# Abstração – abstract 21/12/2020

# Classes Abstratas

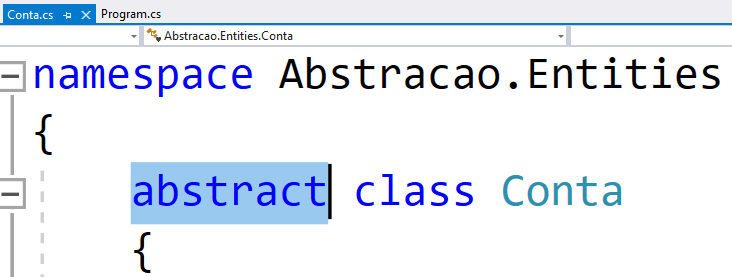
São classes que não podem ser instanciadas

É uma forma de garantir herança total: somente subclasses não abstratas podem ser instanciadas, mas nunca a superclasse abstrata.

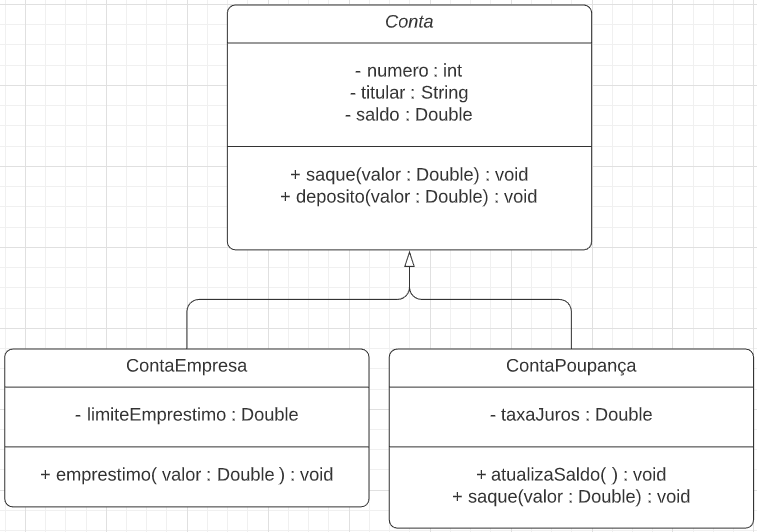
Exemplo:

Suponha que em um negócio relacionado a banco, apenas contas poupança e contas para empresas são permitidas. Não existe conta comum.

Para garantir que contas comuns não possam ser instanciadas, basta acrescentarmos a palavra "abstract" na declaração da classe.



Notação UML: itálico



Vamos para a prática:

## Sugestão:

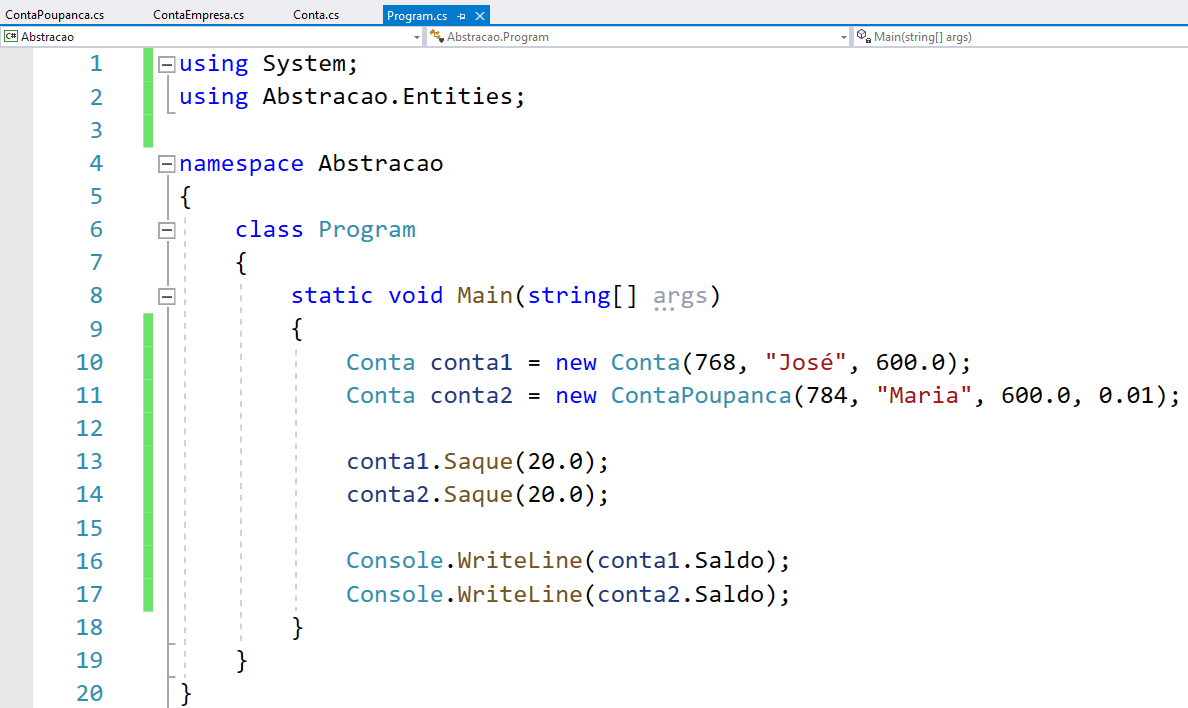
Baixar a solução no link do GitHub:

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

Usar o programa da subpasta:

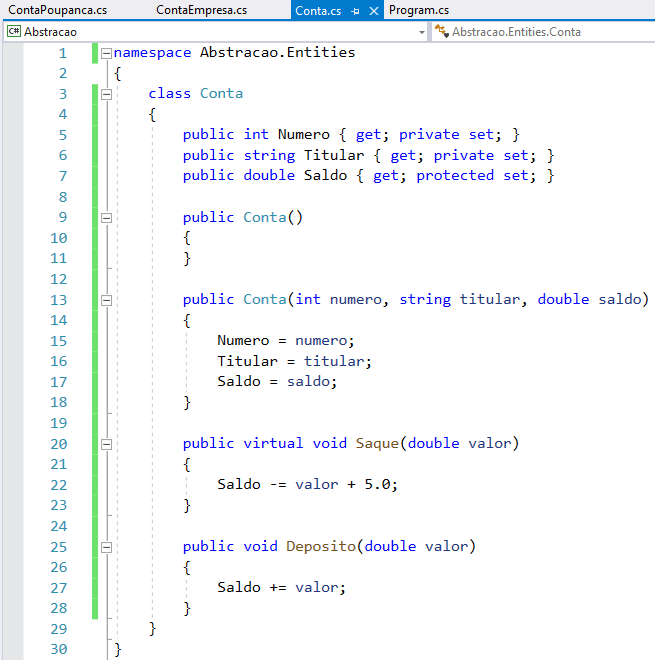
* \Aula\_08\_Classe\SobreposicaoVirtualOverrideBase

Observe o program:

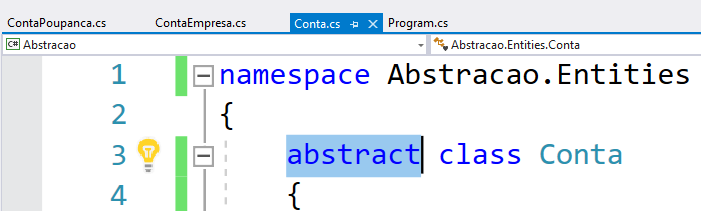


Está sendo realizado um saque em uma Conta comum e em uma ContaPoupanca.

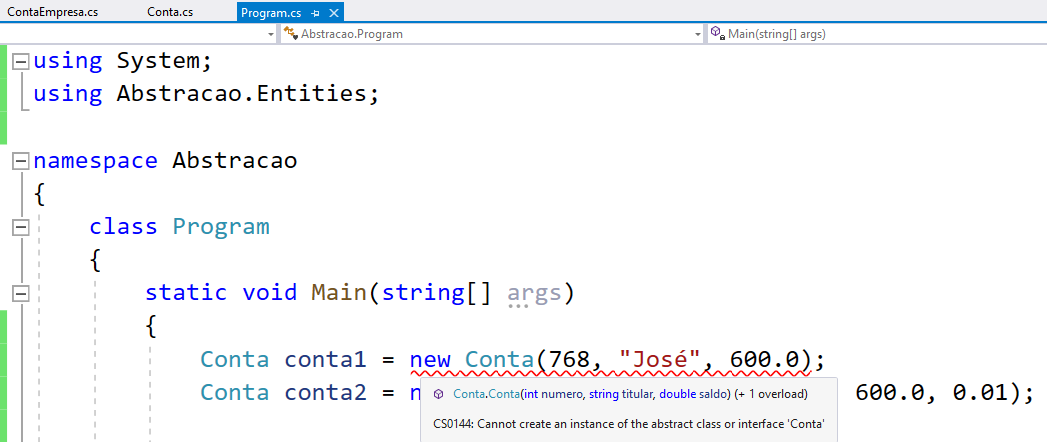
A superclasse Conta está assim:



Agora, vamos dizer que a superclasse Conta será abstrata.



Observe o que aconteceu no Program.cs



Está dizendo que não pode criar uma instancia de uma classe abstrata.

## Questionamento

Se a classe Conta não pode ser instanciada, por que simplesmente não criar somente ContaPoupanca e ContaEmpresa?

Resposta:

* **Reuso:** para não ter que reescrever todos os atributos e métodos nas duas subclasses. A classe abstrata vai permitir isso. Obs. Códigos replicados, aumenta a possibilidade de erros de programação, pois o desenvolvedor ao alterar algum trecho de código terá que refazer essa ação nas outras classes.
* **Polimorfismo**: a superclasse classe genérica nos permite tratar de forma fácil e uniforme todos os tipos de conta, inclusive com polimorfismo se for o caso (como fizemos nos últimos exercícios). Por exemplo, você pode colocar todos tipos de contas em uma mesma coleção.

## Demonstração:

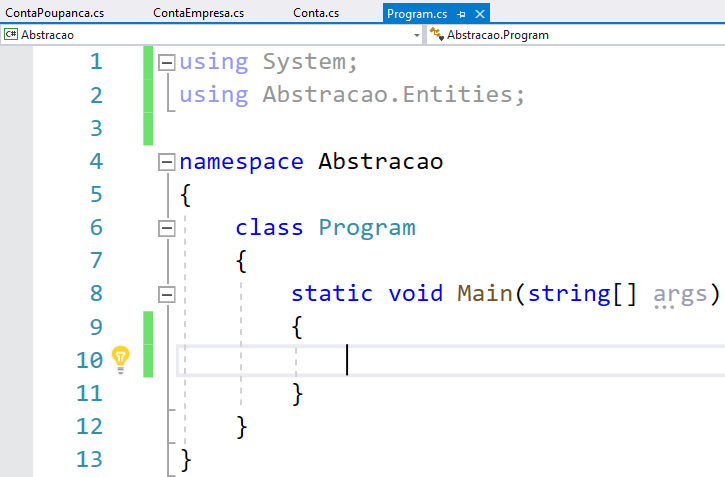
A ideia é utilizar lista para instanciar a classe.

Suponha que você queira:

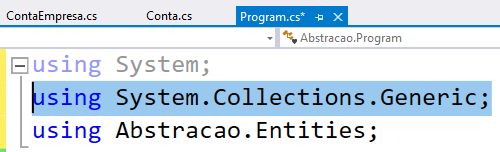
Totalizar o saldo de todas as contas.

Sacar 10.00 de todas as contas.

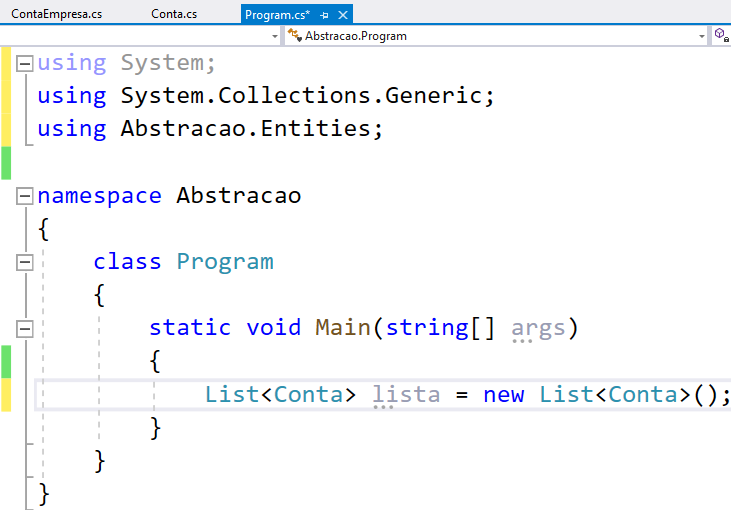
No Program.cs, apague o código para realizarmos testes.



