Gente,
Saúde e
Bem-estar.

RaiaDrogasil S.A.

ORIENTAÇÃO A OBJETOS

Conceitos Básicos

ABORDAGEM DAS AULAS

- Apresentação de conceitos (com slides, exemplos UML)
- Exercícios para verificar absorção dos conceitos
- Avalição final (modelagem de classes)

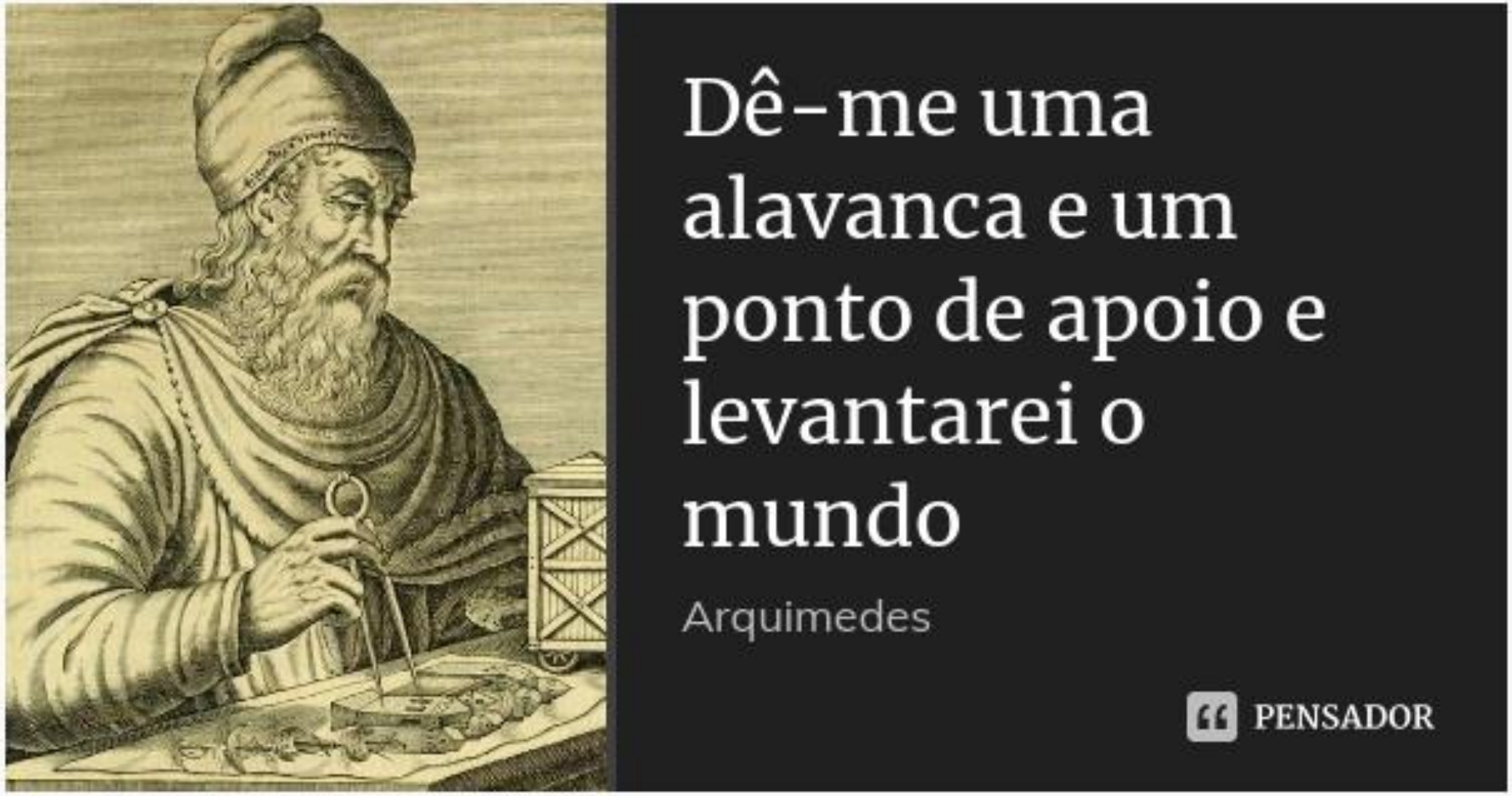
AGENDA

1. Introdução
2. Breve Histórico da Orientação a Objetos
3. Por que usar Orientação a Objetos?
4. Orientação a Objetos
5. Os Conceitos Estruturais
6. Os Conceitos Relacionais
7. Os Conceitos Organizacionais
8. Avaliação final

01

Introdução

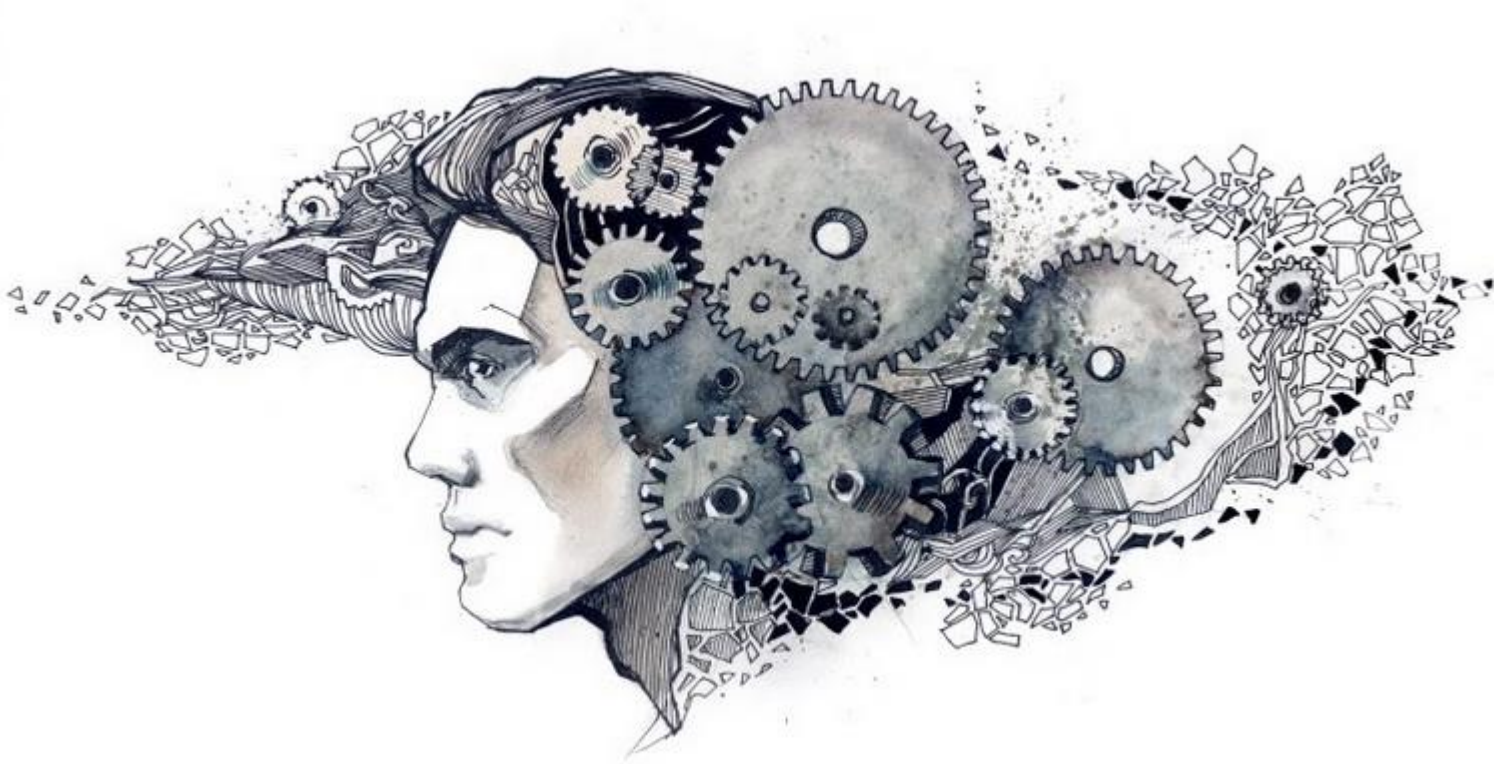
INTRODUÇÃO


Dê-me uma alavanca e um ponto de apoio e levantarei o mundo
Arquimedes
PENSADOR

INTRODUÇÃO

Paradigmas de Programação

"Paradigma de programação é a maneira de classificar uma linguagem de programação baseada em suas recursos. As linguagens de programação podem ser classificadas em múltiplos paradigmas." [1, Wikipedia]

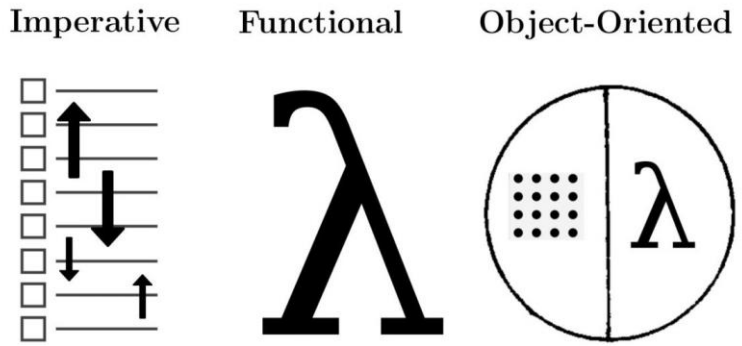


INTRODUÇÃO

Paradigmas de programação mais comuns:

- Imperativo: programador instrui a máquina como mudar seu estado
  - Estruturado, Procedural: as instruções (controles de fluxo de seleção e repetição) são agrupadas em procedimentos
  - Orientado a Objetos: as instruções são agrupadas junto com a parte do estado que elas operam
- Declarativo: programador declara as propriedades do resultado desejado, mas não como obtê-lo
  - Funcional: o resultado é o valor obtido pela aplicação e composição de uma séria de funções
  - Lógico: o resultado é a resposta a uma questão sobre um sistema de fatos e regras
  - Matemático: o resultado é a solução de um problema de otimização

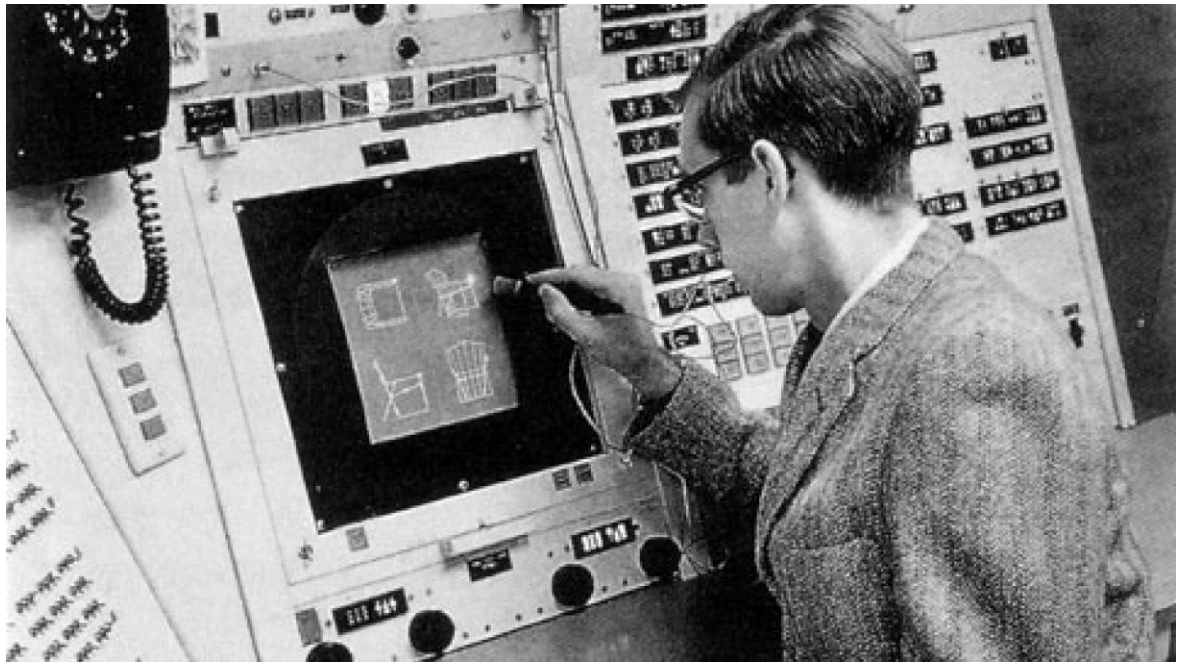
[1, Wikipedia]



02

Breve Histórico da Orientação a Objetos

BREVE HISTÓRIA DA ORIENTAÇÃO A OBJETOS



Ivan Sutherland e o Sketchpad

- Surgimento da Orientação a Objeto – final da década de 1950 [2, 3, 4, 5, Wikipedia]
  - 1960 – LISP: “objetos” seriam como os itens identificados (átomos de LISP) com atributos
  - 1960-61 – Sketchpad: noções de “objeto” e “instância”
  - 1960’s – AED-0: relação direta entre estruturas de dados (plexes) e procedures

- Principal referência – Conceito de Simulação
  - “Simular os eventos do dia a dia em sistemas digitais”
  - Formalização da teoria - Keith Tocher, em The Art of Simulation (1967)

BREVE HISTÓRIA DA ORIENTAÇÃO A OBJETOS

Dahl e Nygaard

- Simula I e Simula 67
  - Linguagens de simulação de eventos discretos criadas nos 1960's por Kristen Nygaard e Ole-Johan Dahl (Noruega)
  - Publicação do paper "Class and Subclass Declarations" (1967)
  - 1ª linguagem OO de renome

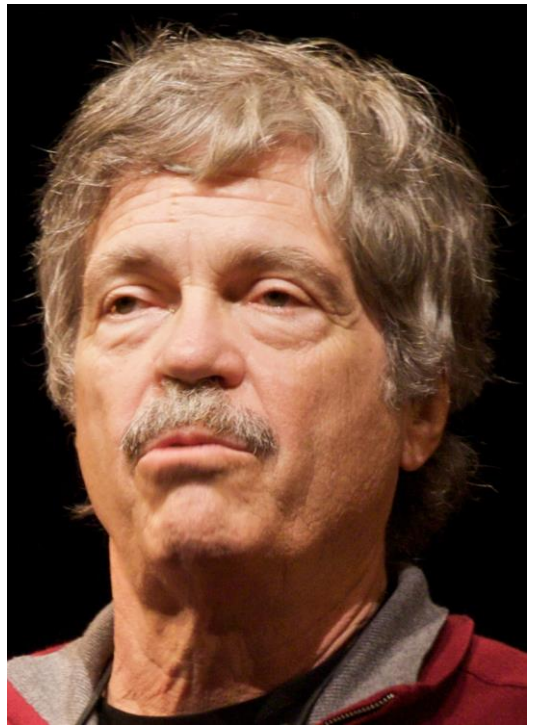
- Smalltalk
  - Smalltalk-71, criada por Alan Kay nos 1970's (Califórnia, EUA)
  - Smalltalk-80, versão estável
  - Para ser usada em PC's
  - Interface amigável e ambiente de desenvolvimento integrado (IDE)

[2, Wikipedia]

BREVE HISTÓRIA DA ORIENTAÇÃO A OBJETOS

"Eu imaginei os objetos como sendo células biológicas ou computadores individuais em uma rede, somente capazes de se comunicar através de mensagens (assim, as mensagens vieram logo no início – levou-se um tempo até perceber como mandar mensagens em um linguagem de programação seria eficientemente útil."

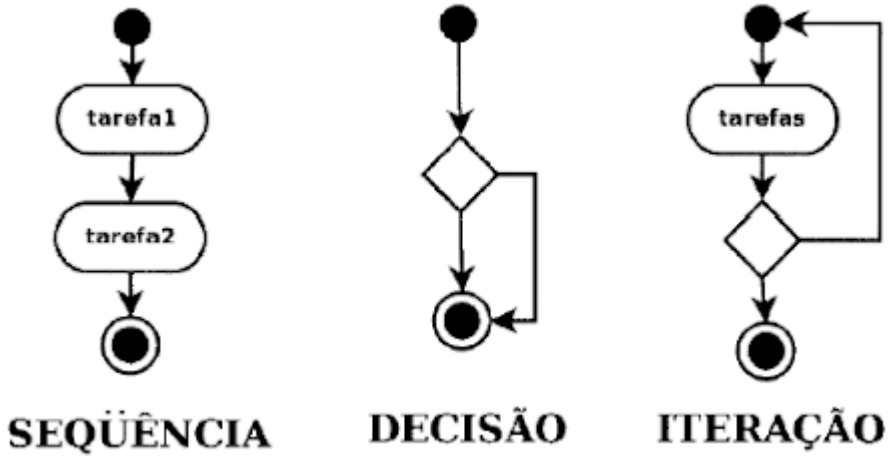
- Alan Kay, em *Dr. Alan Kay on the Meaning of "Object-Oriented Programming" (2003)*



03

Por que usar Orientação a Objetos?

POR QUE USAR ORIENTAÇÃO A OBJETOS?



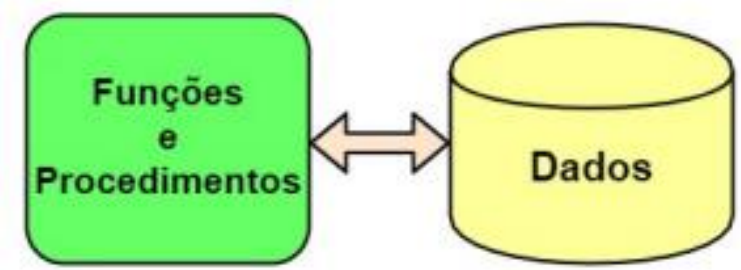
Paradigma Estruturado

- É possível representar qualquer processo do mundo real com:
  - Sequência
  - Decisão
  - Iteração
- Prós
  - Simplicidade
  - Fácil de aprender
- Contras
  - Fraca representatividade (sistemas complexos)
  - Pouca manutenibilidade (difícil reuso)

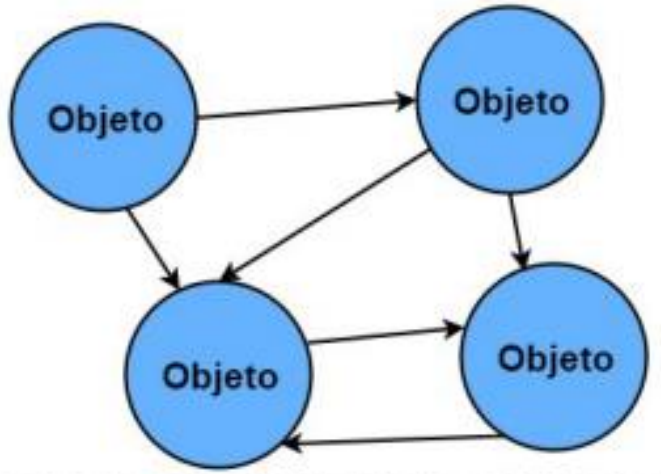
POR QUE USAR ORIENTAÇÃO A OBJETOS?

