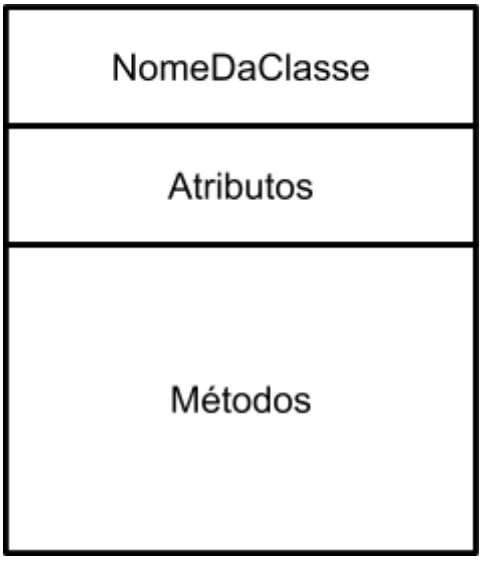
Padrão de Diagrama de Classes

Especificação de uma classe

No nosso diagrama, cada classe está representada por um retângulo que conterá três divisões:

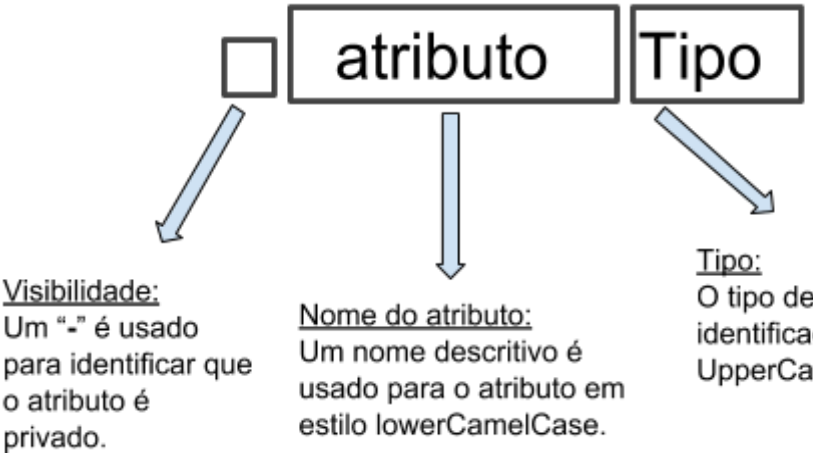
- ✓ Nome da classe
- ✓ Atributos
- ✓ Métodos