Vamos acrescentar um using

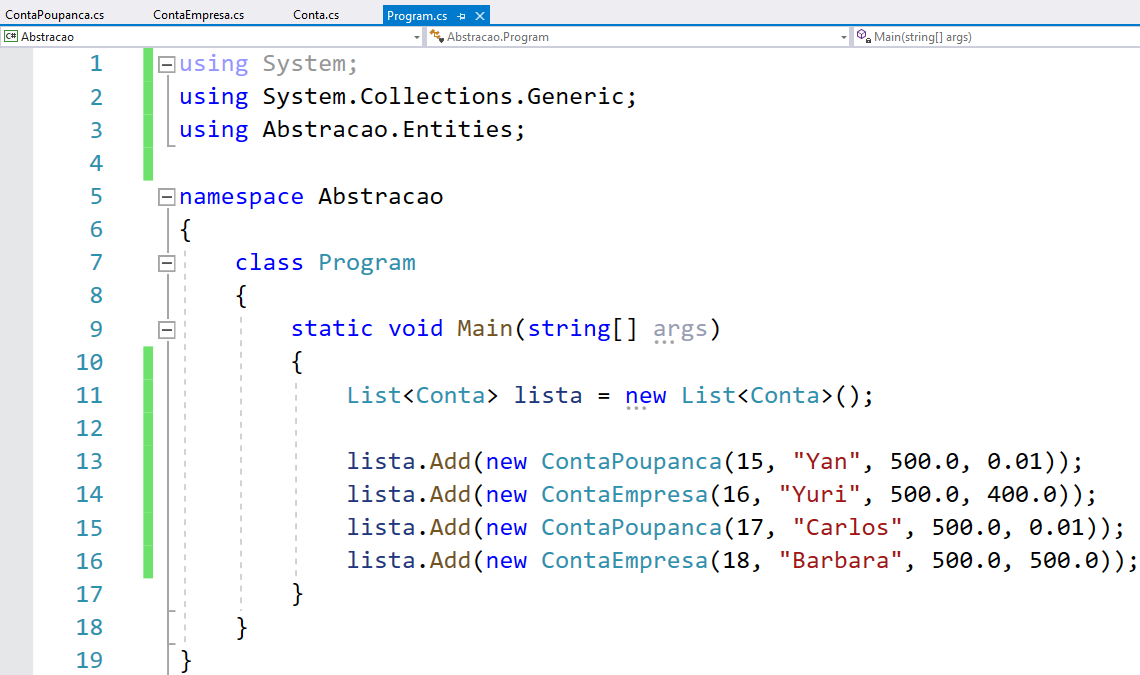


Crie uma lista do tipo Conta

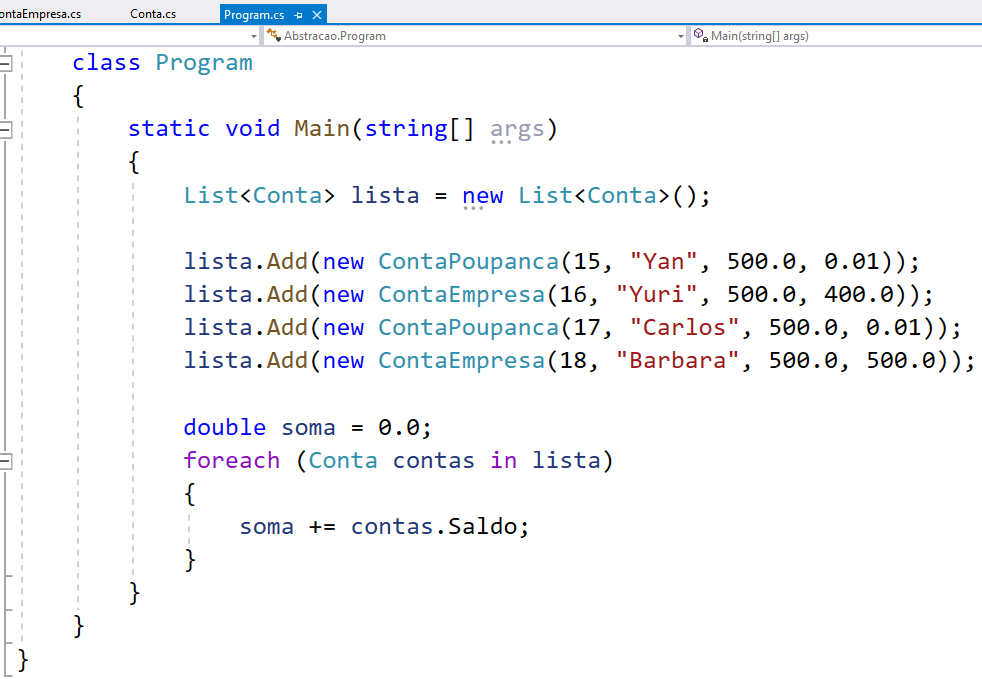


Uma classe abstrata também pode ser utilizada como uma lista.