Paradigma Orientado a Objetos

- Reuso
  - Mecanismos para reuso de código: herança e associação
- Coesão
  - Cada unidade de código, única responsabilidade: classes e associação
- Acoplamento
  - “Nível de interdependência entre os códigos de um programa de computador”: classes e associação



Programação Estruturada



Programação Orientada a Objetos

POR QUE USAR ORIENTAÇÃO A OBJETOS?

Para refletir...

01-para-refletir-pq-usar-oo.docx

04

Orientação a Objetos

ORIENTAÇÃO A OBJETOS

"A Orientação a Objetos é um paradigma de análise, projeto e programação de sistemas de software baseado na composição e interação entre diversas unidades de software chamadas de objetos“ [6, Wikipedia]

Abstração

"Processo pelo qual se isolam características de um objeto, considerando os que tenham em comum certos grupos de objetos"

ORIENTAÇÃO A OBJETOS

Reuso

Quanto mais códigos são repetidos pela aplicação, mais difícil vai se tornando sua manutenção.

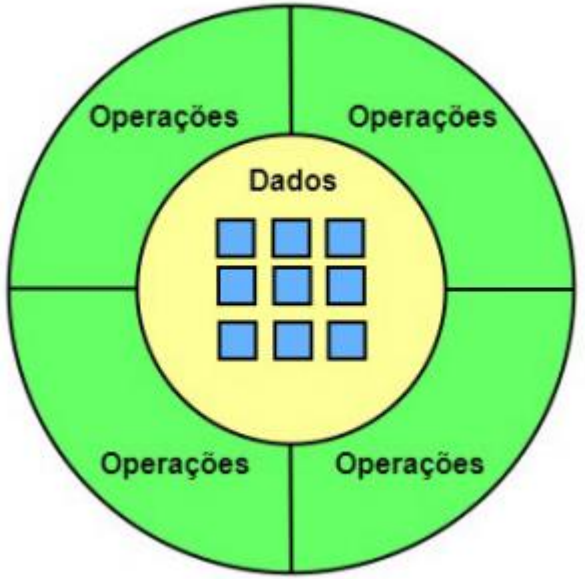
ORIENTAÇÃO A OBJETOS

Encapsulamento

RECEITUÁRIO (COMPLEXO)

- 400mg de ácido acetilsalicílico
- 1mg de maleato de dexclorfeniramina
- 10mg de cloridrato de fenilefrina
- 30mg de cafeína

Misturar bem e ingerir com água. Repetir em momentos de crise.



RECEITUÁRIO (ENCAPSULADO)

1 comprimido de Resfriol. Ingerir com água. Repetir em momentos de crise.

ORIENTAÇÃO A OBJETOS

Para refletir...

02-para-refletir-oo.docx

05

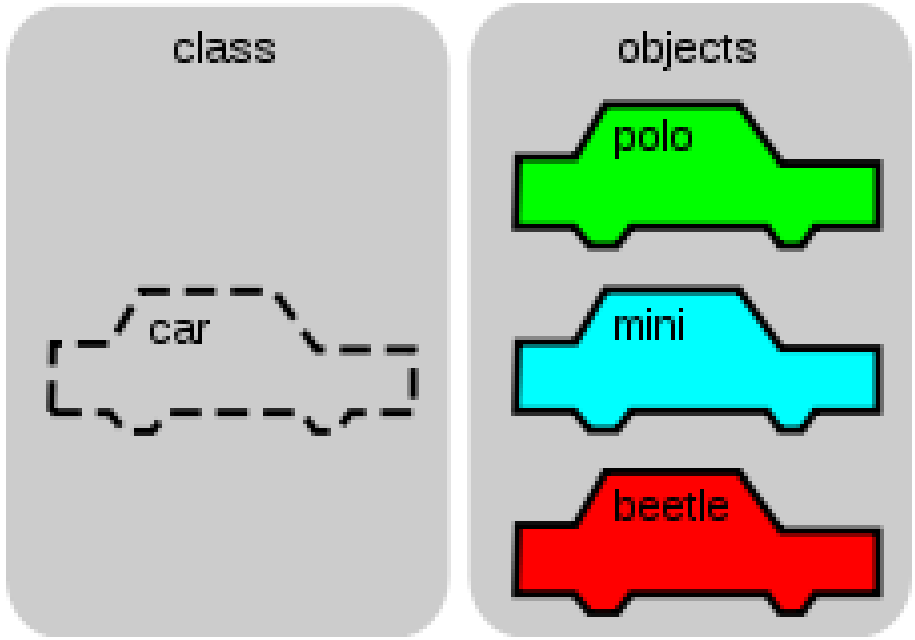
Os Conceitos Estruturais

OS CONCEITOS ESTRUTURAIS

Classe

"Classe é uma estrutura que abstrai um conjunto de objetos com características similares. Uma classe define o comportamento de seus objetos através de métodos e os estados possíveis destes objetos através de atributos. Em outros termos, uma classe descreve os serviços providos por seus objetos e quais informações eles podem armazenar"

[7, Wikipedia]



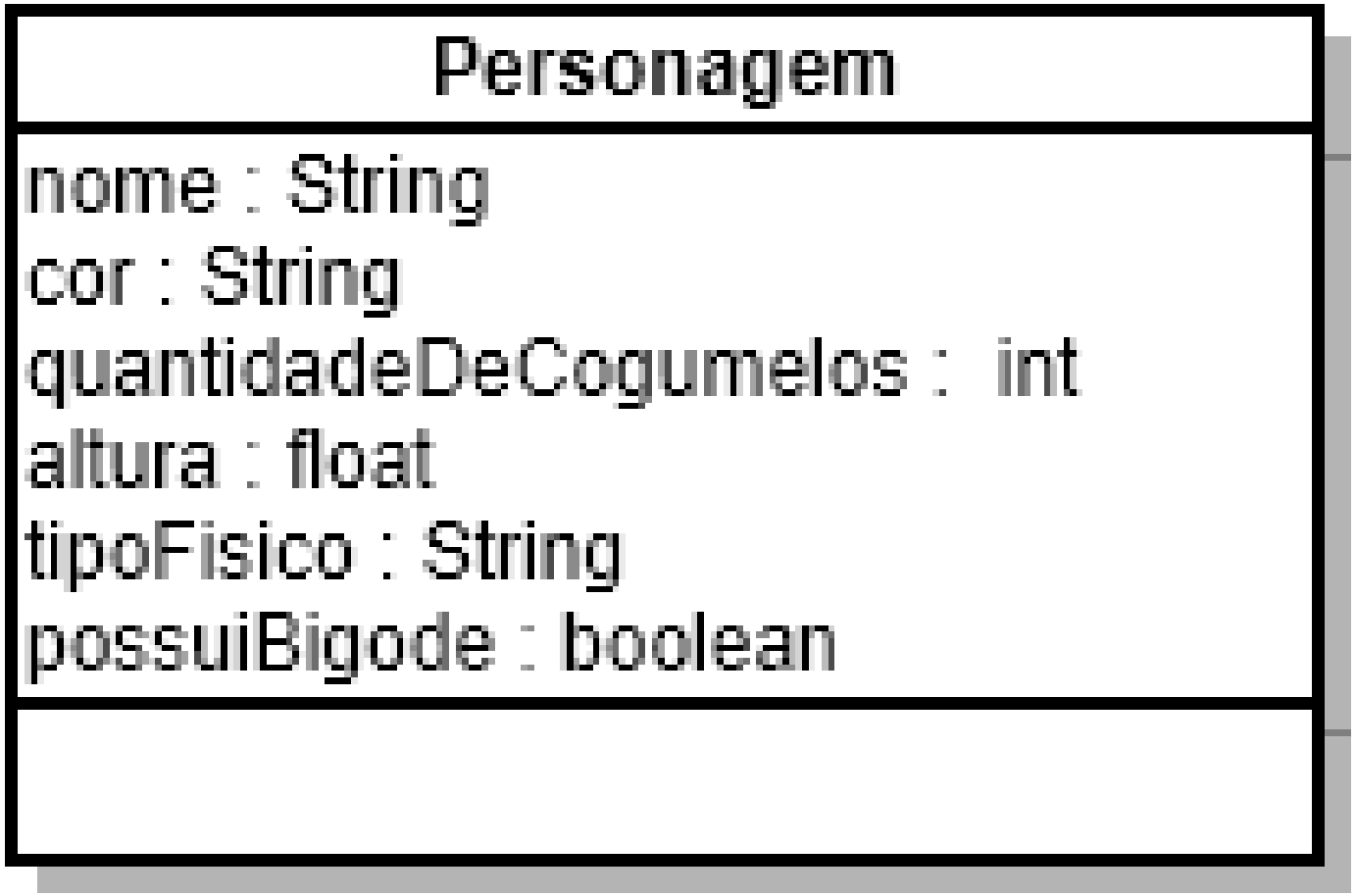
OS CONCEITOS ESTRUTURAIS

Atributo

"Atributo é o elemento de uma classe responsável por definir sua estrutura de dados. O conjunto destes será responsável por representar suas características e fará parte dos objetos criados a partir da classe."

CLASSE CARRO		OBJETO CARRO A	OBJETO CARRO B
Atributos de objeto	Marca	Ford	Mitsubishi
	Modelo	Fiesta	L-200
	Cor	branco	azul royal
	Combustível	gasolina	diesel

Exemplo de Atributos

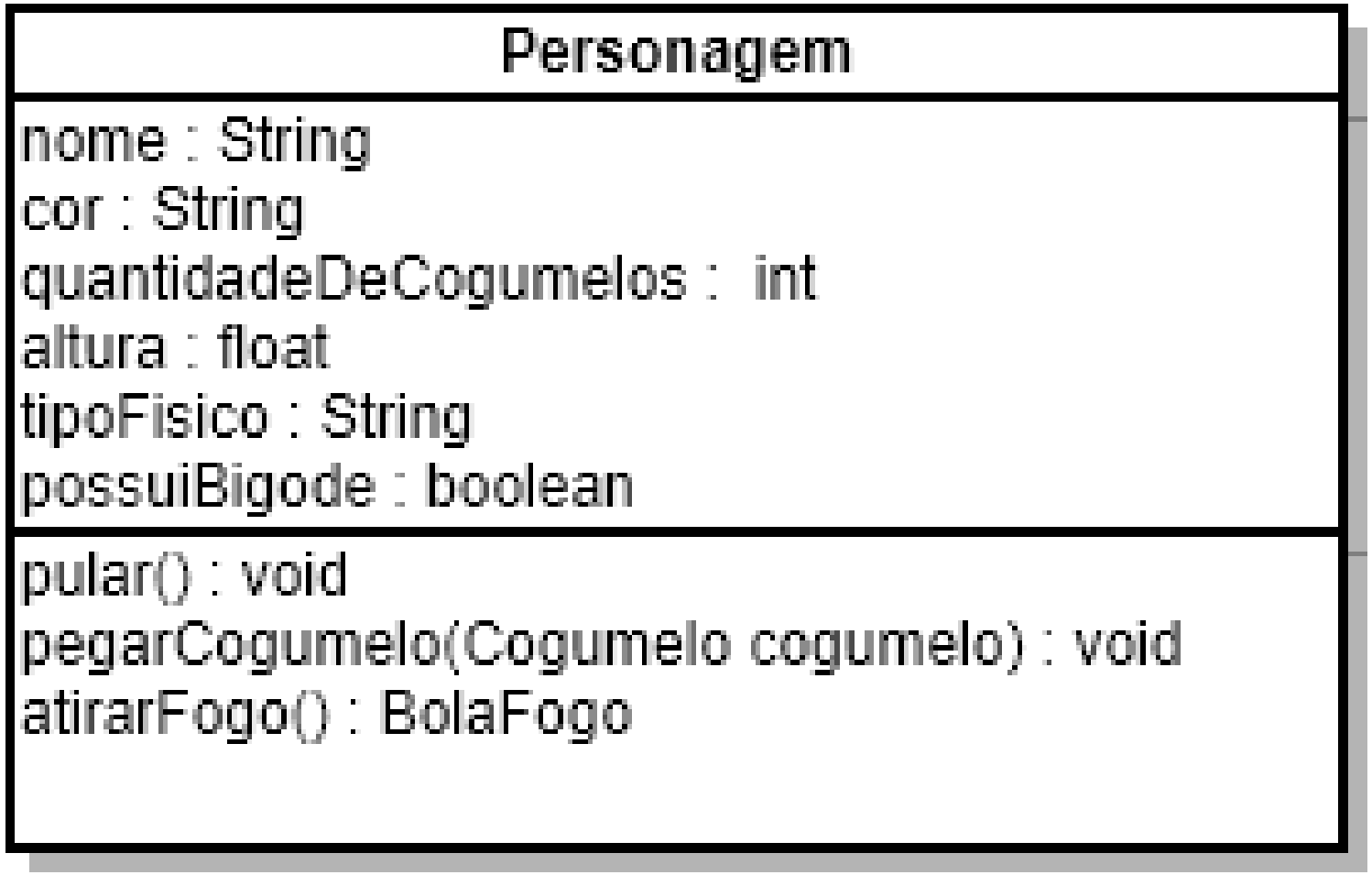


Método

"Método é uma porção de código (sub-rotina) que é disponibilizada pela classe. Esta é executado quando é feita uma requisição a ela. Um método serve para identificar quais serviços, ações, que a classe oferece. Eles são responsáveis por definir e realizar um determinado comportamento."

CLASSE CARRO		OBJETO CARRO A	OBJETO CARRO B
Atributos de objeto	Marca	Ford	Mitsubishi
	Modelo	Fiesta	L-200
	Cor	branco	azul royal
	Combustível	gasolina	diesel
Métodos	ligar		
	acelerar		
	frear		

Exemplo de Métodos



OS CONCEITOS ESTRUTURAIS

Método Construtor

É responsável por criar objetos a partir da classe em questão.

Método Destrutor

É responsável por destruir o objeto criado a partir da classe.

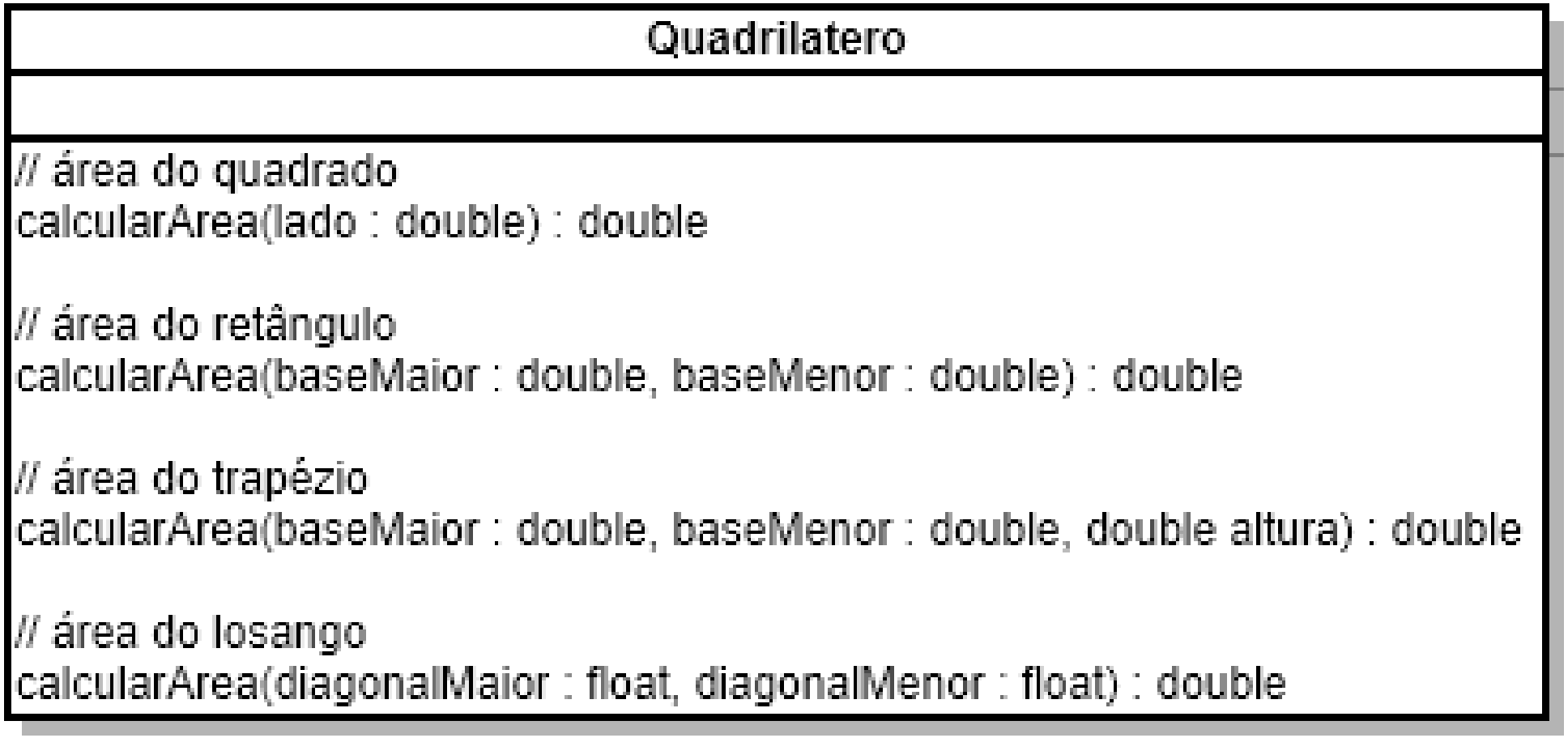
OS CONCEITOS ESTRUTURAIS

Sobrecarga de Método

Quando um mesmo método possui entradas (parâmetros) diferentes para realizar operações diferentes em determinado contexto.