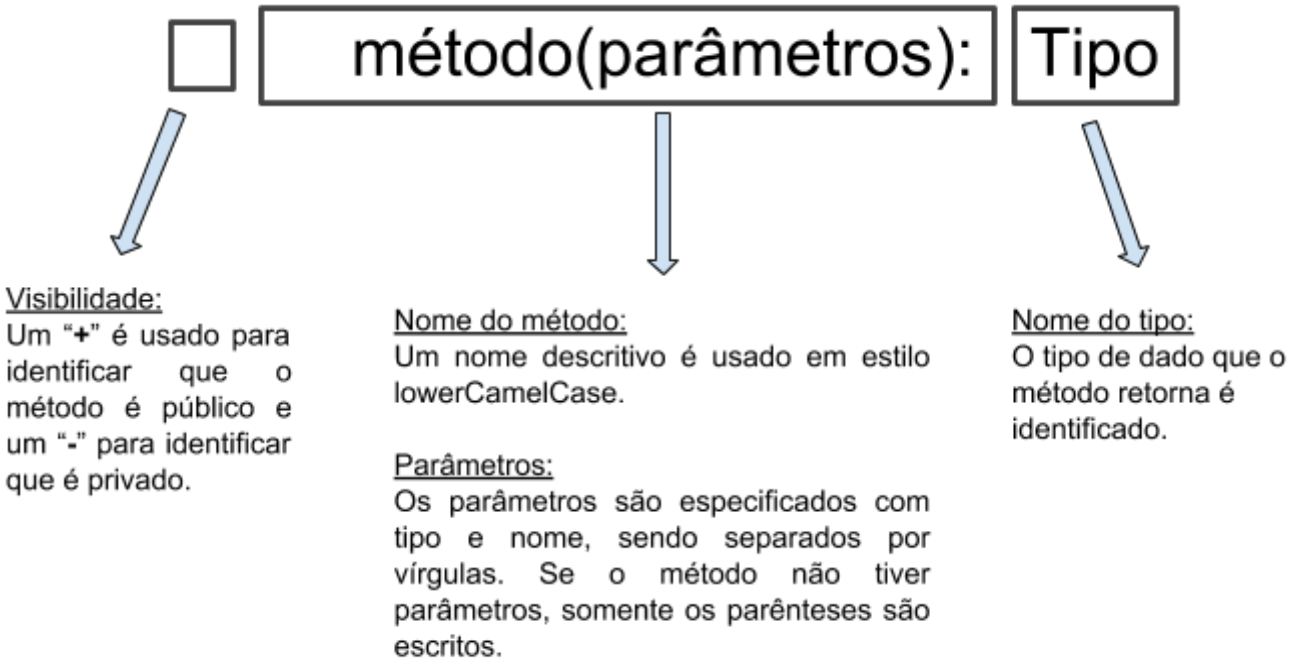
- **<u>Especificação do nome de uma classe:</u>** um nome descritivo deverá ser escolhido para a classe que estiver sendo diagramada. O nome da classe sempre começa com letra maiúscula. Para nomes compostos, escrevemos em UpperCamelCase[1].

[1] **CamelCase** é uma forma de escrever que se aplica a frases ou palavras compostas.
UpperCamelCase é quando a primeira letra de cada uma das palavras é maiúscula.
Exemplo: *ExemploDeUpperCamelCase.*
lowerCamelCase é igual à forma anterior, mas com a primeira letra em minúsculo.
Exemplo: *exemploDeLowerCamelCase*

