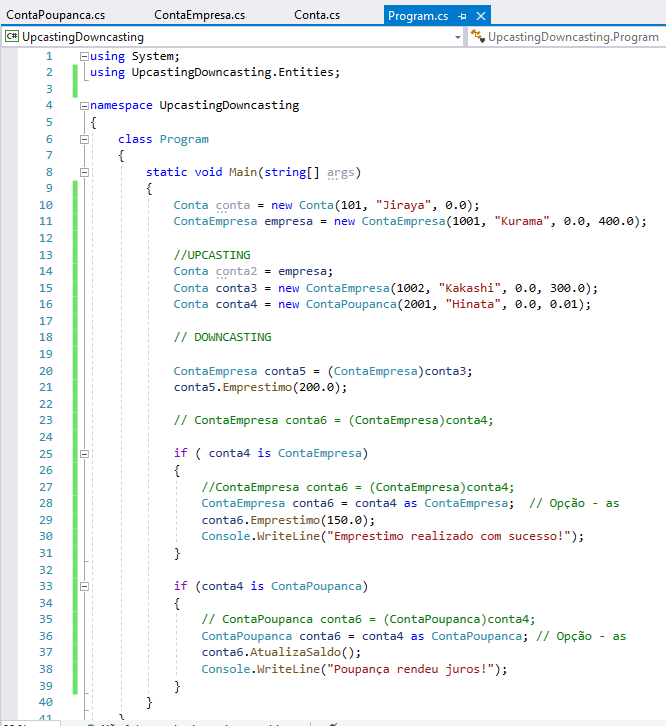
Opção comum:



Utilizando a sintaxe alternativa as



Programa principal:



# Códigos

## Código classe Conta:

namespace UpcastingDowncasting.Entities

{

class Conta

{

public int Numero { get; private set; }

public string Titular { get; private set; }

public double Saldo { get; protected set; }

public Conta()

{

}

public Conta(int numero, string titular, double saldo)

{

Numero = numero;

Titular = titular;

Saldo = saldo;

}

public void Saque(double valor)

{

Saldo -= valor;

}

public void Deposito(double valor)

{

Saldo += valor;

}

}

}

## Código classe ContaEmpresa:

namespace UpcastingDowncasting.Entities

{

class ContaEmpresa : Conta

{

public double LimiteEmprestimo { get; set; }

public ContaEmpresa()

{

}

public ContaEmpresa(int numero, string titular, double saldo, double limiteEmprestimo)

: base(numero, titular, saldo)

{

LimiteEmprestimo = limiteEmprestimo;

}

public void Emprestimo(double valor)

{

if (valor <= LimiteEmprestimo)

{

Saldo += valor;

}

}

}

}

## Código classe ContaPoupanca:

namespace UpcastingDowncasting.Entities

{

class ContaPoupanca : Conta

{

public double TaxaJuros { get; set; }

public ContaPoupanca()

{

}

public ContaPoupanca(int numero, string titular, double saldo, double taxaJuros)

: base(numero, titular, saldo)

{

TaxaJuros = taxaJuros;

}

public void AtualizaSaldo()

{

Saldo += Saldo \* TaxaJuros;

}

}

}

## Código programa principal:

using System;

using UpcastingDowncasting.Entities;

namespace UpcastingDowncasting

{

class Program

{

static void Main(string[] args)

{

Conta conta = new Conta(101, "Jiraya", 0.0);

ContaEmpresa empresa = new ContaEmpresa(1001, "Kurama", 0.0, 400.0);

//UPCASTING

Conta conta2 = empresa;

Conta conta3 = new ContaEmpresa(1002, "Kakashi", 0.0, 300.0);

Conta conta4 = new ContaPoupanca(2001, "Hinata", 0.0, 0.01);

// DOWNCASTING

ContaEmpresa conta5 = (ContaEmpresa)conta3;

conta5.Emprestimo(200.0);

// ContaEmpresa conta6 = (ContaEmpresa)conta4;

if ( conta4 is ContaEmpresa)

{

//ContaEmpresa conta6 = (ContaEmpresa)conta4;

ContaEmpresa conta6 = conta4 as ContaEmpresa; // Opção - as

conta6.Emprestimo(150.0);

Console.WriteLine("Emprestimo realizado com sucesso!");

}

if (conta4 is ContaPoupanca)

{

// ContaPoupanca conta6 = (ContaPoupanca)conta4;

ContaPoupanca conta6 = conta4 as ContaPoupanca; // Opção - as

conta6.AtualizaSaldo();

Console.WriteLine("Poupança rendeu juros!");

}

}

}

}

# GitHub

O código está na subpasta **\Aula\_08\_Classe\UpcastingDowncasting**

https://github.com/endroni/ProgramacaoDeAplicativos.git

# Referência:

Social msdn Microsoft – Upcasting and Downcasting – < https://social.msdn.microsoft.com/Forums/en-US/b05eb341-2657-495c-bfad-2ddd536175b7/upcasting-and-downcasting?forum=csharpgeneral> Acessado dia 11 de dezembro de 2020.