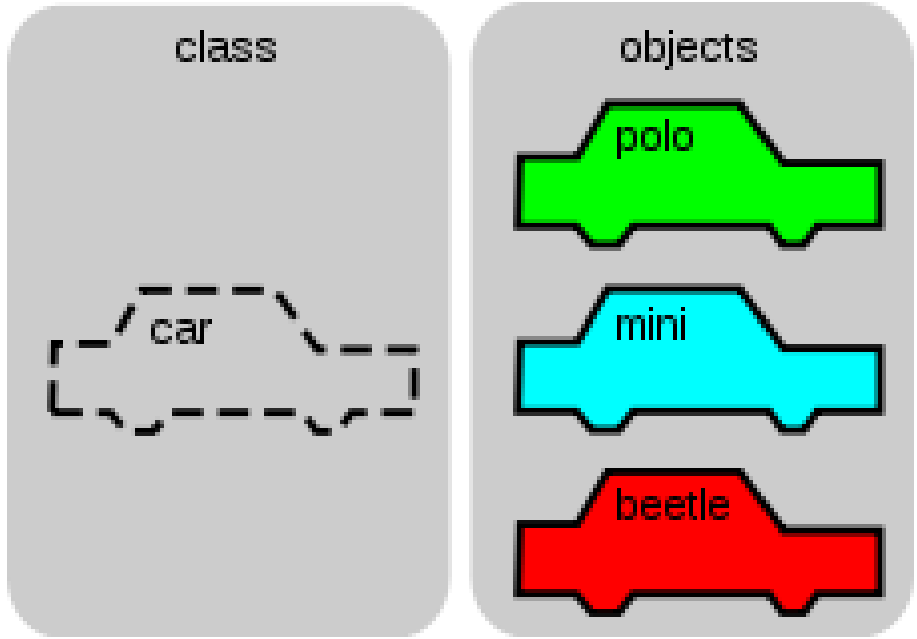
OS CONCEITOS ESTRUTURAIS

Sobrecarga de Método



Objeto

Um objeto é a representação de um conceito/entidade do mundo real, que pode ser física ou conceitual e possui um significado bem definido para um determinado software.



OS CONCEITOS ESTRUTURAIS

O estado de um objeto

O estado de um objeto é o conjunto dos valores dos seus atributos em um determinado momento.

A identidade (igualdade) de um objeto

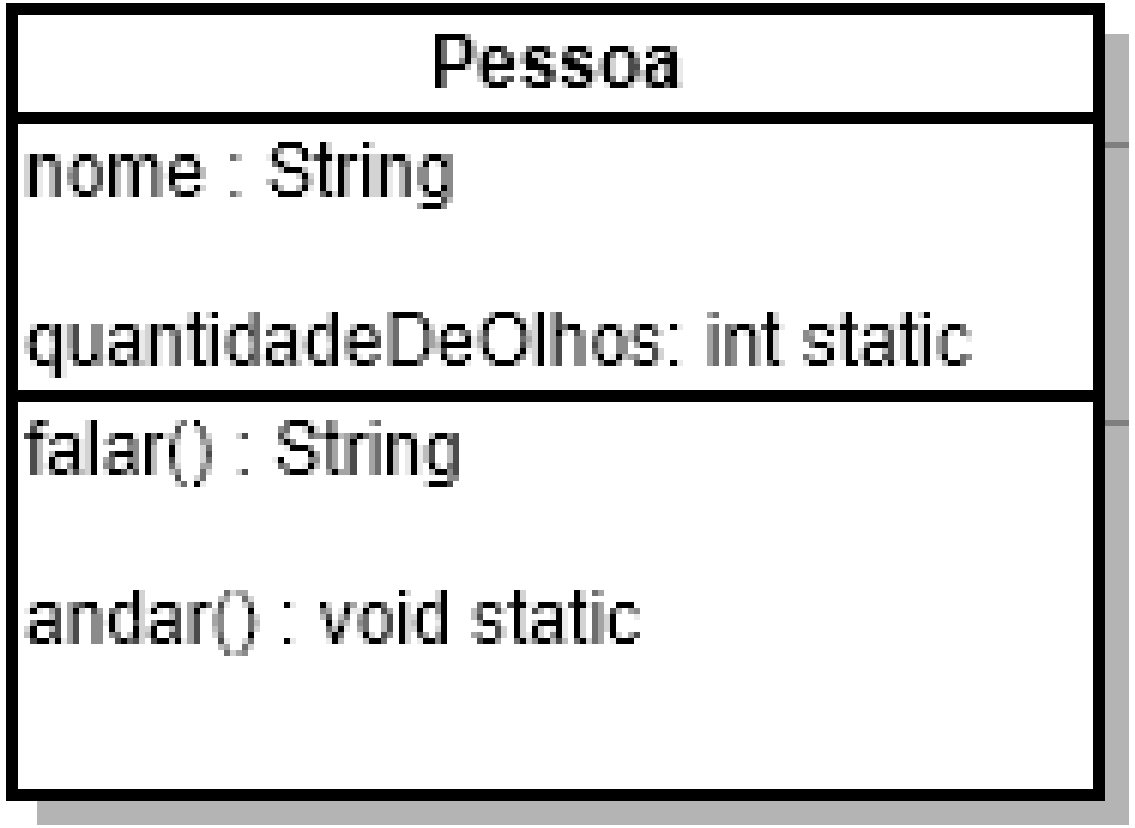
Todo objeto é único, mesmo que sejam criados da mesma classe e que seus estados sejam iguais por coincidência.

Tipos de Atributos e Métodos

Atributos e métodos podem ser de instância ou estáticos.

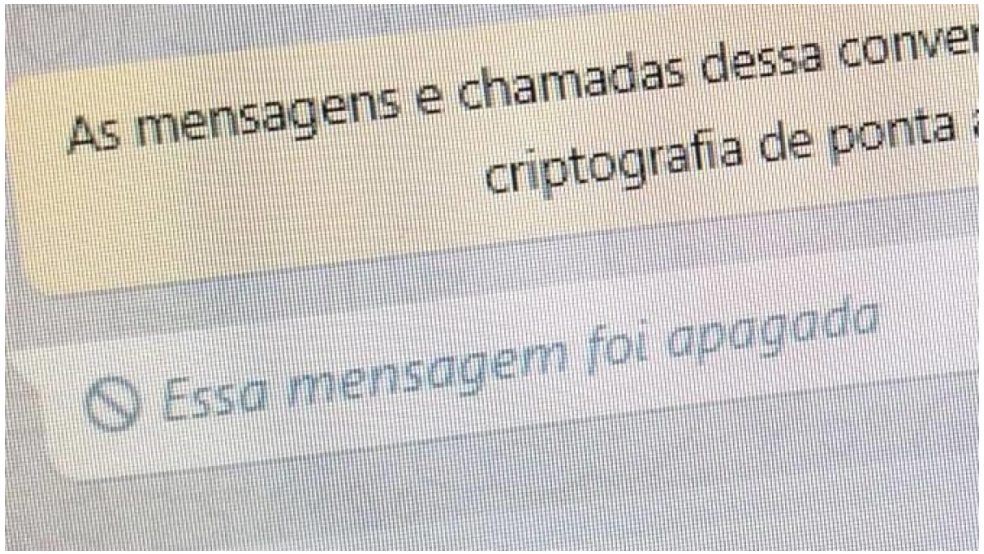
Atributos/métodos de instância pertencem ao objeto.

Atributos/métodos estáticos pertencem à classe.



Mensagem

Mensagem é o processo de ativação de um método de um objeto.



OS CONCEITOS ESTRUTURAIS

O que é UML?

Unified Modeling Language (UML), linguagem gráfica de modelagem de sistemas Orientados a Objetos.

http://www.uml.org/

Classe: Personagem

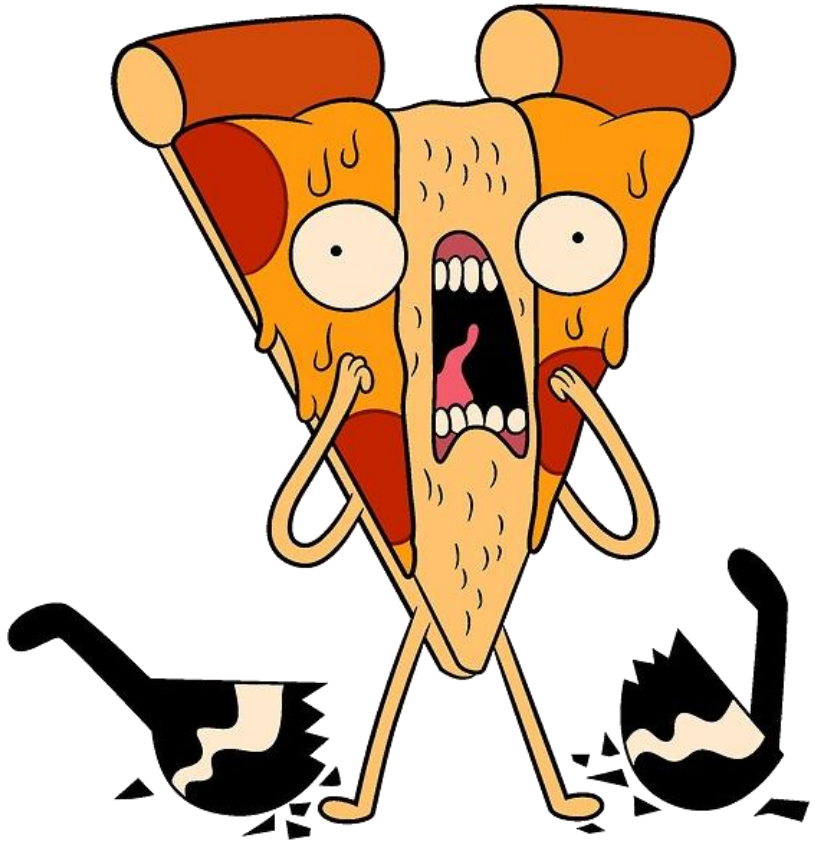
Atributos:
- -Nome
- -Cor
- -Quantidade de Cogumelos
- -Altura
- -Tipo Físico
- -Possui Bigode

Métodos:
- -Pular
- -Pegar Cogumelo
- -Atirar Fogo

Personagem
nome : String cor : String quantidadeDeCogumelos : int altura : float tipoFisico : String possuiBigode : boolean
pular() : void pegarCogumelo(cogumelo : Cogumelo) : void atirarFogo() : BolaFogo

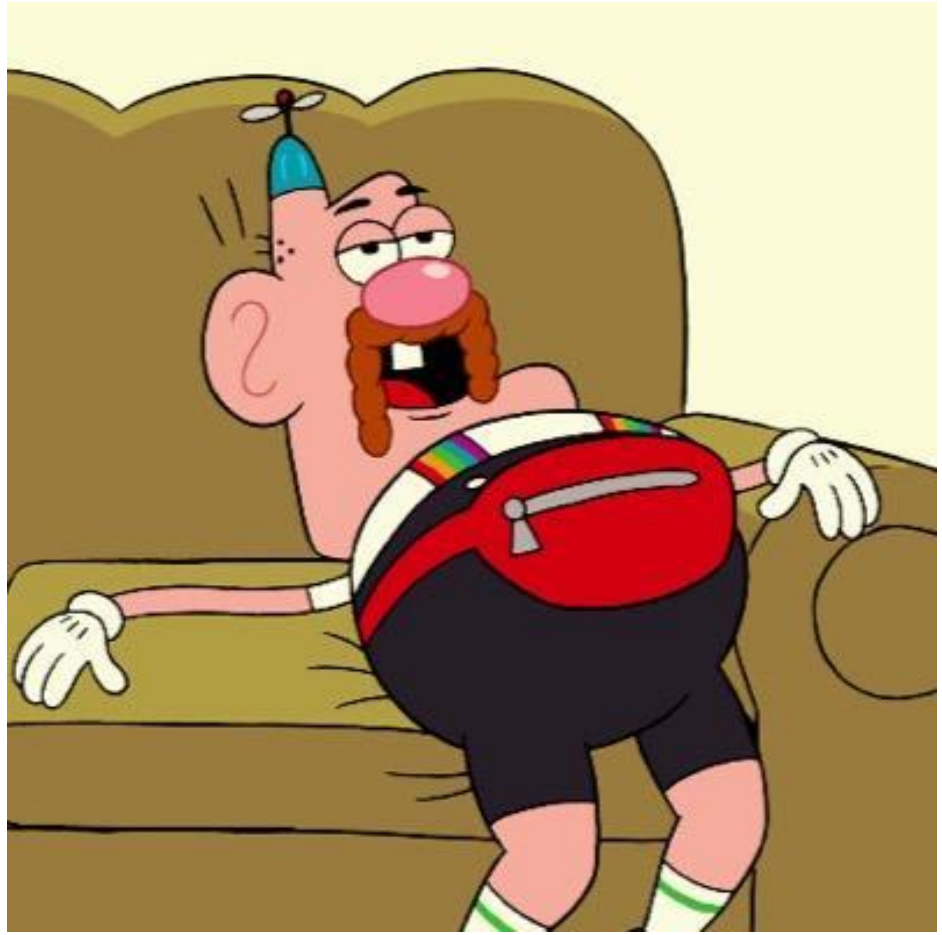
OS CONCEITOS ESTRUTURAIS

Exercícios



OS CONCEITOS ESTRUTURAIS

Para refletir...



03-para-refletir-conceitos-estruturais.docx

06

Os Conceitos Relacionais

OS CONCEITOS RELACIONAIS

Herança

Herança é o relacionamento entre classes em que uma classe chamada de subclasse (classe filha ou classe derivada) é um subtipo de outra classe chamada de superclasse (classe mãe ou classe base).

exemplo espécie humana:

Reino: Animalia

Filo: Chordata

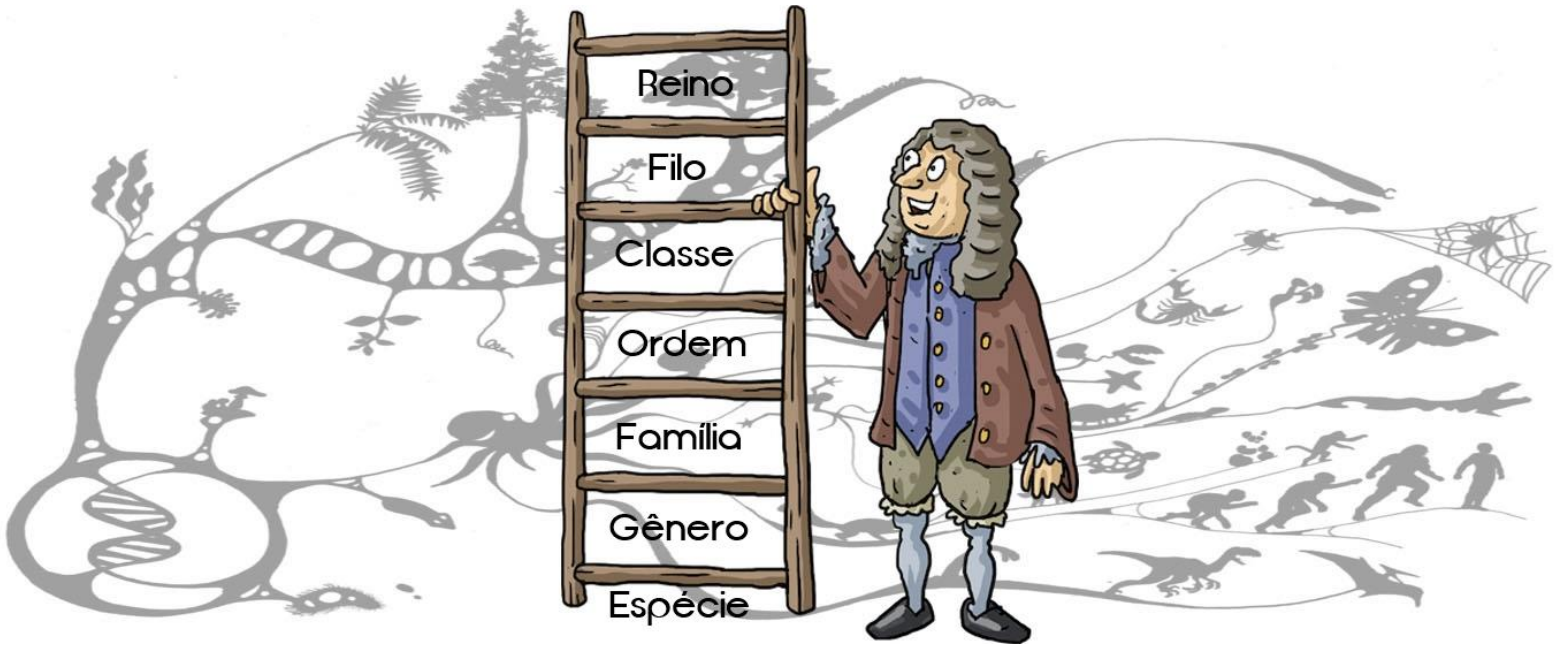
Classe: Mammalia

Ordem: Primates

Família: Hominidae

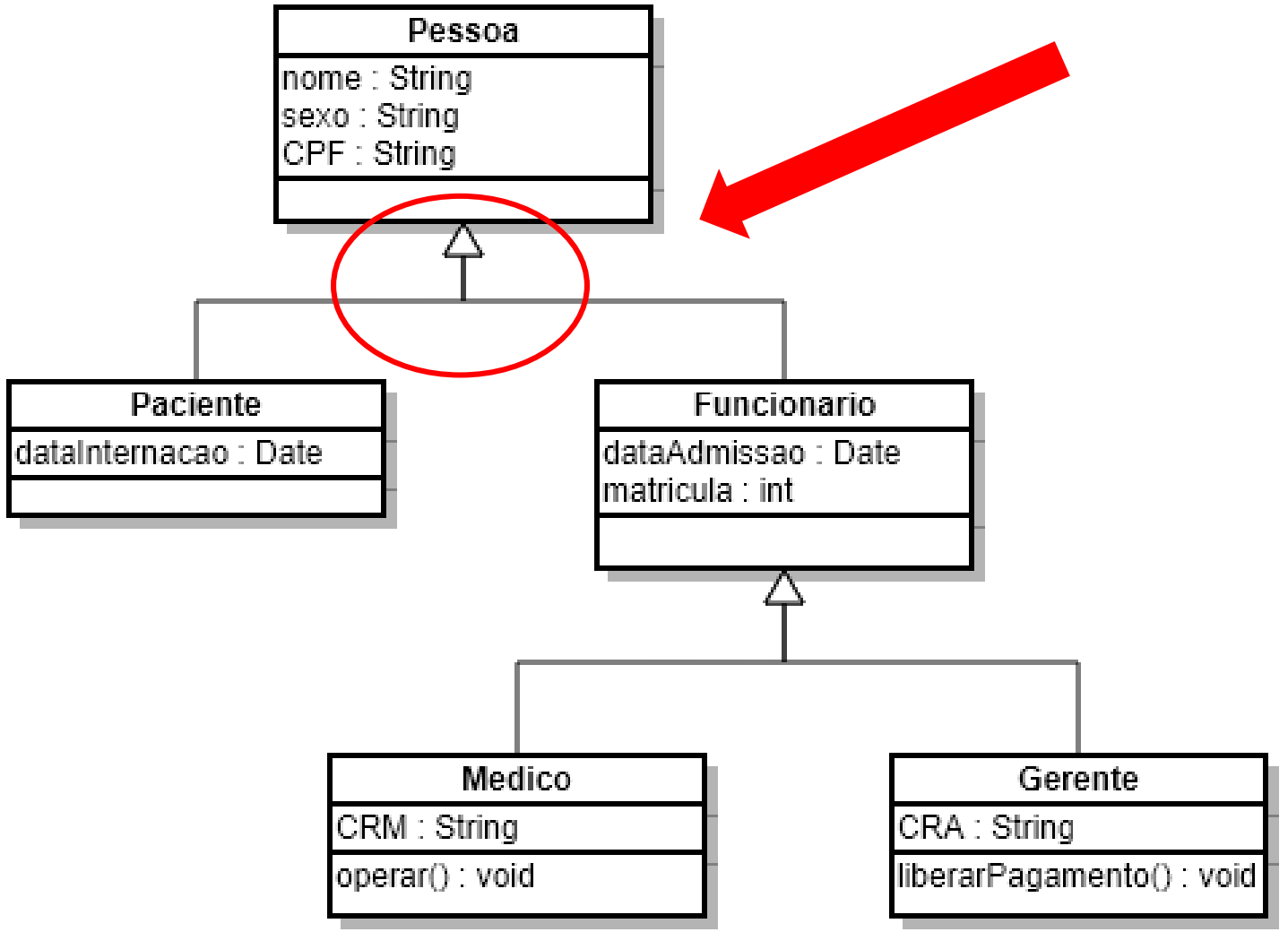
Gênero: Homo

Espécie: Homo Sapiens



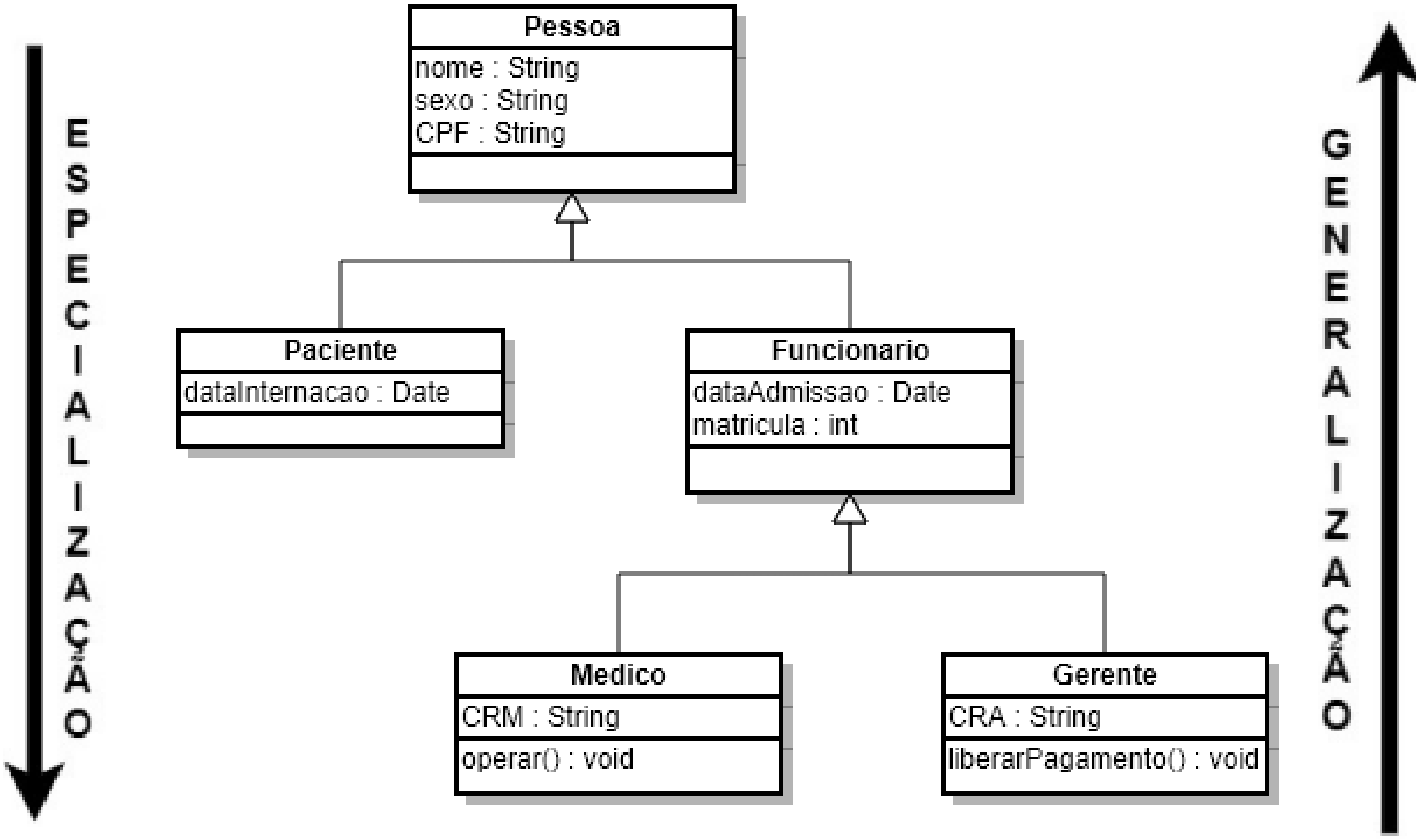
OS CONCEITOS RELACIONAIS

Herança



OS CONCEITOS RELACIONAIS

Herança



OS CONCEITOS RELACIONAIS

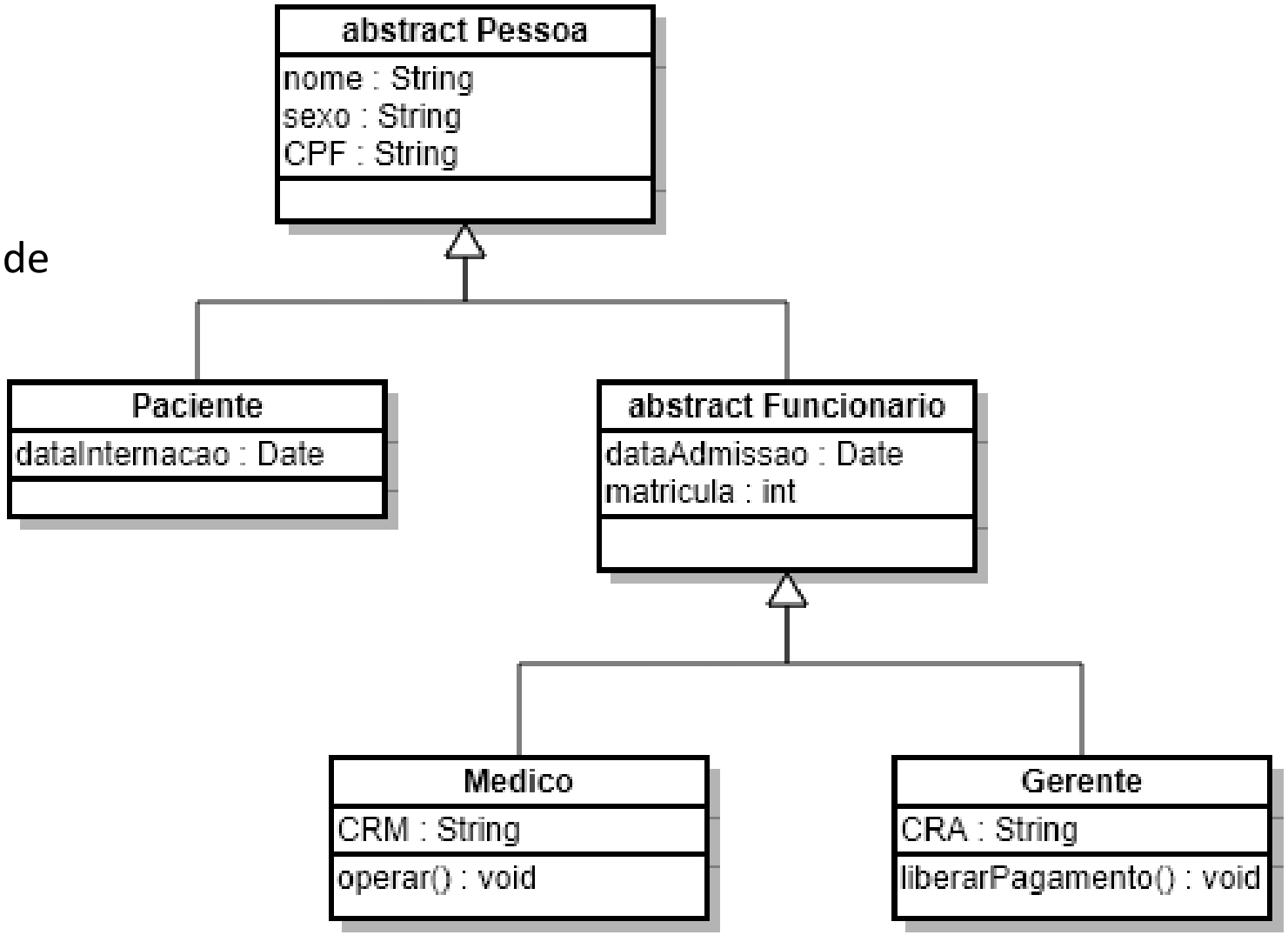
Tipos de Classes

Classes abstratas

É a implementação completa do conceito de abstração.

Representam conceitos genéricos e incompletos.

Não podem ser instanciadas



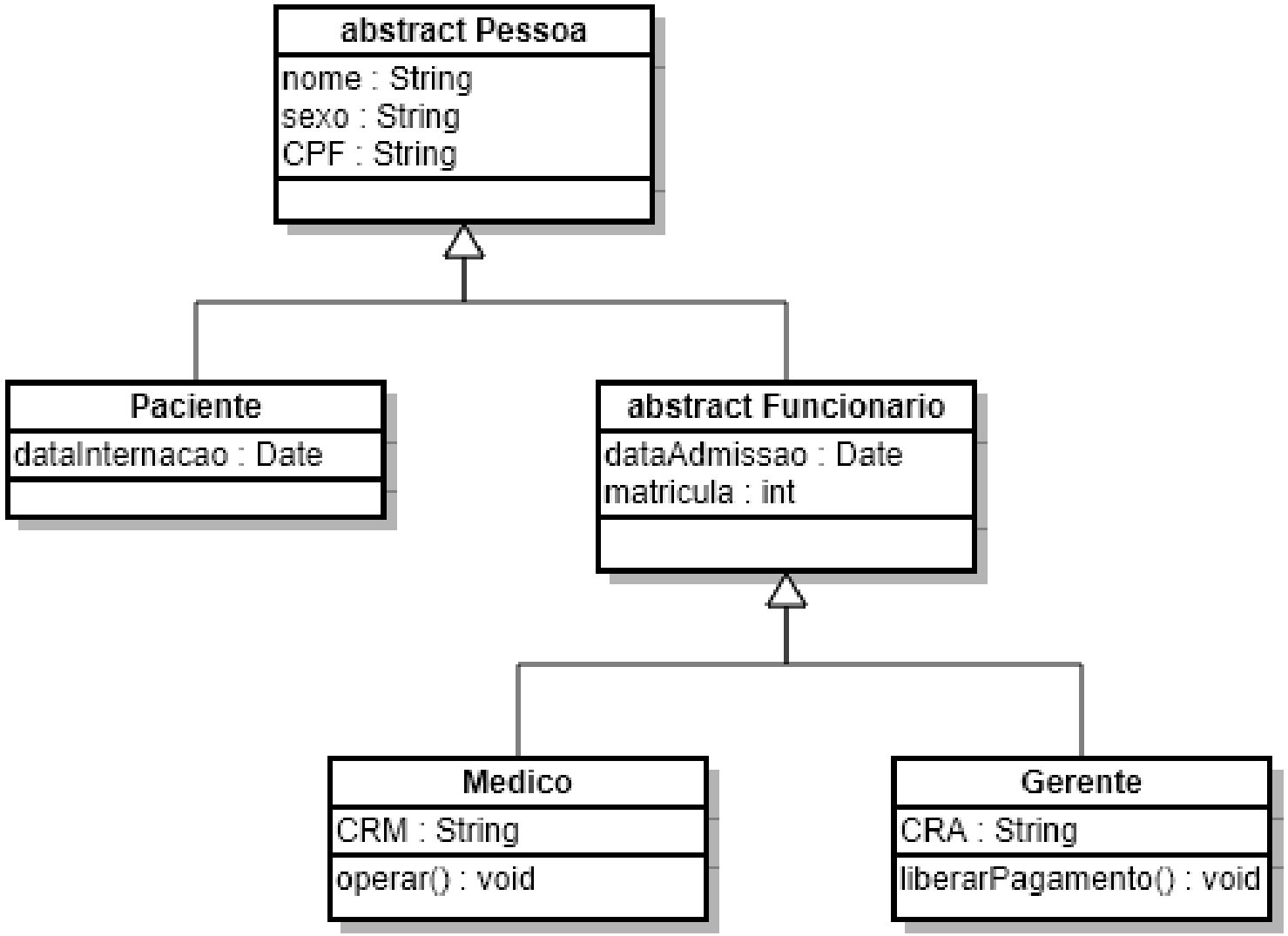
Tipos de Classes

Classes concretas

É o contrário das classes abstratas.

Representam conceitos bem específicos.

Devem ser instanciadas.



OS CONCEITOS RELACIONAIS

Tipos de Herança

Herança Simples

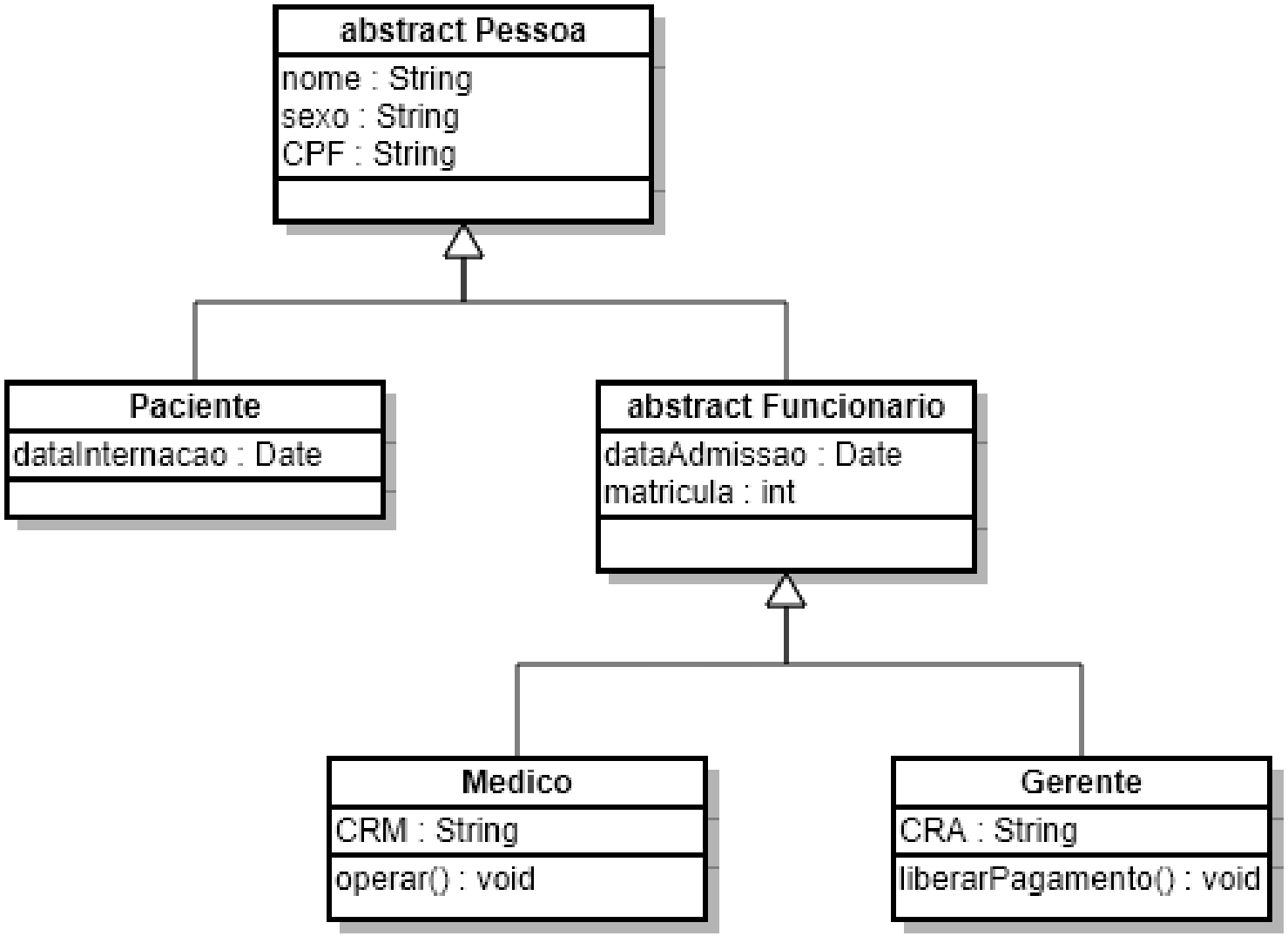
Quando uma subclasse herda de apenas uma superclasse.

Herança Múltipla

Quando uma subclasse herda de duas ou mais superclasses.