- **Especificação dos atributos:**

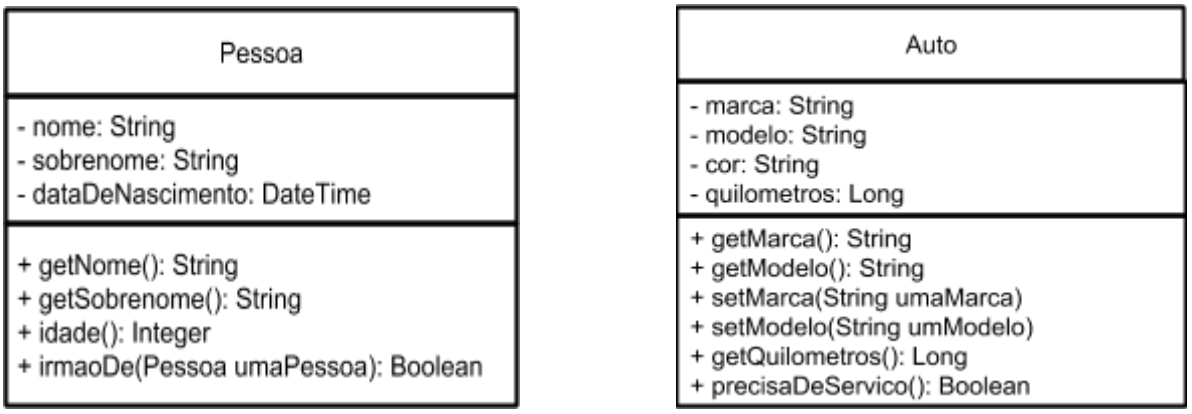


- **Especificação dos métodos:**



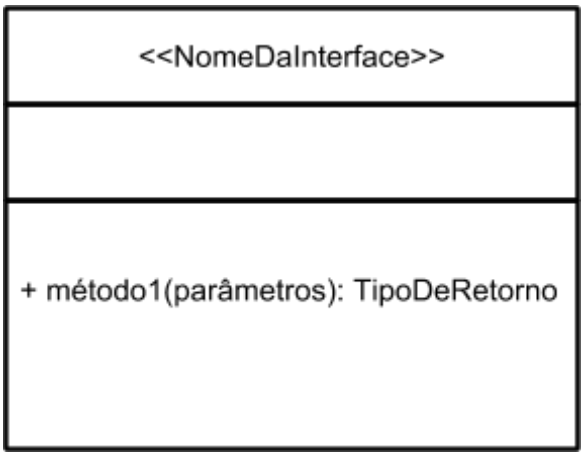
DigitalHouse >
Coding School

Exemplos



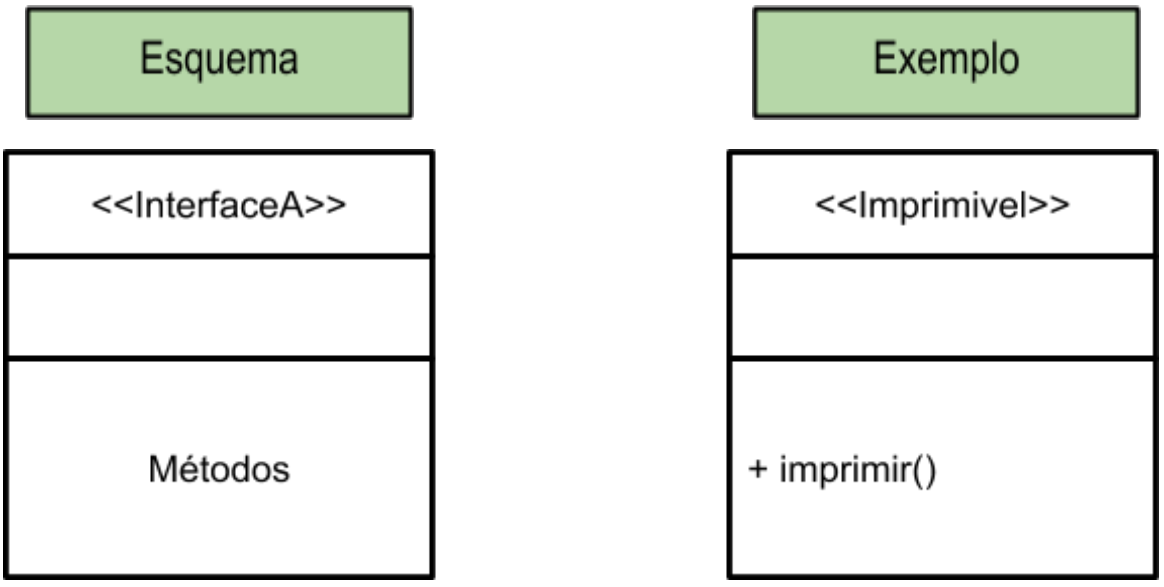
Especificação de uma Interface

No nosso diagrama, uma interface é especificada de forma semelhante a uma classe, mas sem atributos:



- **<u>Especificação do nome de uma interface:</u>** um nome descritivo deverá ser escolhido para a interface que estiver sendo diagramada. O estilo usado é o UpperCamelCase, igual ao nome de uma classe. Além disso, usamos os símbolos << >> para rodear o nome (ver esquema e exemplo).