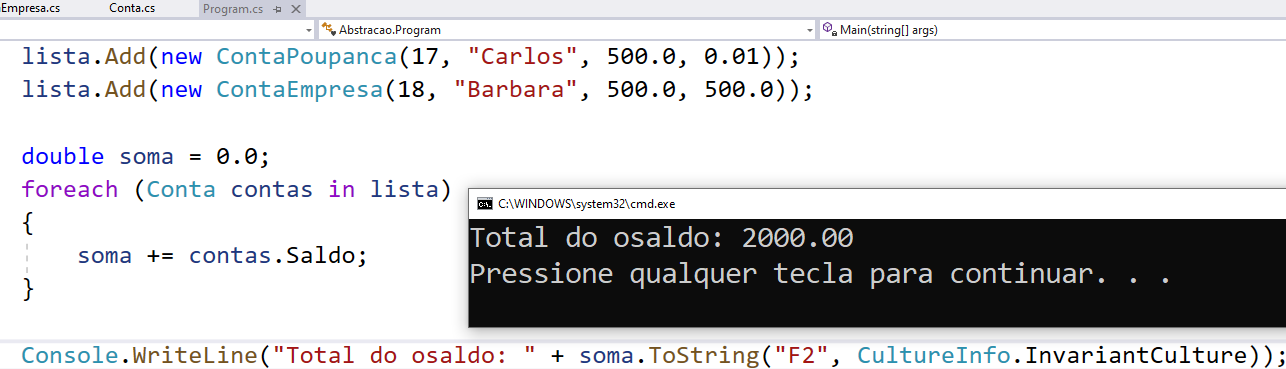
Vamos adicionar alguns elementos nessa lista.



Para saber o saldo total das contas, basta percorrer a lista realizando a soma.

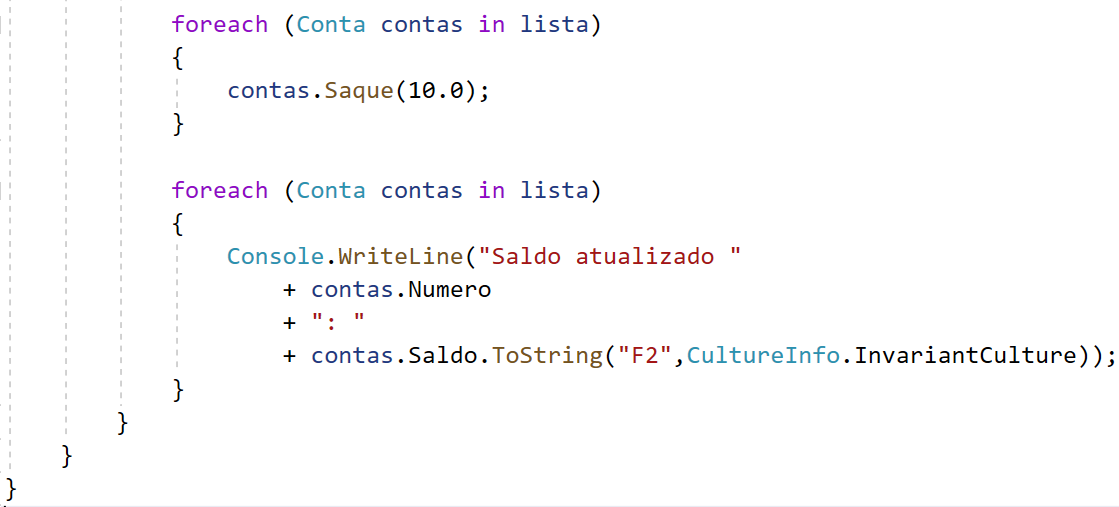


Só mandar exibir agora:

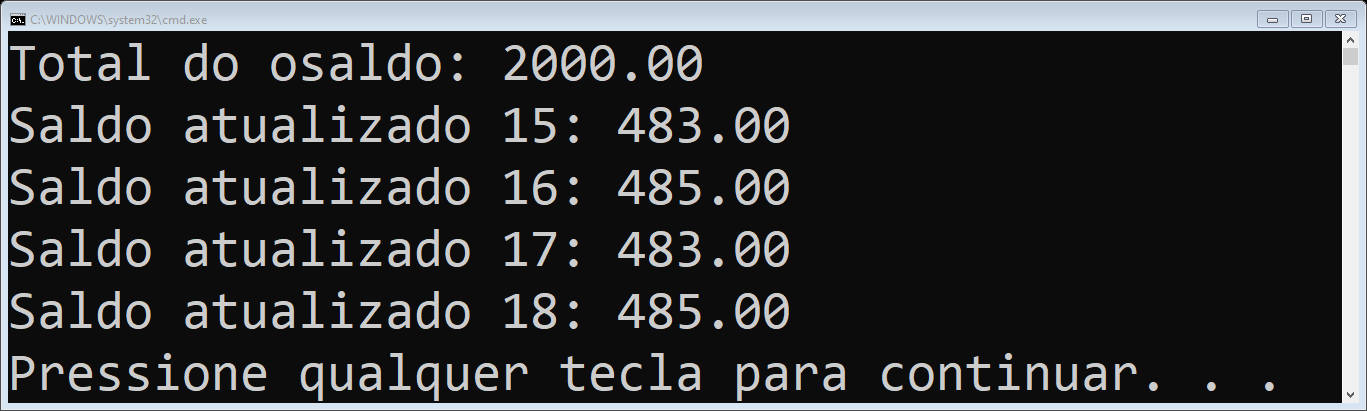


Podemos também, usar métodos sendo executados de forma polimórfica.

Vamos sacar 10.0 em cada conta percorrendo a lista.



Observe o resultado:



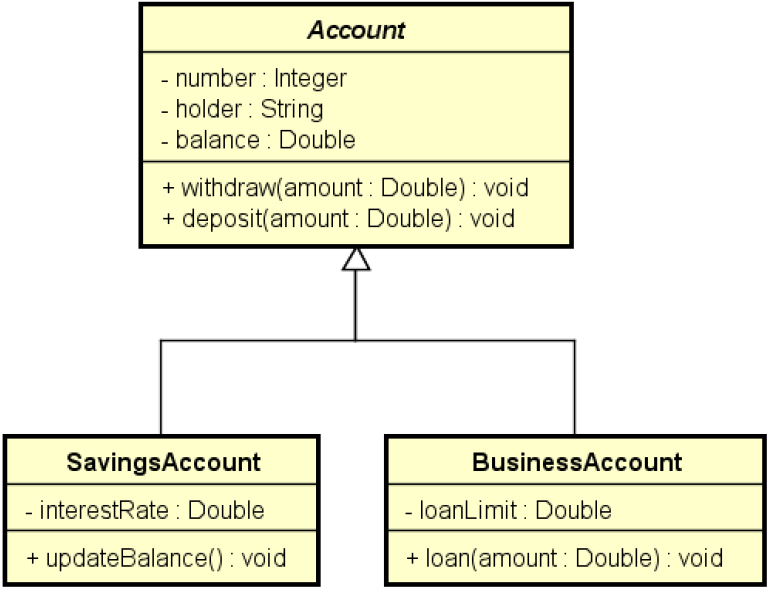
Todos estavam com o saldo de 500.00

Quando ContaEmpresa, foi incluído a taxa de 5.0

Quando ContaPoupanca, foi incluído a taxa de 5.0 + 2.0

Ou seja, estamos utilizando polimorfismo em uma classe abstrata.