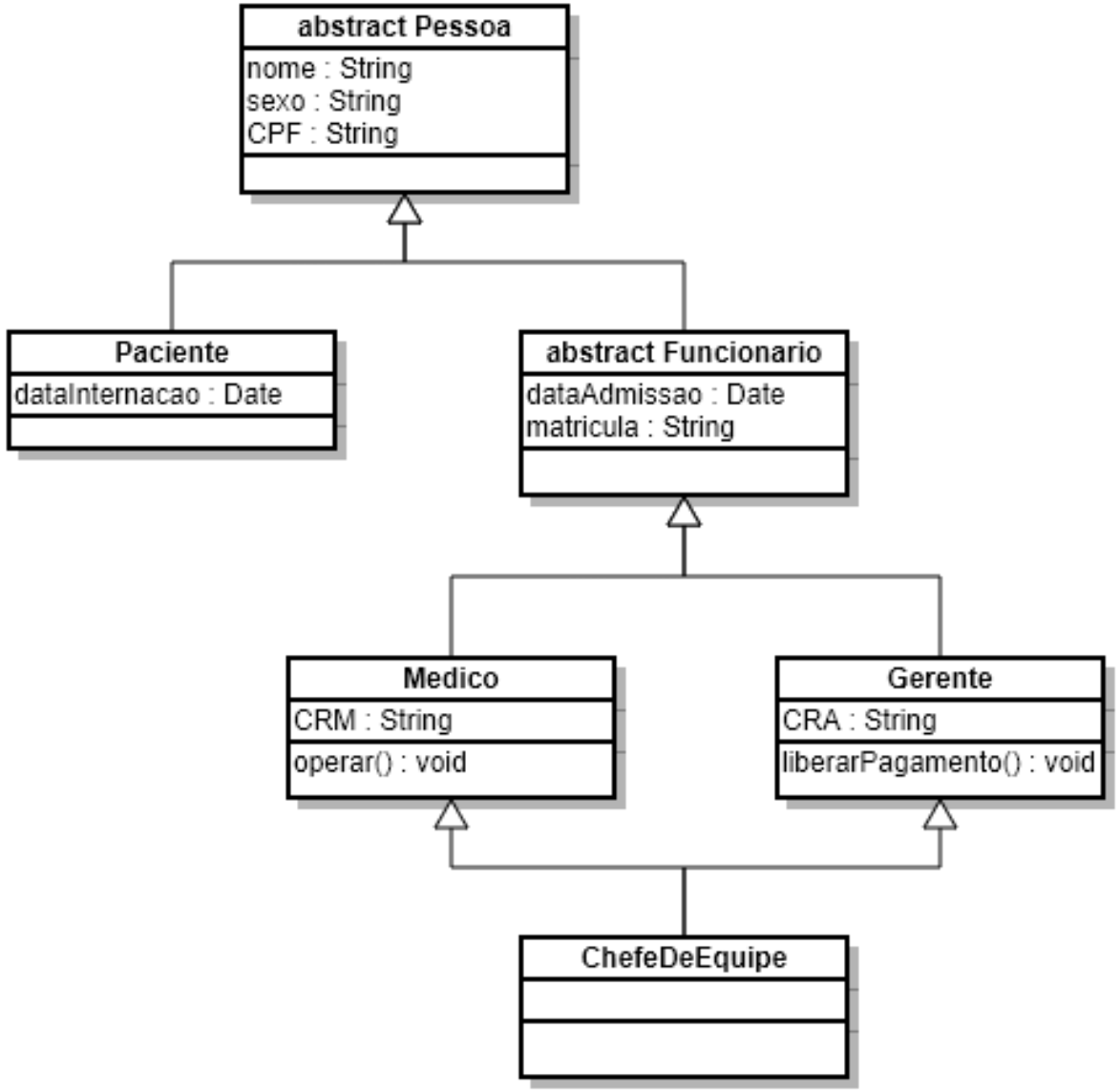
OS CONCEITOS RELACIONAIS

Herança Simples



OS CONCEITOS RELACIONAIS

Herança Múltipla



OS CONCEITOS RELACIONAIS

Upcast

É uma operação de conversão em que o objeto do tipo de uma subclasse é promovido ao tipo da superclasse.

```
//Java
Pessoa pessoa;
pessoa = new Medico();
pessoa = new Paciente();
pessoa = new Funcionario();
Funcionario funcionario = new Gerente();
Medico medico = new Anestesista();
```

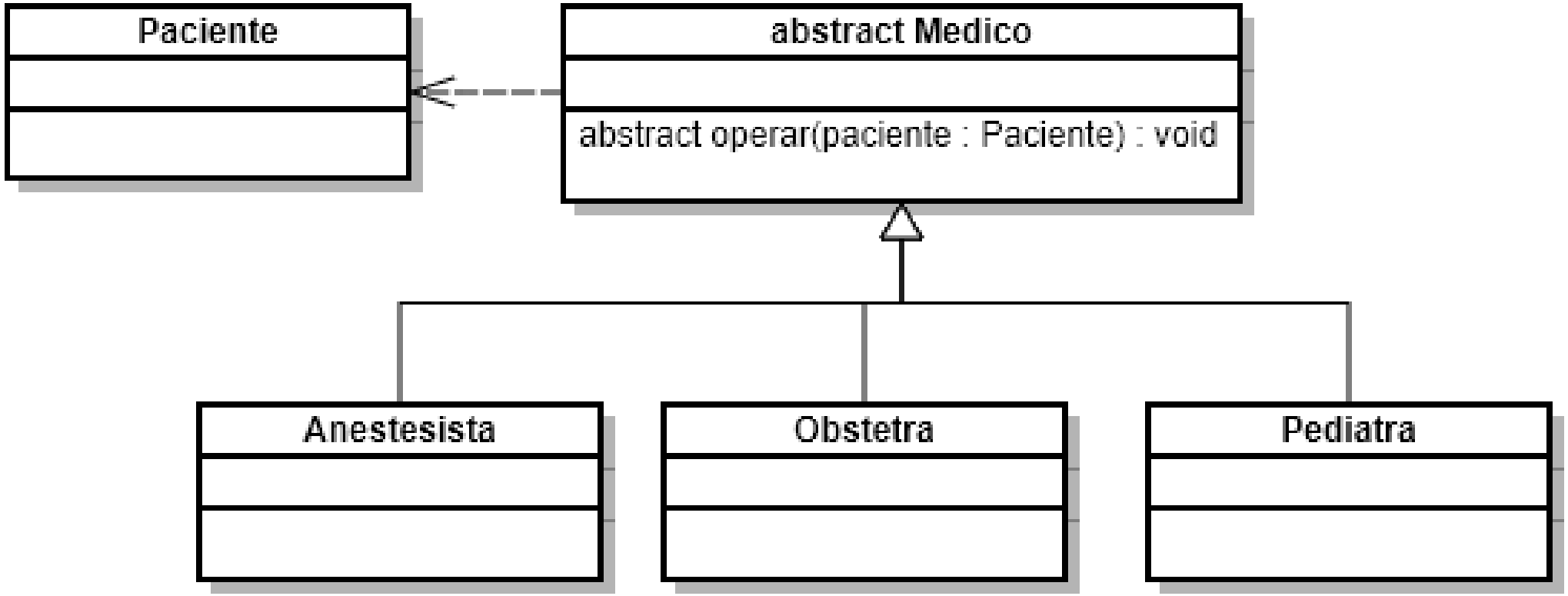
Downcast

É uma operação de conversão em que o objeto do tipo de uma superclasse é convertido ao tipo da subclasse.

```
//Java
Funcionario funcionario1 = new Gerente();
Gerente gerente1 = (Gerente) funcionario1;
Funcionario funcionario2 = new Funcionario();
Gerente gerente2 = (Gerente) funcionario2; // erro em tempo de execução !!!
```

Polimorfismo

Quando, numa hierarquia de classes, uma mesma assinatura de método se comporta de formas diferentes, dependendo do objeto instanciado a partir de uma classe da hierarquia.



OS CONCEITOS RELACIONAIS

Polimorfismo

```
//Java
class Parto extends Procedimento {
    Medico[] medicos = new Medico[]
                        {new Anestesista(), new Obstetra(), new Pediatra()};
    void realizarParto() {
        for (int i = 0; i < medicos.length; i++) {
            Medico medico = medicos[i];
            medico.operar();
        }
    }
}
```

Polimorfismo

```
class Anestesista extends Medico {
    @Override
    void operar() {// passos a serem seguidos para aplicar a anestesia}
}
class Obstetra extends Medico {
    @Override
    void operar() {// passos a serem seguidos para realizar o parto em si}
}
class Pediatra extends Medico {
    @Override
    void operar() {// passos a serem seguidos para averiguar a saúde do recém-nascido}
}
```

OS CONCEITOS RELACIONAIS

Sobrescrita

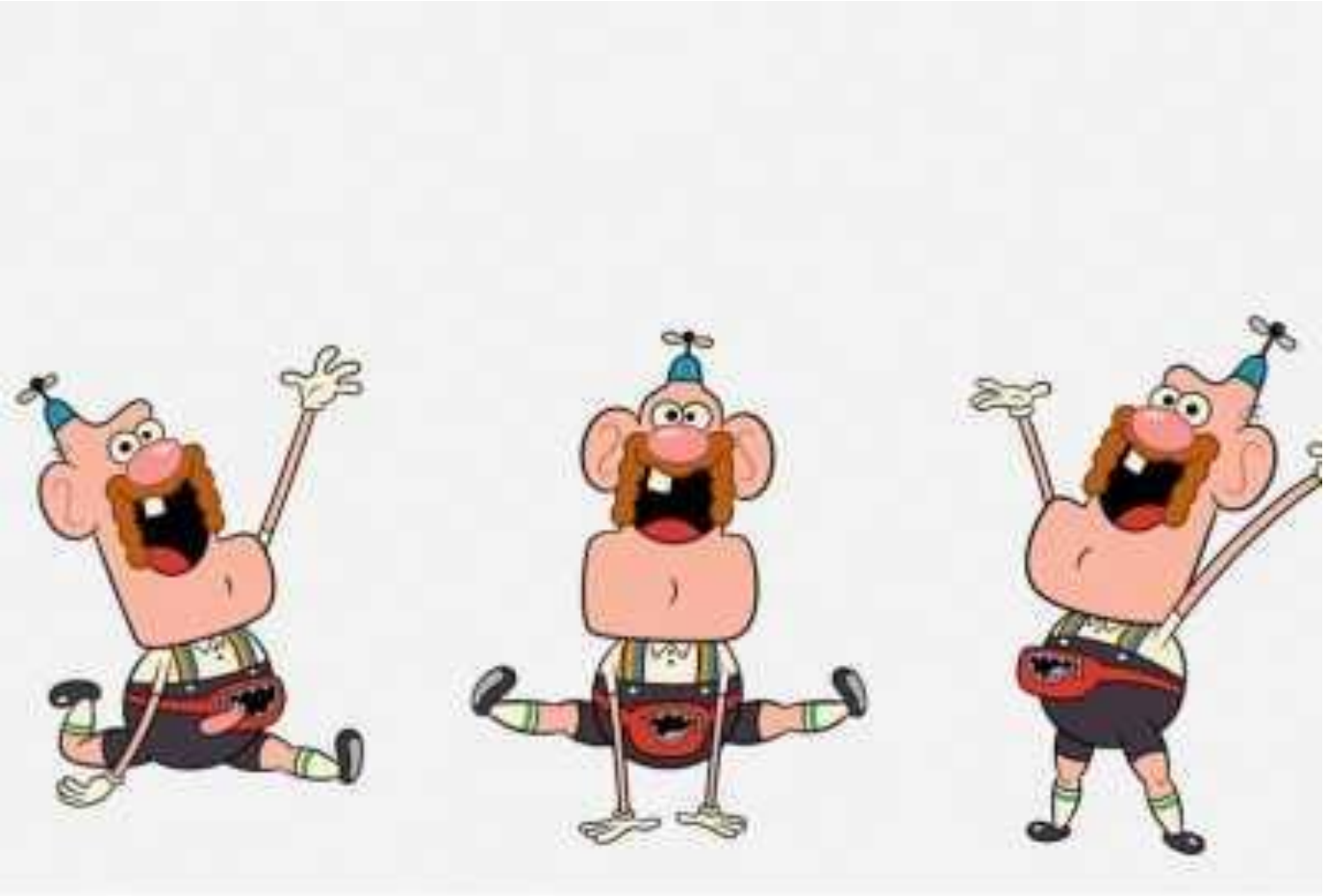
Quando a implementação de um método é redefinida na classe que o herdou.


```
//Java
class Anestesista extends Medico {
    void operar() {
        // calcular quantidade de anestésico
        // esterilizar local de aplicação
        // aplicar injeções
        // monitorar sensibilidade
        // ...
    }
}

class ResidenteAnestesista extends Anestesista {
    @Override
    void operar() {
        // calcular quantidade de anestésico
        // esterilizar local de aplicação
    }
}
```

OS CONCEITOS RELACIONAIS

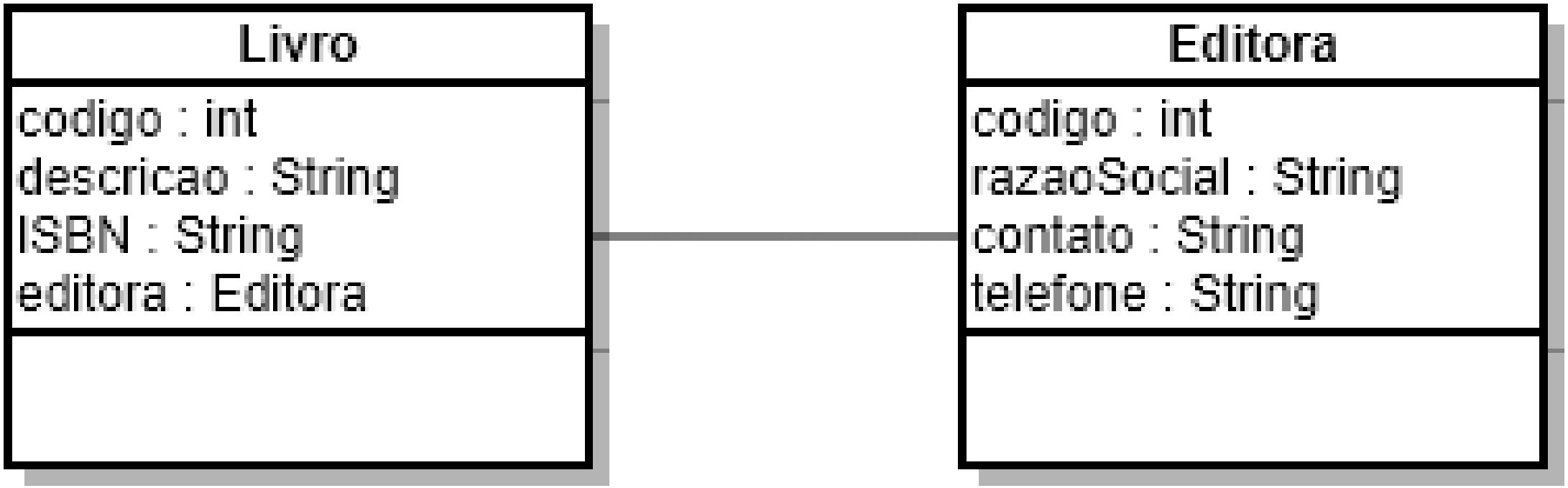
Exercícios



OS CONCEITOS RELACIONAIS

Associação

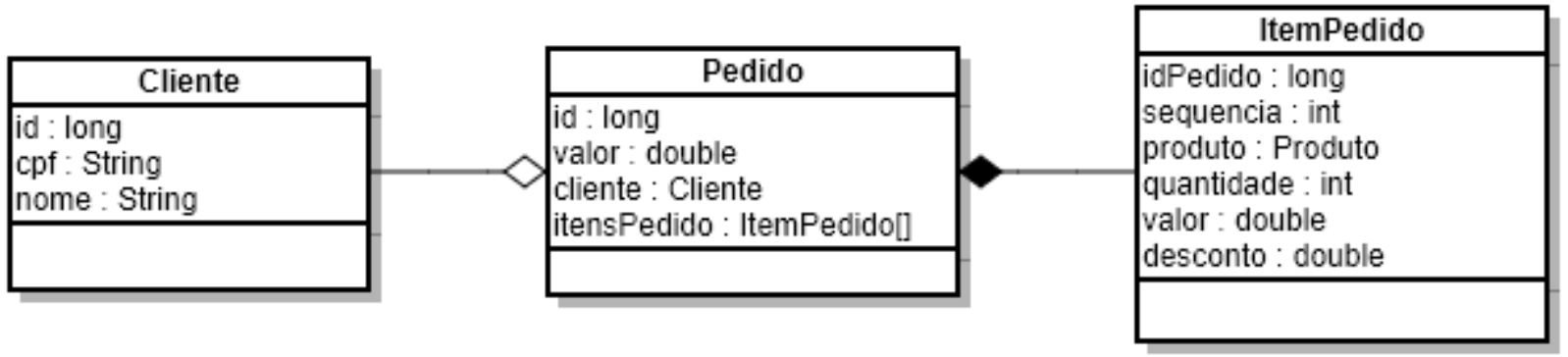
A Associação é um tipo de relacionamento entre classes/objetos, em que estes recorrem a outros para representar de forma completa o conceito na qual se destinam.



OS CONCEITOS RELACIONAIS

Tipos de Associação

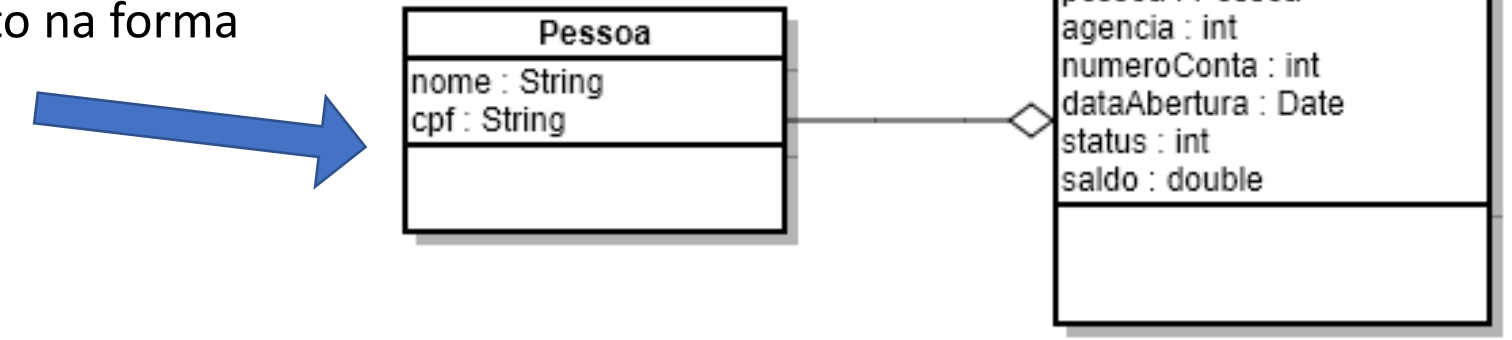
Associação Estrutural



Sua característica é a associação ocorrer na estrutura de dados da classe.