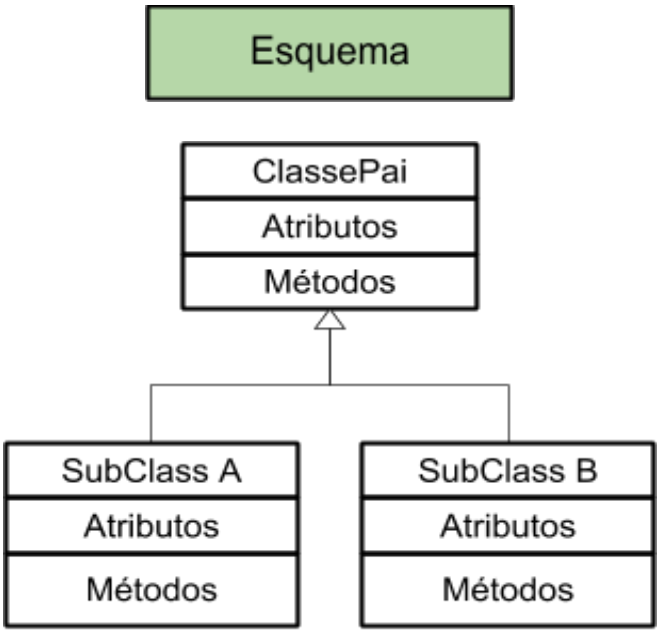
- **Especificação dos métodos de uma interface:** os métodos em uma interface são públicos por definição, portanto sua visibilidade sempre será "**+**".

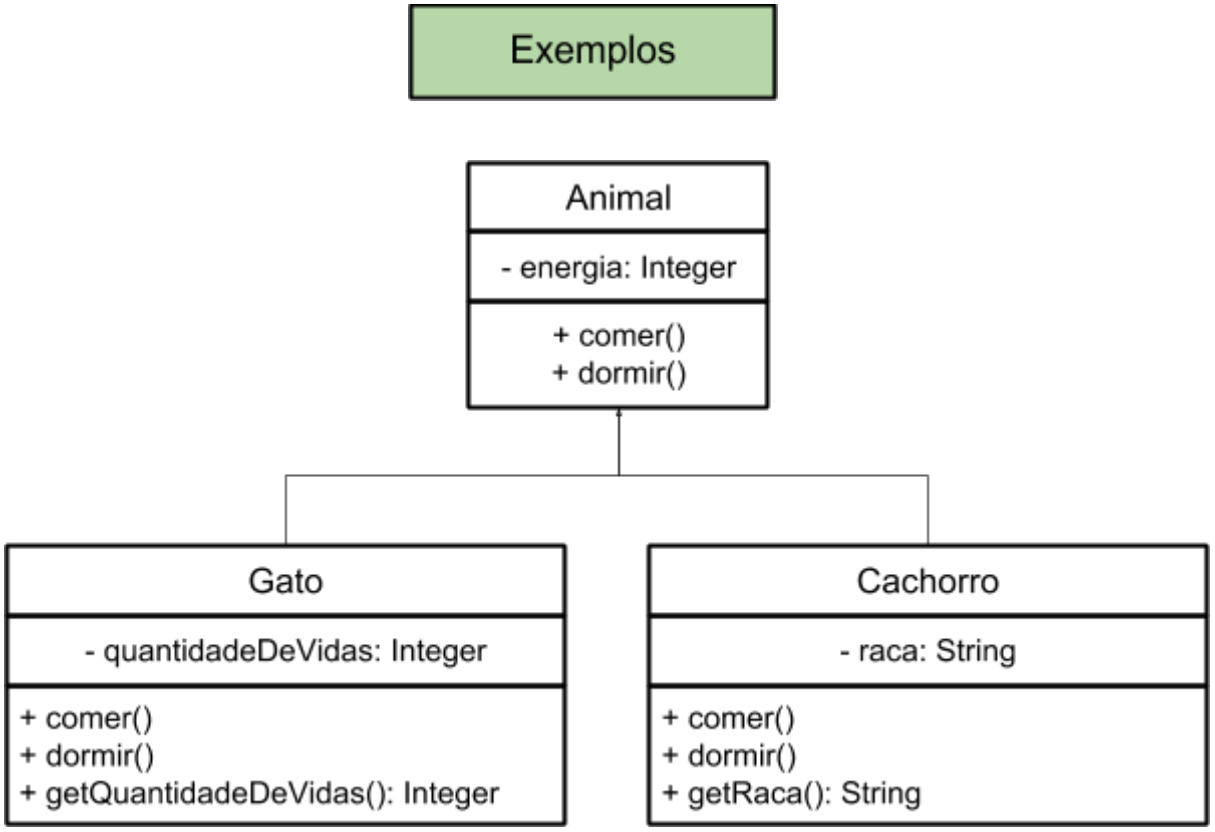


Especificação dos relacionamentos entre classes:

Abaixo, vamos descrever como especificar, de acordo com o nosso critério, os diversos relacionamentos que podem existir entre as classes.

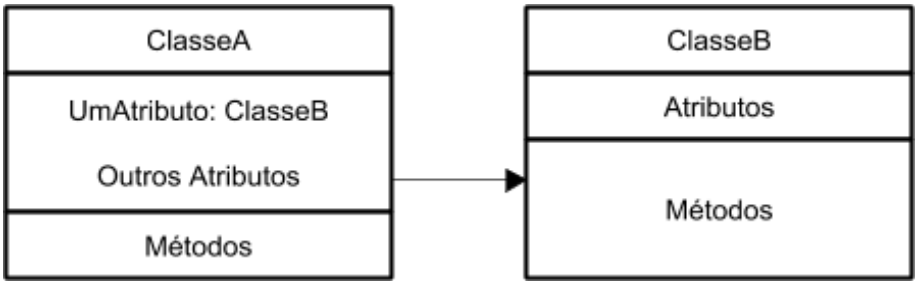
- **Especificação do relacionamento de herança:** uma classe pode ser uma única classe pai. A subclasse herda os atributos e os métodos de sua classe pai.



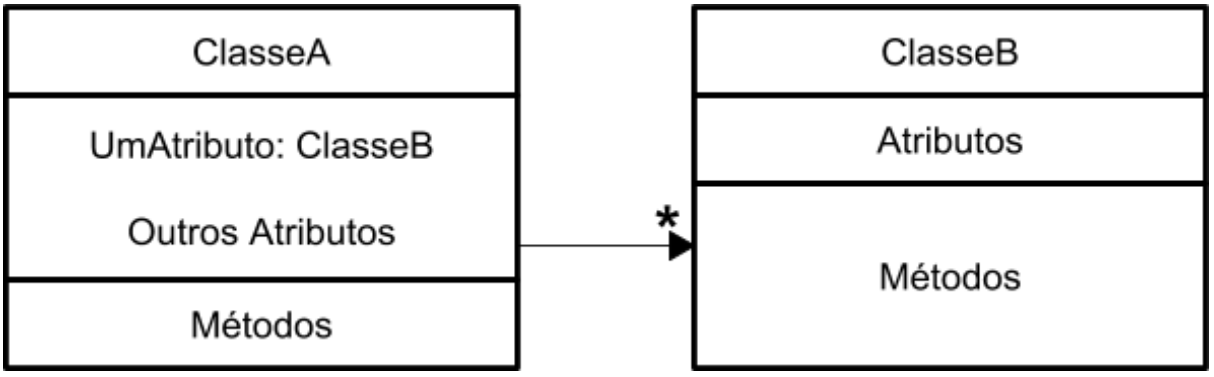


- **<u>Especificação do relacionamento de associação:</u>** O relacionamento de associação é estabelecido quando um objeto de uma classe colabora com um ou mais objetos de outra classe. Na UML padrão, existem diversos tipos de relacionamentos de associação e composição, mas vamos especificar somente uma. No entanto, vamos especificar a paridade do relacionamento; ou seja, com quantos objetos da outra classe ela se relaciona.