Modelo em inglês.



# Métodos Abstratos

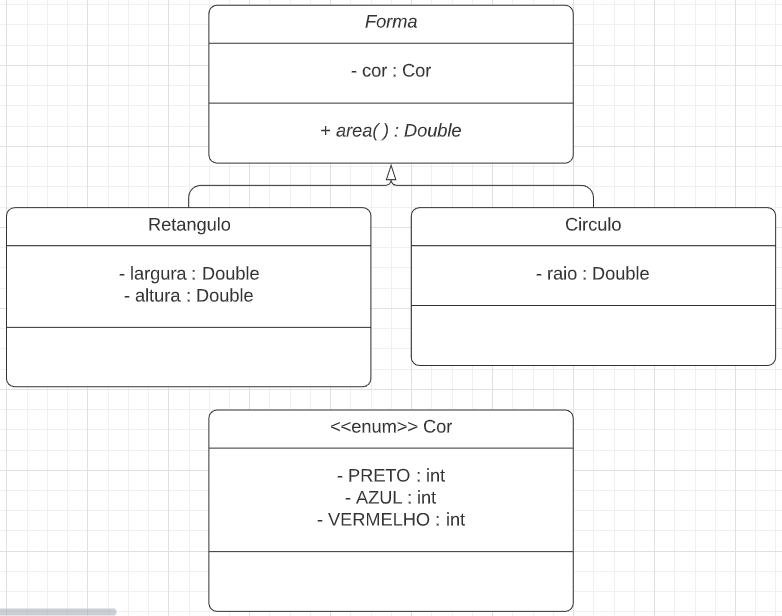
São métodos que não possuem implementação.

Métodos precisam ser abstratos quando a classe é genérica demais para conter sua implementação.

Se uma classe possuir pelo menos um método abstrato, então esta classe também é abstrata.

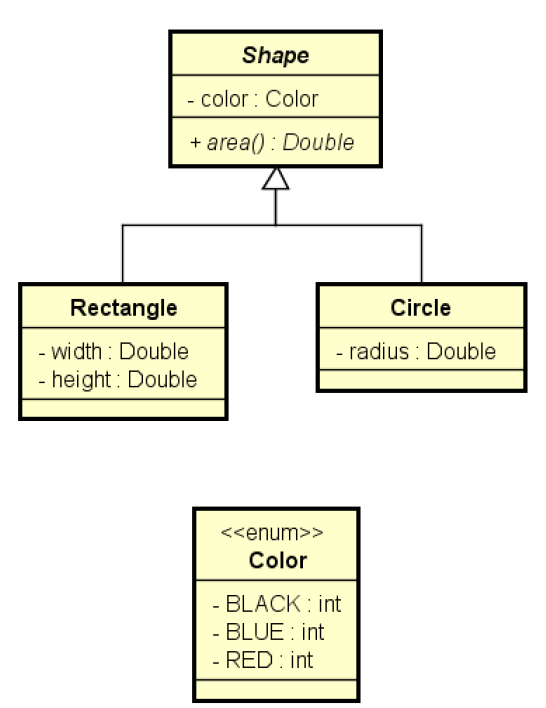
Notação UML: itálico (método *area* e a classe *Forma*)

Exemplo: A forma de se calcular a área de um triangulo, é multiplicando a largura pela altura. Porém, a forma de se calcular área de um círculo, é (A = π r²). Portanto, a área da forma, depende de cada tipo, se é um retângulo ou um círculo. Não sendo possível assim, calcular a área sem saber sua Forma. A classe Forma é muito genérica.



No modelo acima, não podemos colocar a área nas subclasses Retangulo e Circulo, devido ao polimorfismo.

Modelo em inglês.



# GitHub

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

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

# Referência:

Docs Microsoft – Herança em C# .NET – < https://docs.microsoft.com/pt-br/dotnet/csharp/tutorials/inheritance > Acessado dia 10 de dezembro de 2020.

Tutorial W3Schools – Inheritance - < https://www.w3schools.com/cs/cs\_inheritance.asp > Acessado dia 10 de dezembro de 2020.