Tipo Composição - tem relacionamento na forma "parte todo“

Tipo Agregação - não tem relacionamento na forma "parte todo"

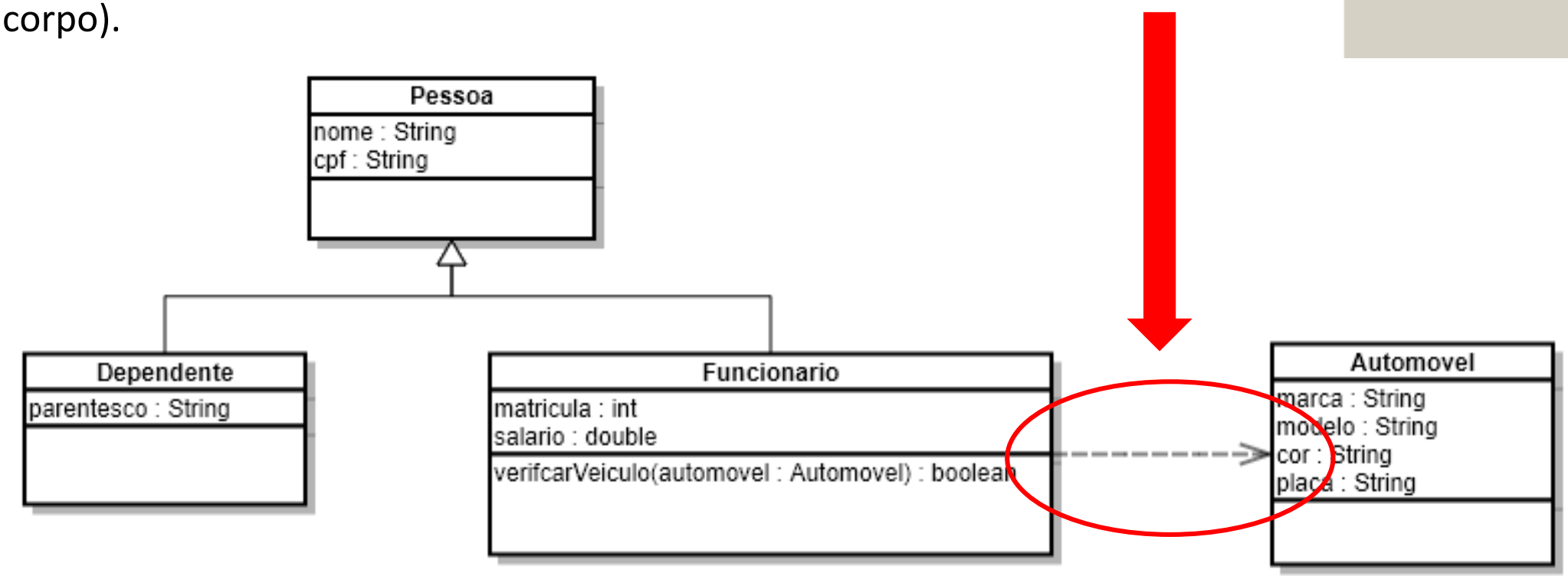


OS CONCEITOS RELACIONAIS

Associação Comportamental

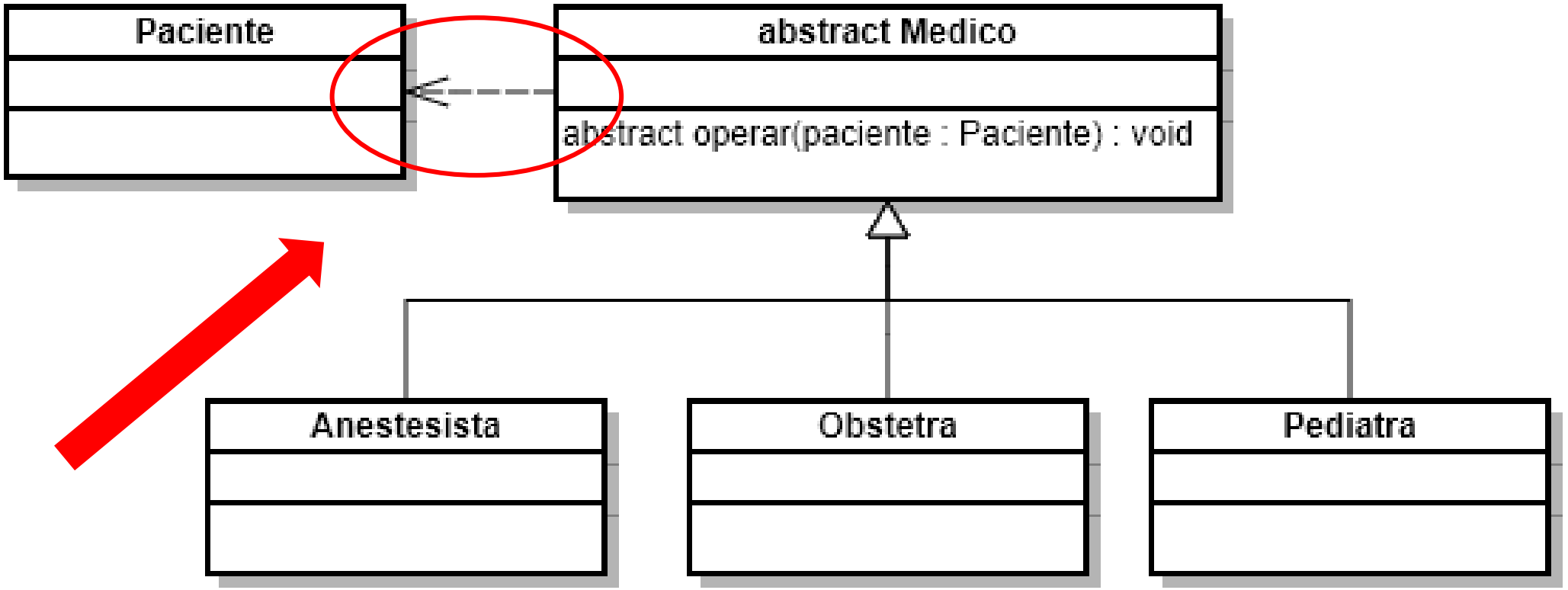
Dependência

Sua característica é a associação ocorrer nos métodos (objetos de parâmetros ou objetos instanciados no corpo).



OS CONCEITOS RELACIONAIS

Associação Comportamental

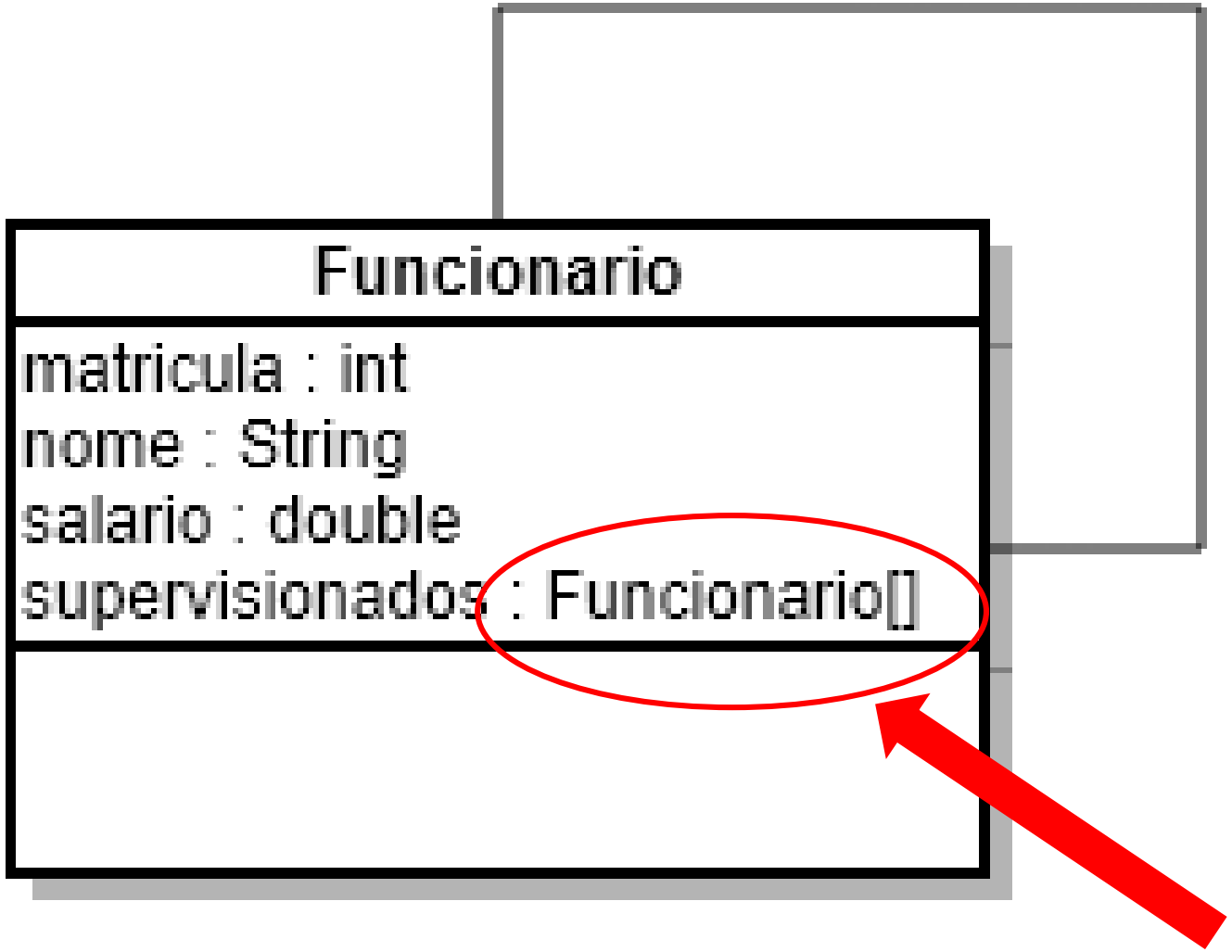


OS CONCEITOS RELACIONAIS

Características de uma Associação

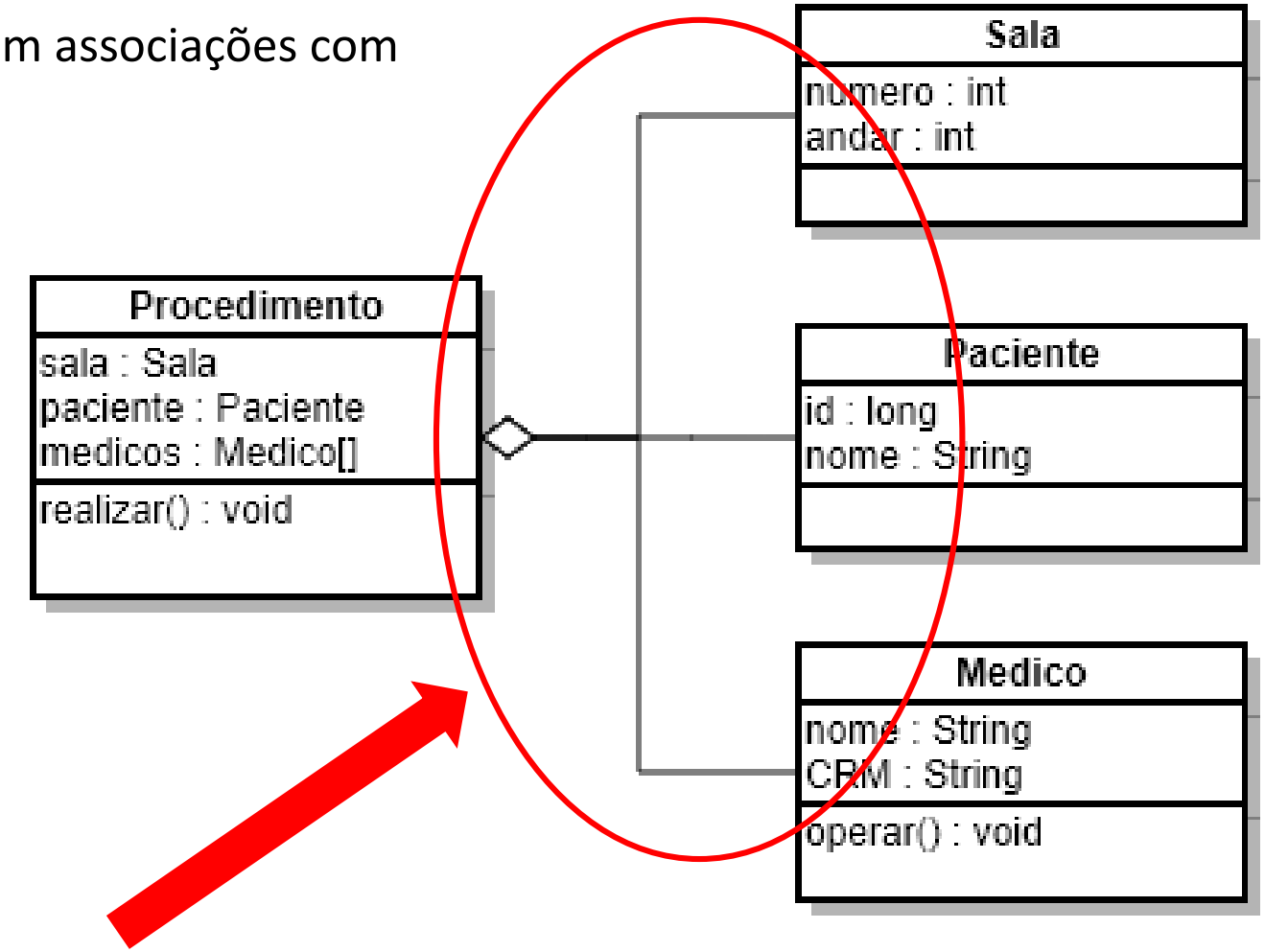
Associação Unária

Quando uma classe tem um atributo que é do seu próprio tipo.



Associação Múltipla

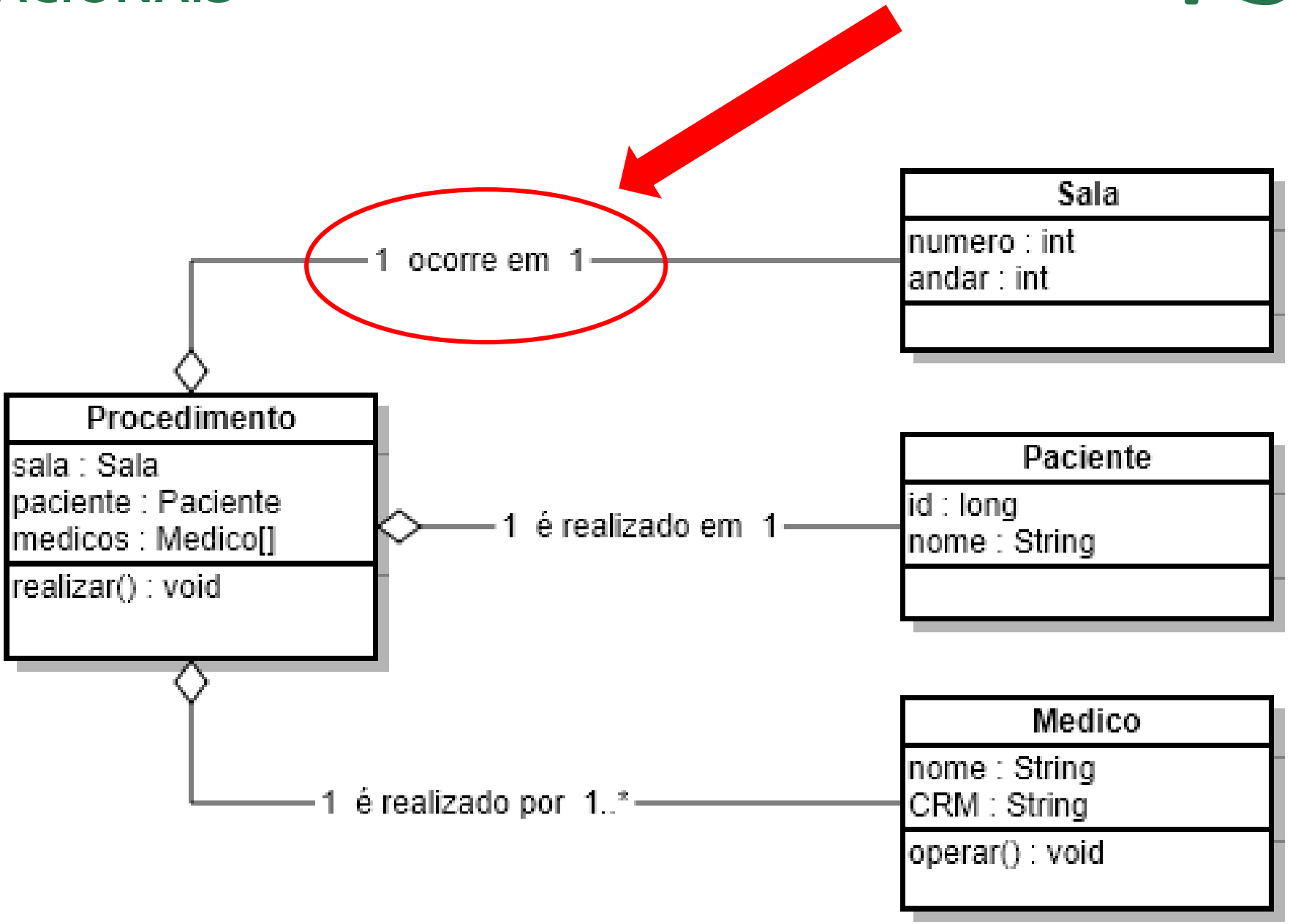
Quando uma classe tem associações com duas ou mais classes.



OS CONCEITOS RELACIONAIS

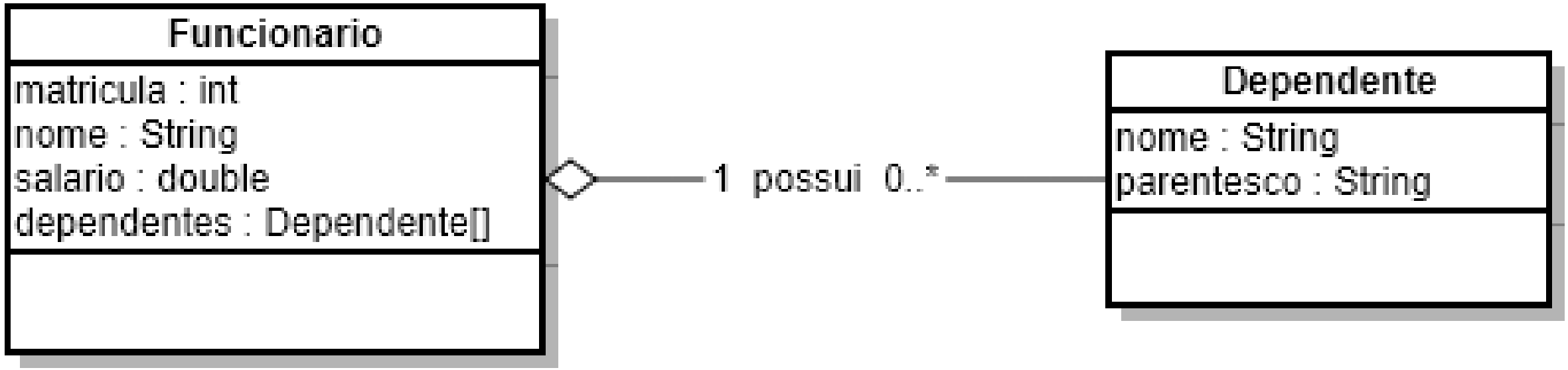
Cardinalidade

É a quantidade de objetos permitidas na associação. Pode ser fixa ou variável.