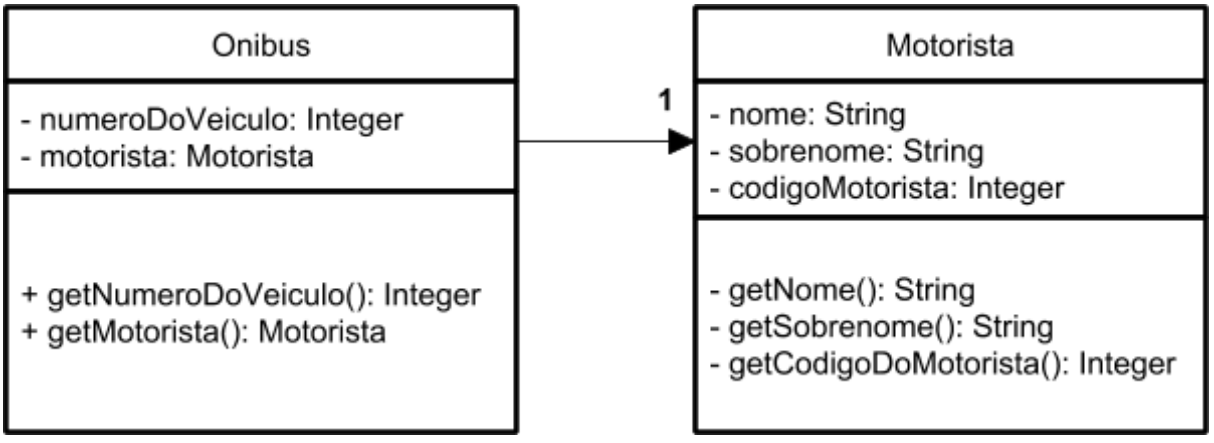
Neste diagrama, vamos dizer que a **ClasseA** conhece **uma instância** da **ClasseB**. No entanto, devemos levar em consideração que, neste caso, a **ClasseB** não conhece a **ClasseA**.

