

Este padrão está pensado para trabalhar com os diagramas de classes dentro dos cursos da Digital

# A. Especificação de uma classe:

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

- ✓ Nome da classe
- ✓ Atributos
- √ Métodos

NomeDaClasse
Atributos
Métodos

Especificação do nome de uma classe: 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



Nome do tipo:

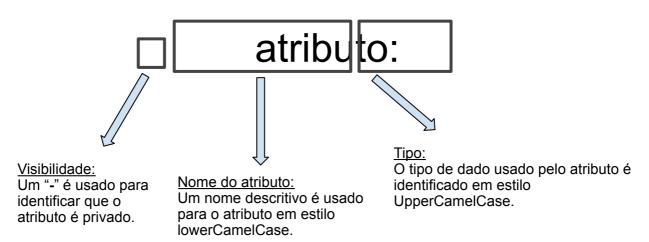
identificado.

O tipo de dado que o

método retorna é

UpperCamelCase<sup>1</sup>.

## Especificação dos atributos:





## Visibilidade:

Um "+" é usado para identificar que o método é público e um "-" para identificar que é privado.

### Nome do método:

Um nome descritivo é usado em estilo lowerCamelCase.

### Parâmetros:

Os parâmetros são especificados com

tipo e nome sendo separados por 1 CamelCase é uma forma de escrever que se aplica a frases ou palavras compostas.

UpperCamelCase คุลหลายใกละprisopire letra separen lese da a palavras é maiúscula.

Exemplo: Exemplo@elthpperCamelCase.

lowerCamelCase é igual à forma anterior, mas com a primeira letra em minúsculo.

Exemplo: exemploDeLowerCamelCase

## Esquema

## NomeDaClasse

- atributo1:Tipo - atributo2: Tipo
- +método1(parâmetros): TipoDeRetorno

# Exemplos

## Pessoa

- nome: String
- sobrenome: StringdataDeNascimento: DateTime

+getNome(): String +getSobrenome(): String +idade(): Integer

+éIrmaoDe(Pessoa uma Pessoa): Boolean



### Auto

- marca: String - modelo: String - cor: String

- quilometros: Long

+getMarca(): String +getModelo{): String

+setMarca(String umaMarca)

+ setModelo(String umModelo)

+getQuilometro(): Long

+ precisaDeService(): Boolean

# B. <u>Especificação de uma interface</u>:

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

- ✓ Nome da interface
- ✓ Métodos
- Especificação do nome de uma interface: 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 "+"

	Esquema
	< <imprimivel>&gt;</imprimivel>
+ in	nprimir()
	< <interfacea>&gt;</interfacea>
	Métodos

Exemplo

# C. <u>Especificação dos relacionamentos entre classes:</u>

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.

Esquema ClassePai **Atributos** Métodos

SubClasseB Atributos Métodos SubClasseA **Atributos** Métodos

Exemplos

Animal

# Digital HOUS⊖ > Padrão de diagrama de classes

- energia:	Integer

+ comer() + dormir()

Gato	
- quantidadeDeVidas: Integer	
+ comer() + dormir() + getQuantidadeDeVidas(): Int	eger
Cachorro	
Cachorro - raca: String	

Especificação do relacionamento de associação: 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 um. Por outro lado, especificaremos a paridade do relacionamento, ou seja, com quantos objetos de outra classe se relaciona.

Esquema

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.



ClasseA			
umAtributo: ClasseB Outros Atributos			
Métodos			
ClasseB			
Atributos			
Métodos			

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.

ClasseA umAtributo: List<ClasseB> **Outros Atributos** Métodos

# Digital HOUSE > Padrão de diagrama de classes

ClasseB

**Atributos** 

Métodos

Exemplos

## Motorista

- nome: Stringsobrenome: StringcodigoMotorista: Integer

- getNome(): StringgetSobrenome(): String
- getCodigoMotorista(): Integer

## Ônibus

- numeroDoVeiculo: Integer
- motorista: Motorista
- + getNumeroDoVeiculo(): Integer + getMotorista(): Motorista



### Aluno

- nome: String - sobrenome: String - codigoAluno: Integer

- getNome(): String - getSobrenome(): String - getCodigoDoAluno(): Integer

### Curso

- nome: String

- codigoDoCurso: Integer - matriculados: List<Aluno>

+ getNome(): String

+ getCodigoDoCurso(): Integer + getAlunosMatriculados(): List<Aluno>

+ matricularAluno(Aluno aluno)

### Prateleira

- codigoDaPrateleira: Integer - produtos: List<Produto>
- + getCodigoDaPrateleira(): Integer
- + getProdutos(): List<Produto> + adicionarProduto(Produto umProduto)

## Produto

- codigoDoProduto: Integer
- descrição: String
- getCodigoDoProduto(): Integer
- getDescricao(): String



Especificação do relacionamento de uso: O relacionamento de uso é estabelecido quando um objeto de uma classe usa algum objeto de outra classe do nosso modelo. Em geral, os relacionamentos de uso são estabelecidos quando, em algum método, um objeto de outra classe é usado.

Esquema

ClasseA Atributos Métodos ClasseB Atributos Métodos



<uses>

## **Exemplos**

## ContaCorrente

- numeroDeConta: Integer
- titular: String - saldo: Float
- + getNumeroDaConta(): Integer
- + getTitular(): String + getSaldo(): Float
- + sacar(Float umValor): Float
- + depositar(Cheque umCheque)

## Cheque

- numeroDeCheque: Integer
- valor: Float
- bancoEmissor: String
- dataDePagamento: DateTime
- + getNumeroDoCheque(): Integer
- + getDescricao(): String

<uses>

Especificação do relacionamento de implementação: O relacionamento de implementação é estabelecido quando uma classe implementa uma interface.

Esquema

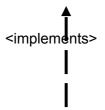
Exemplo

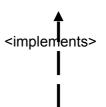
<<InterfaceA>>

Métodos

<<Imprimivel>>

+ imprimir()







## Contrato

- numeroDoContrato: Integerdescrição: String
- + getNumeroDoContrato(): Integer getDescricao(): String + imprimir():

ClasseB

Atributos

Métodos