OS CONCEITOS RELACIONAIS

Cardinalidade

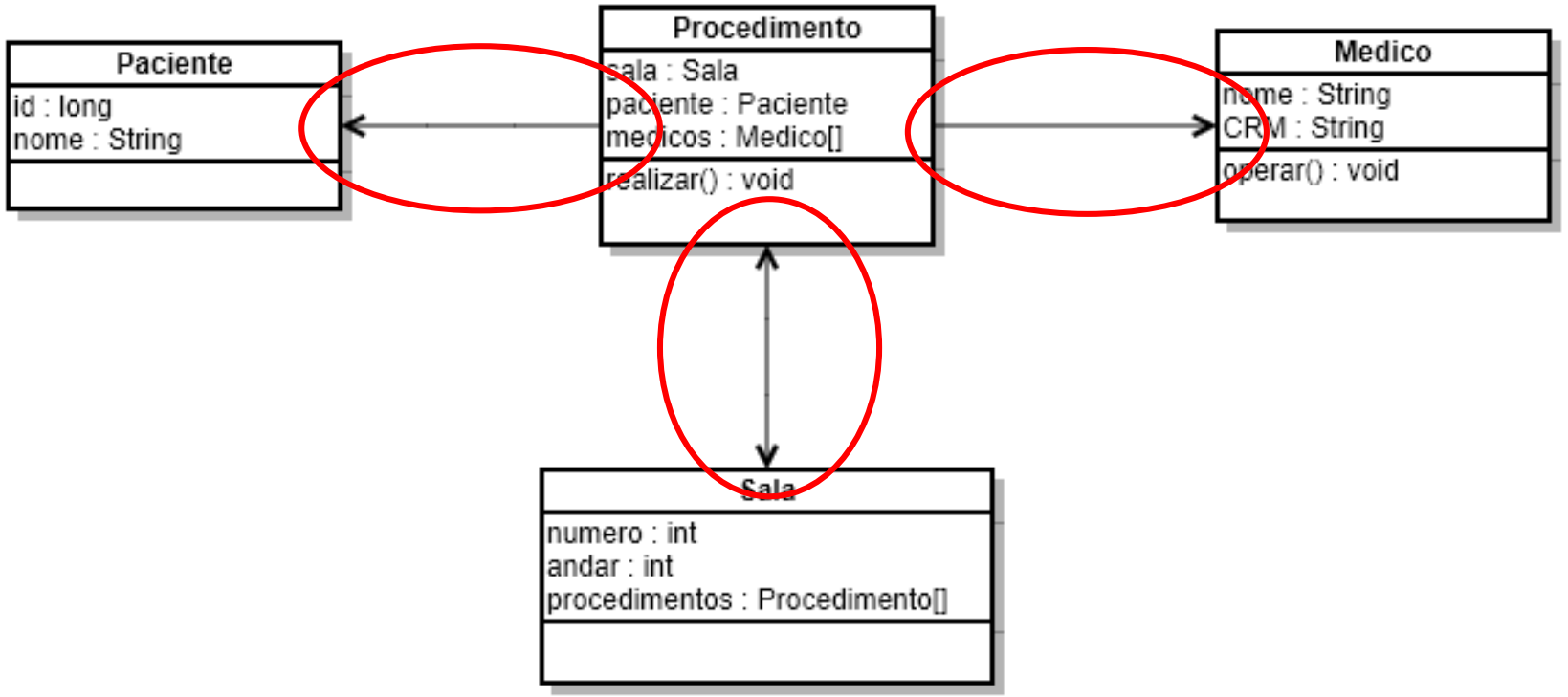


OS CONCEITOS RELACIONAIS

Navegabilidade

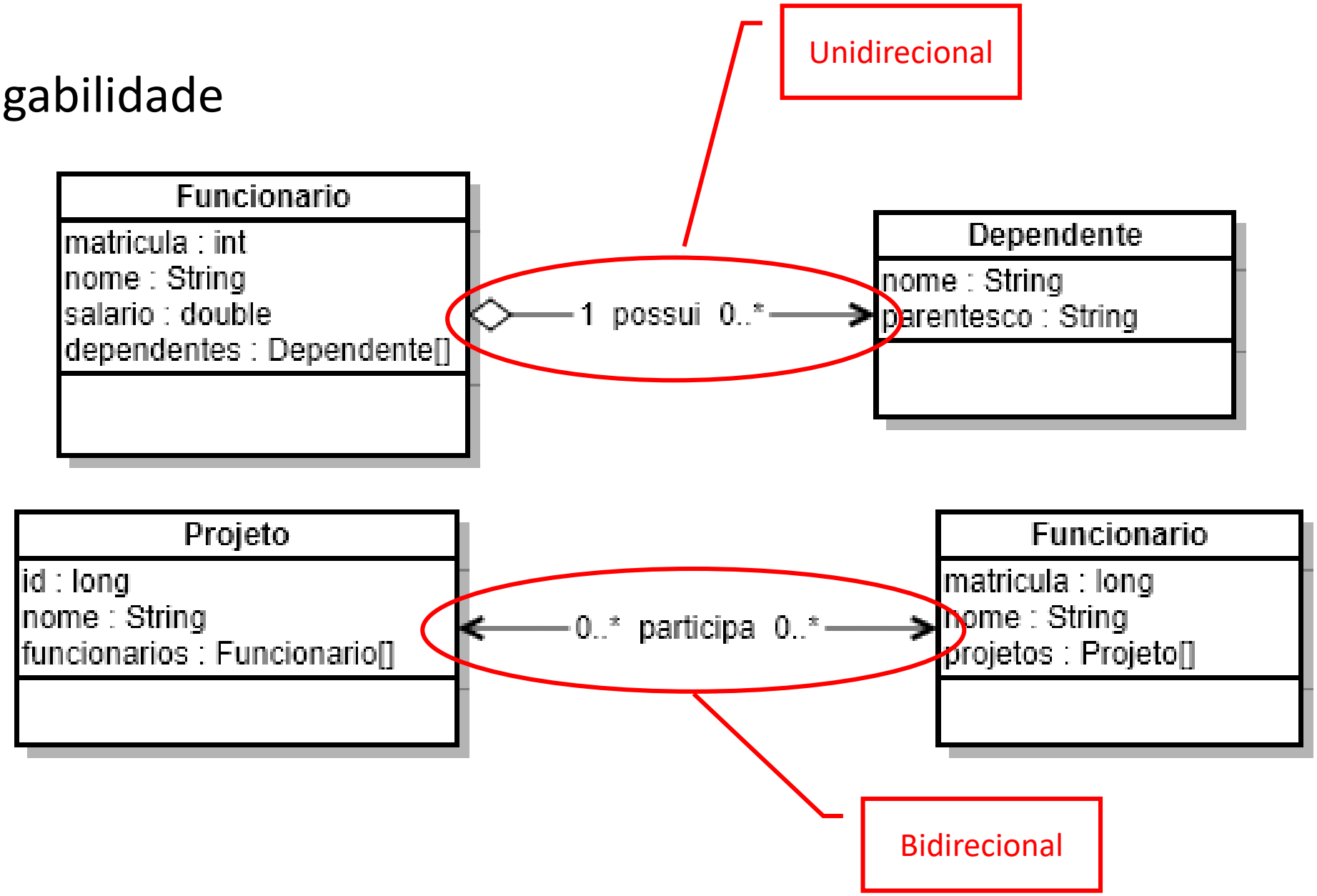
Unidirecional - Associação acontece somente de um lado.

Bidirecional - Associação acontece nos dois lados.



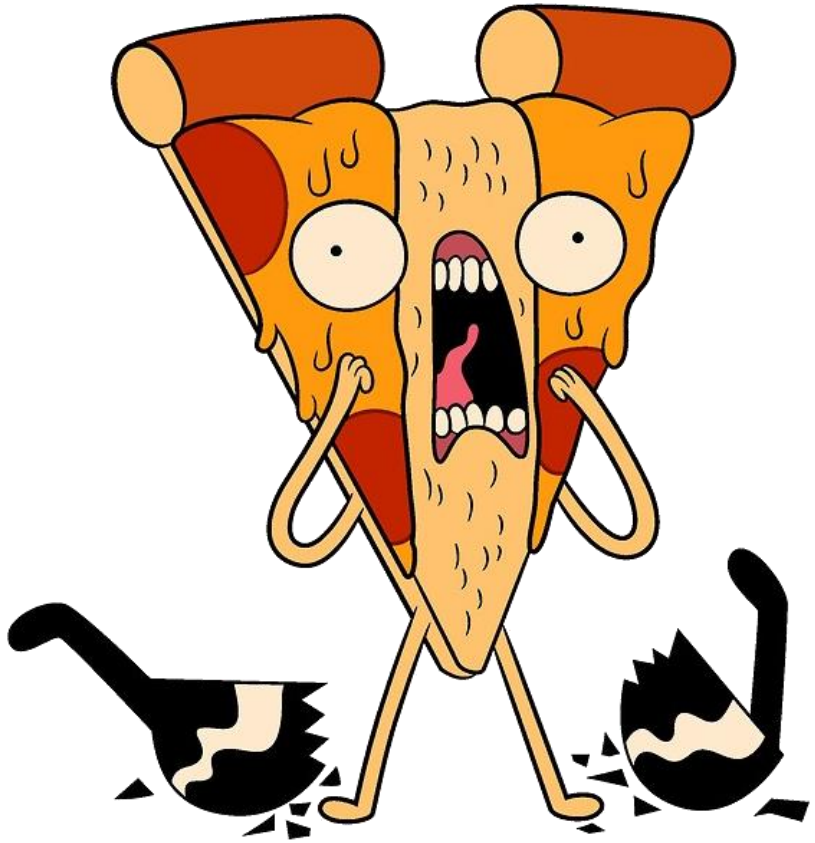
OS CONCEITOS RELACIONAIS

Navegabilidade



OS CONCEITOS RELACIONAIS

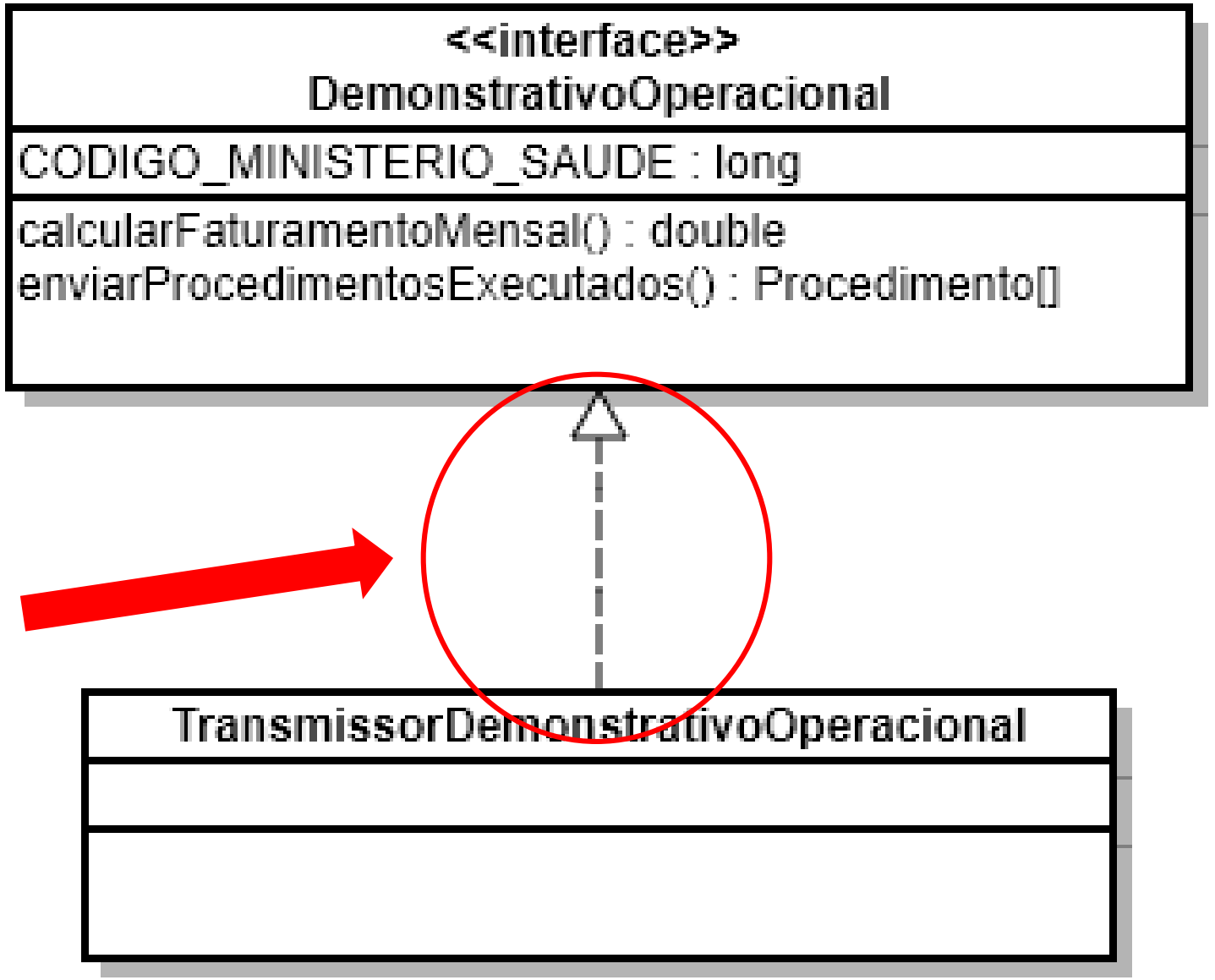
Exercícios



OS CONCEITOS RELACIONAIS

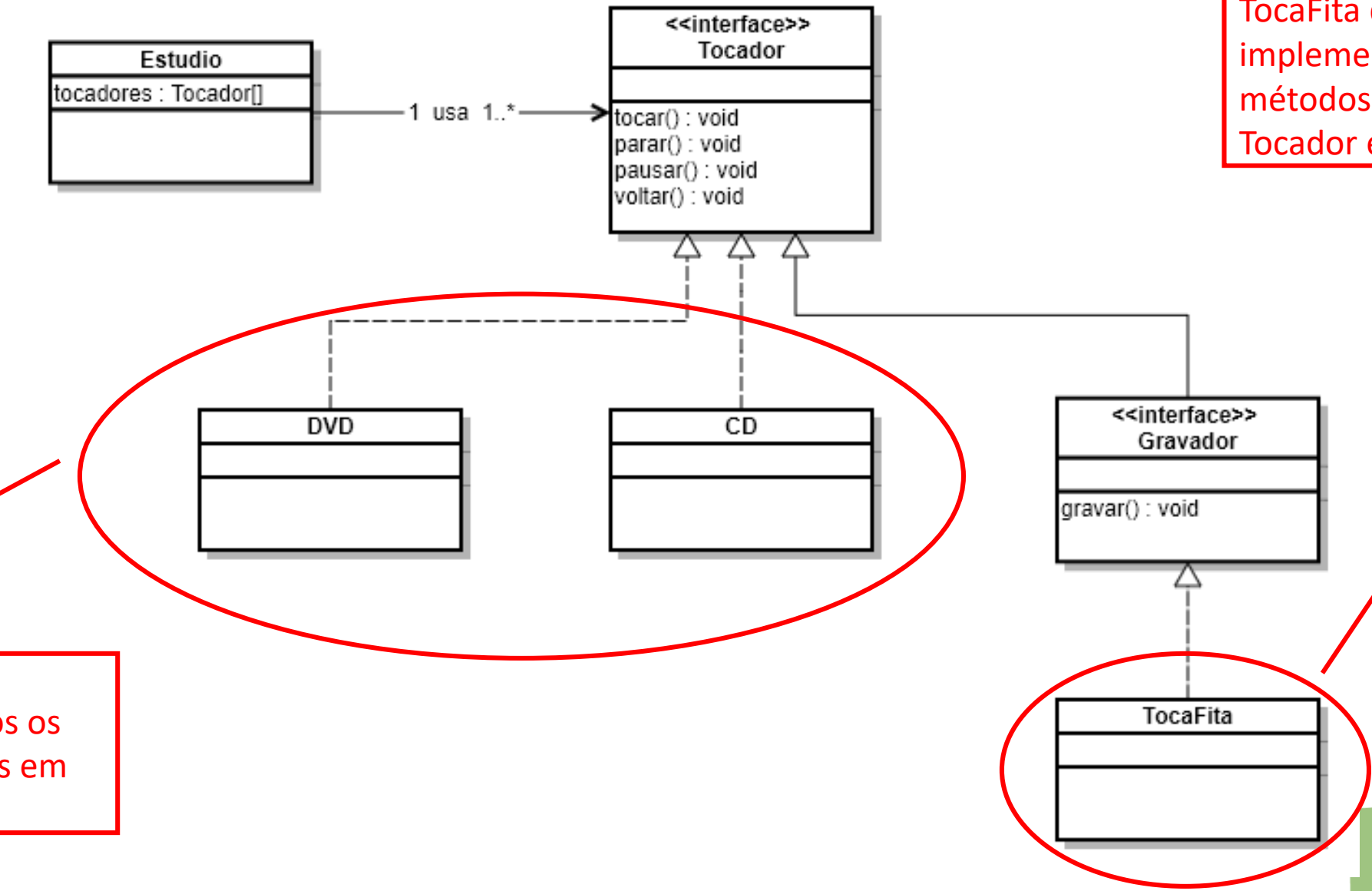
Interface

A Interface define um contrato que deve ser seguido pela classe que a implementa.



OS CONCEITOS RELACIONAIS

Interface



OS CONCEITOS RELACIONAIS

Para refletir...

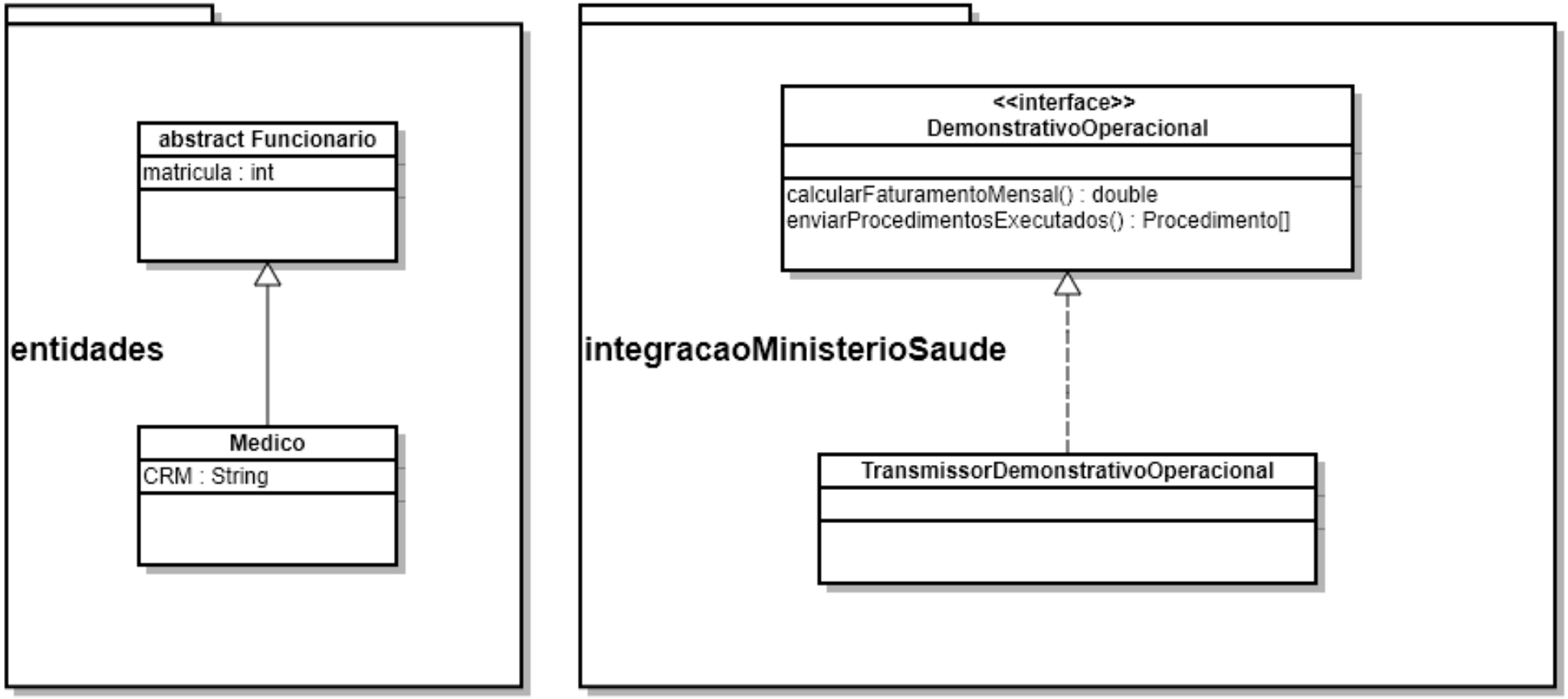
07

Os Conceitos Organizacionais

OS CONCEITOS ORGANIZACIONAIS

Pacotes

É uma organização física ou lógica criada para separar classes com responsabilidades distintas.