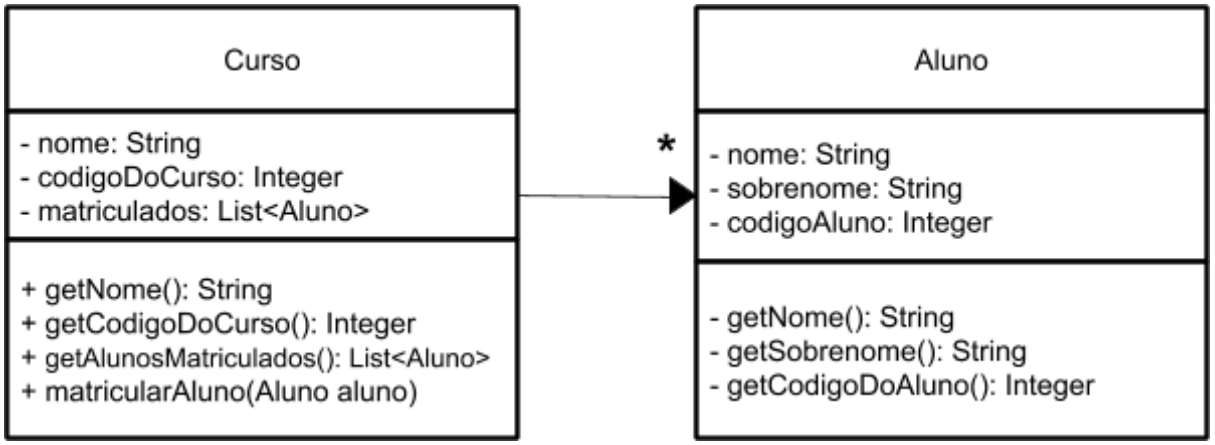


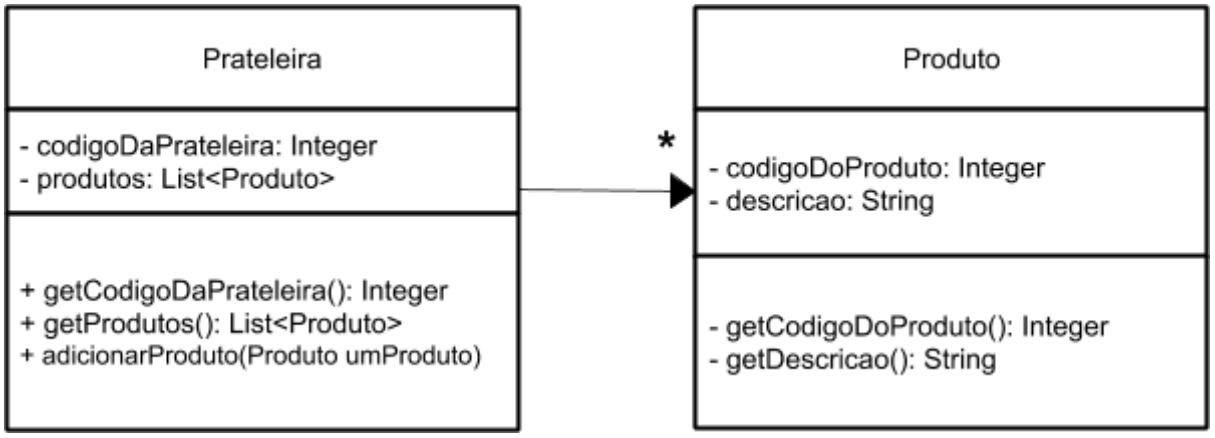
Neste diagrama, vamos dizer que a **ClasseA** conhece **várias instâncias** da **ClasseB**. No entanto, devemos levar em consideração que, neste caso, a **ClasseB** não conhece a **ClasseA**.