OS CONCEITOS ORGANIZACIONAIS

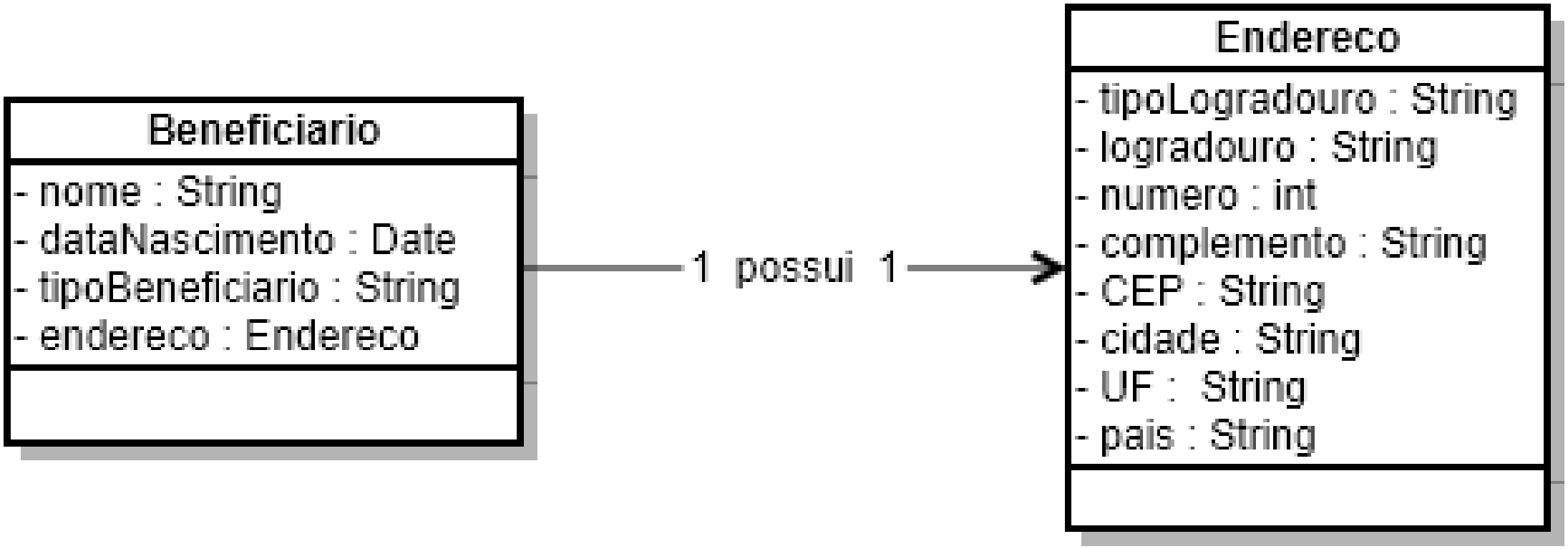
Visibilidades (ou modificadores de acesso)

Controlam o acesso (manipulação) de classes, atributos e métodos.

Tipos de visibilidades: privado, protegido e público

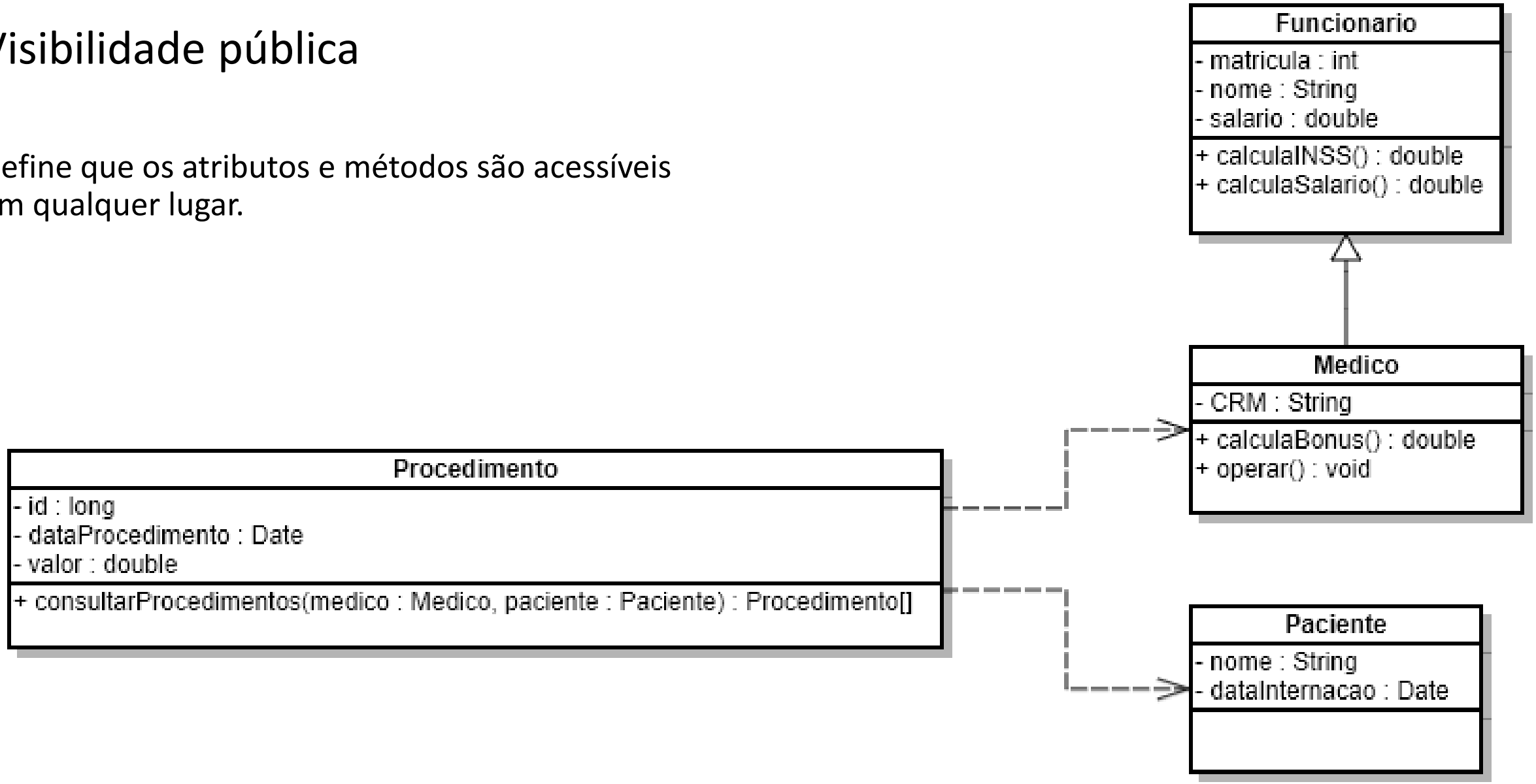
Visibilidade privada

Define que os atributos e métodos só podem ser manipulados apenas no local de sua definição.



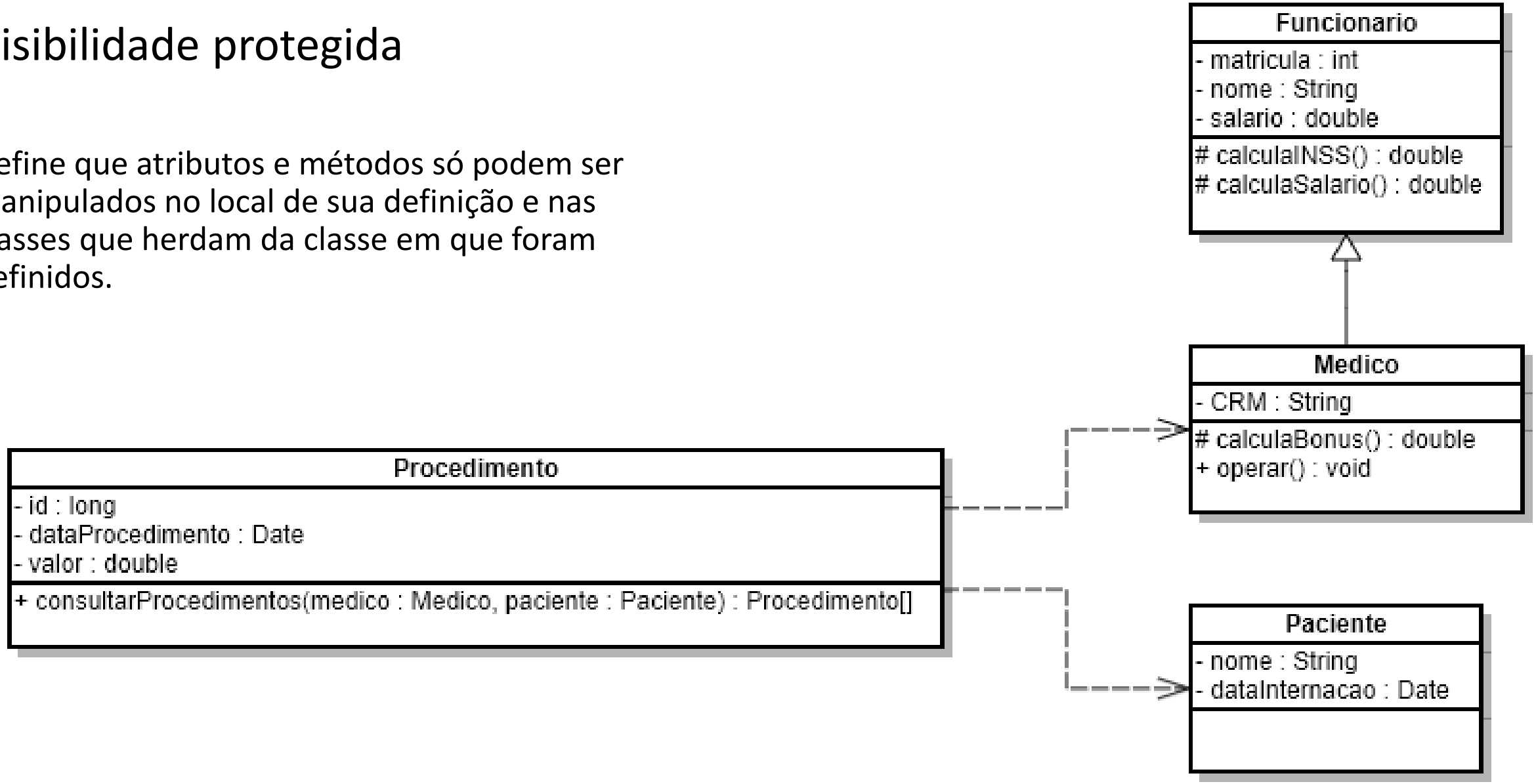
Visibilidade pública

Define que os atributos e métodos são acessíveis em qualquer lugar.



Visibilidade protegida

Define que atributos e métodos só podem ser manipulados no local de sua definição e nas classes que herdam da classe em que foram definidos.



OS CONCEITOS ORGANIZACIONAIS

Para refletir...

UML Gráficos

Associação simples

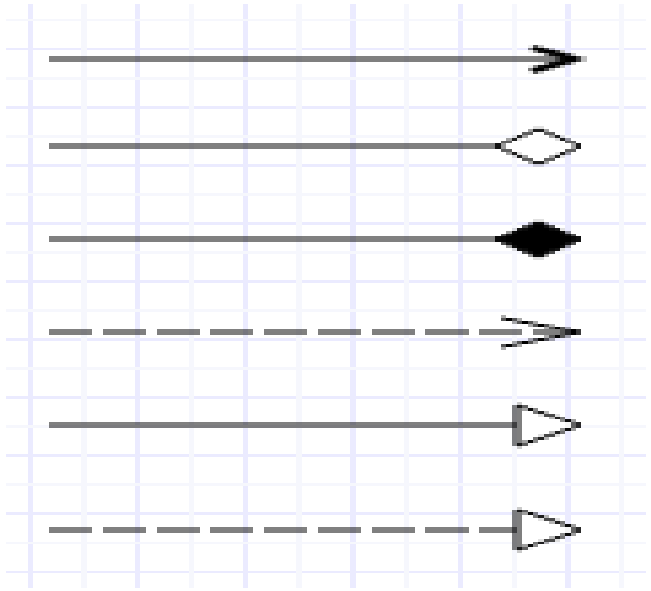
Agregação

Composição

Dependência

Herança

Interface

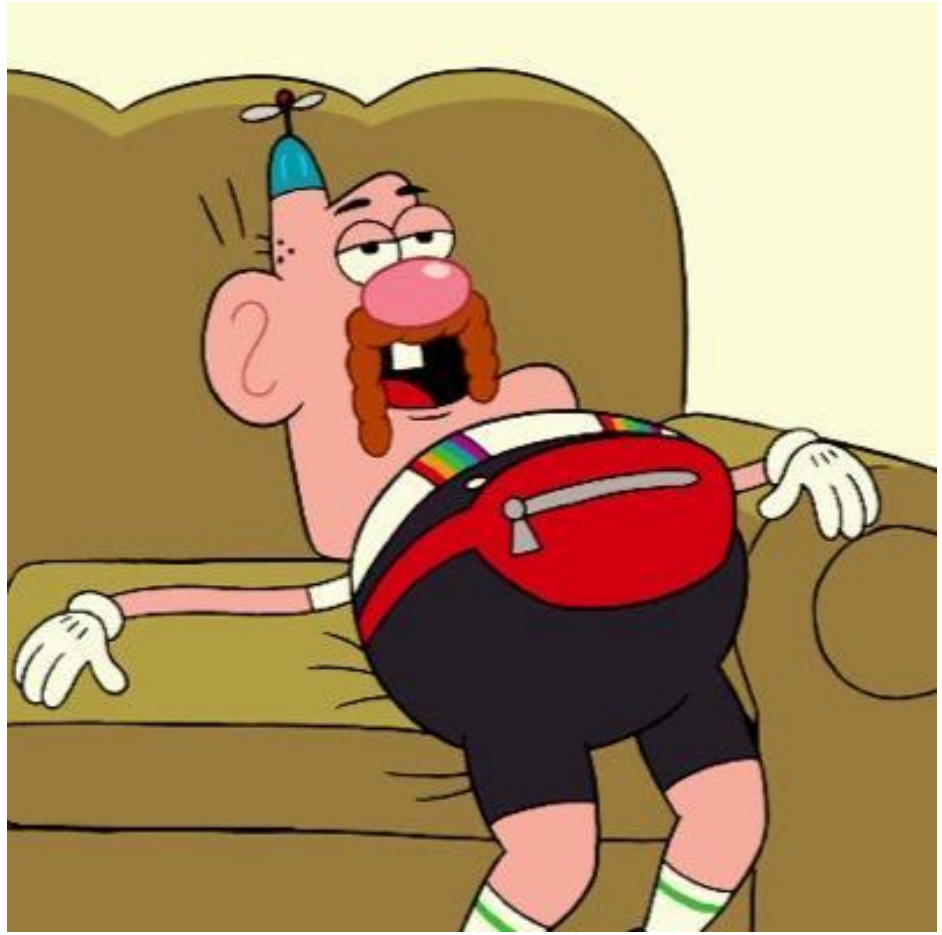


Acesso privado -

Acesso público +

Acesso protegido #

Exercícios



04-uml-modelagem-ecommerce.docx

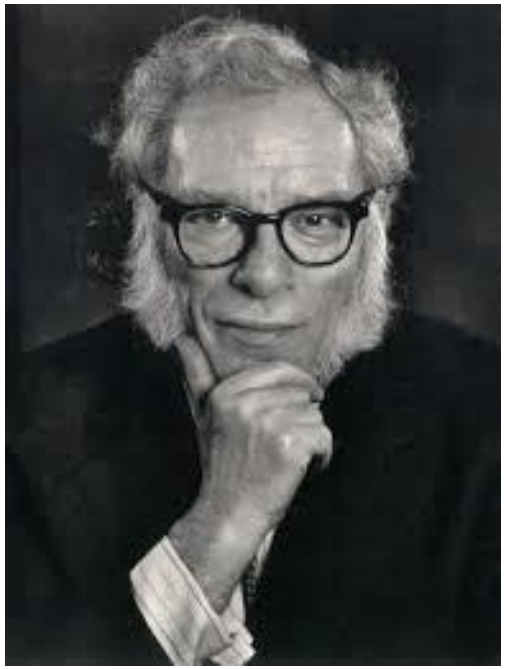
MOMENTO “MÁRIO SÉRGIO CORTELLA”

"Os programas devem ser escritos para as pessoas lerem e, incidentalmente, para as máquinas executarem."

- Harold Abelson, in “Structure and Interpretation of Computer Programs”

“Parte da desumanidade do computador é que, uma vez que é programado com competência e funcionando sem problemas, é completamente honesto.”

- Isaac Asimov



REFERÊNCIAS BIBLIOGRÁFICAS

Livro “Orientação a Objetos – Aprenda seus conceitos e suas aplicações de forma efetiva”, Carvalho, T. L., Ed. Casa do Código, 2020.

[1] https://en.wikipedia.org/wiki/Programming_paradigm

[2] https://en.wikipedia.org/wiki/Object-oriented_programming

[3] https://en.wikipedia.org/wiki/Sketchpad

[4] https://en.wikipedia.org/wiki/Lisp_(programming_language)

[5] https://en.wikipedia.org/wiki/ALGOL

[6] https://pt.wikipedia.org/wiki/Orientação_a_objetos

[7] https://pt.wikipedia.org/wiki/Classe_(programação)

AVALIAÇÃO

Concluir a modelagem de classes de um sistema hospitalar

Enunciado: ORIENTACAO_A_OBJETOS - Avaliação.docx

Ferramenta Gliffy (extensão do Chrome)

FIM

Obrigado!

☺