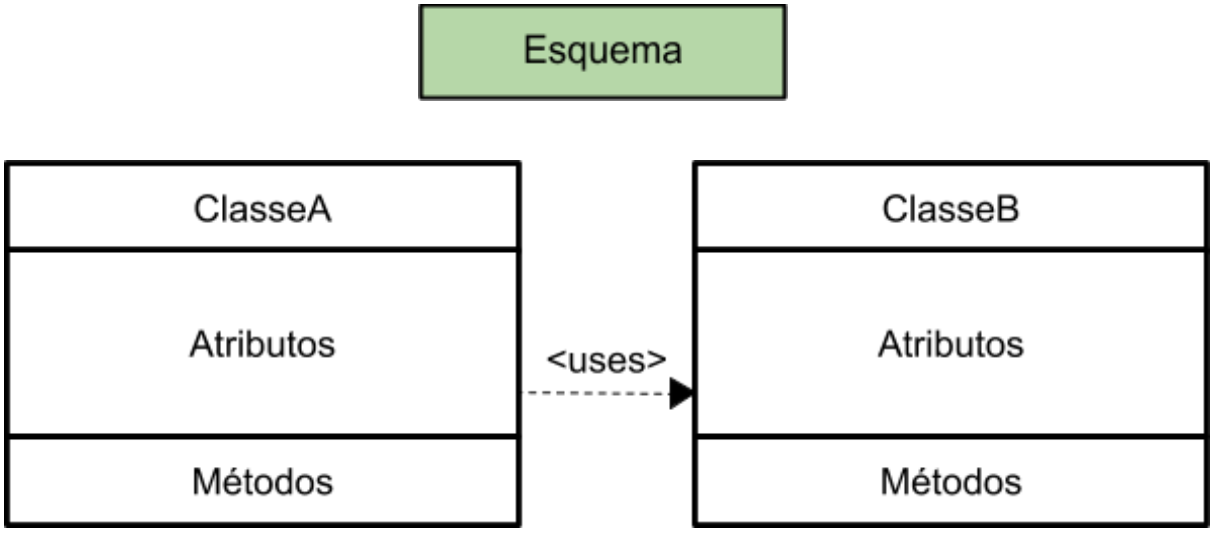
Exemplos

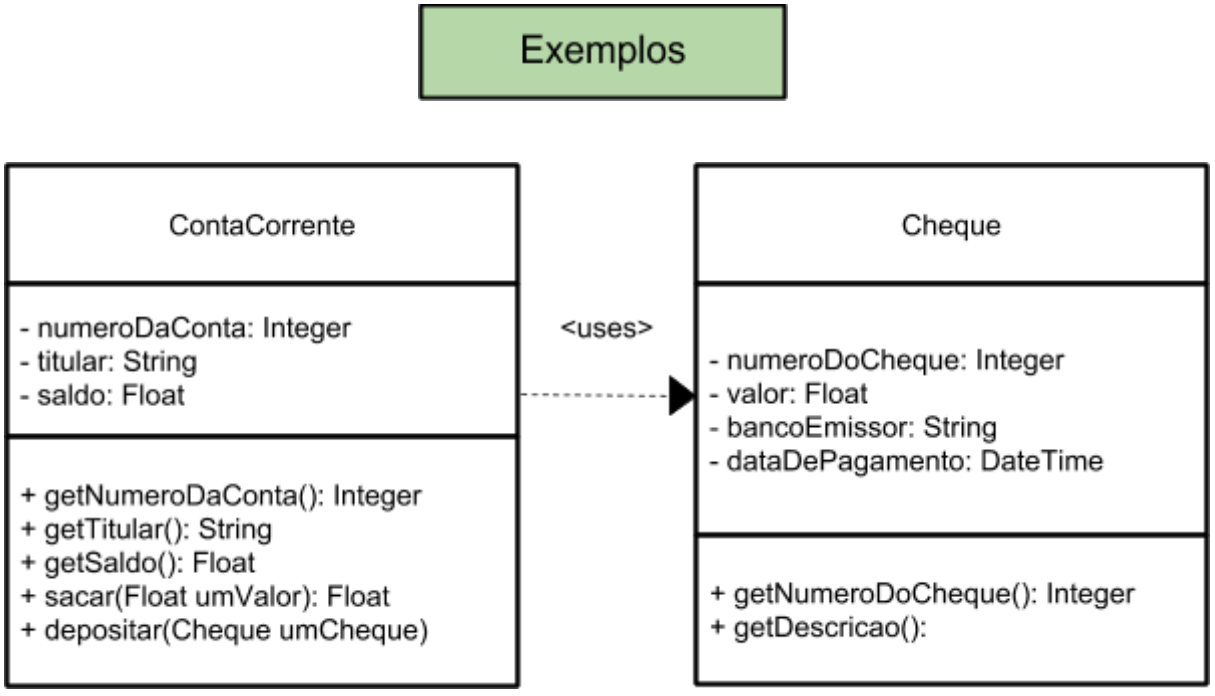



Curso
- nome: String
- codigoDoCurso: Integer
- matriculados: List<Aluno>
+ getNome(): String
+ getCodigoDoCurso(): Integer
+ getAlunosMatriculados(): List<Aluno>
+ matricularAluno(Aluno aluno)
*
Aluno
- nome: String
- sobrenome: String
- codigoAluno: Integer
- getNome(): String
- getSobrenome(): String
- getCodigoDoAluno(): Integer


Prateleira
- codigoDaPrateleira: Integer
- produtos: List<Produto>
+ getCodigoDaPrateleira(): Integer
+ getProdutos(): List<Produto>
+ adicionarProduto(Produto umProduto)
*
Produto
- codigoDoProduto: Integer
- descricao: String
- getCodigoDoProduto(): Integer
- getDescricao(): String

- **<u>Especificação do relacionamento de uso:</u>** O relacionamento de uso é estabelecida quando um objeto de uma classe usa algum objeto de outra classe do nosso modelo. Em geral, os relacionamentos de uso são estabelecidas quando, em algum método, um objeto de outra classe é usado.





- **<u>Especificação do relacionamento de implementação:</u>** O relacionamento de implementação é estabelecida quando uma classe implementa uma interface.