

INSTITUTO MAUÁ DE TECNOLOGIA



Linguagens I

Diagrama de Classes

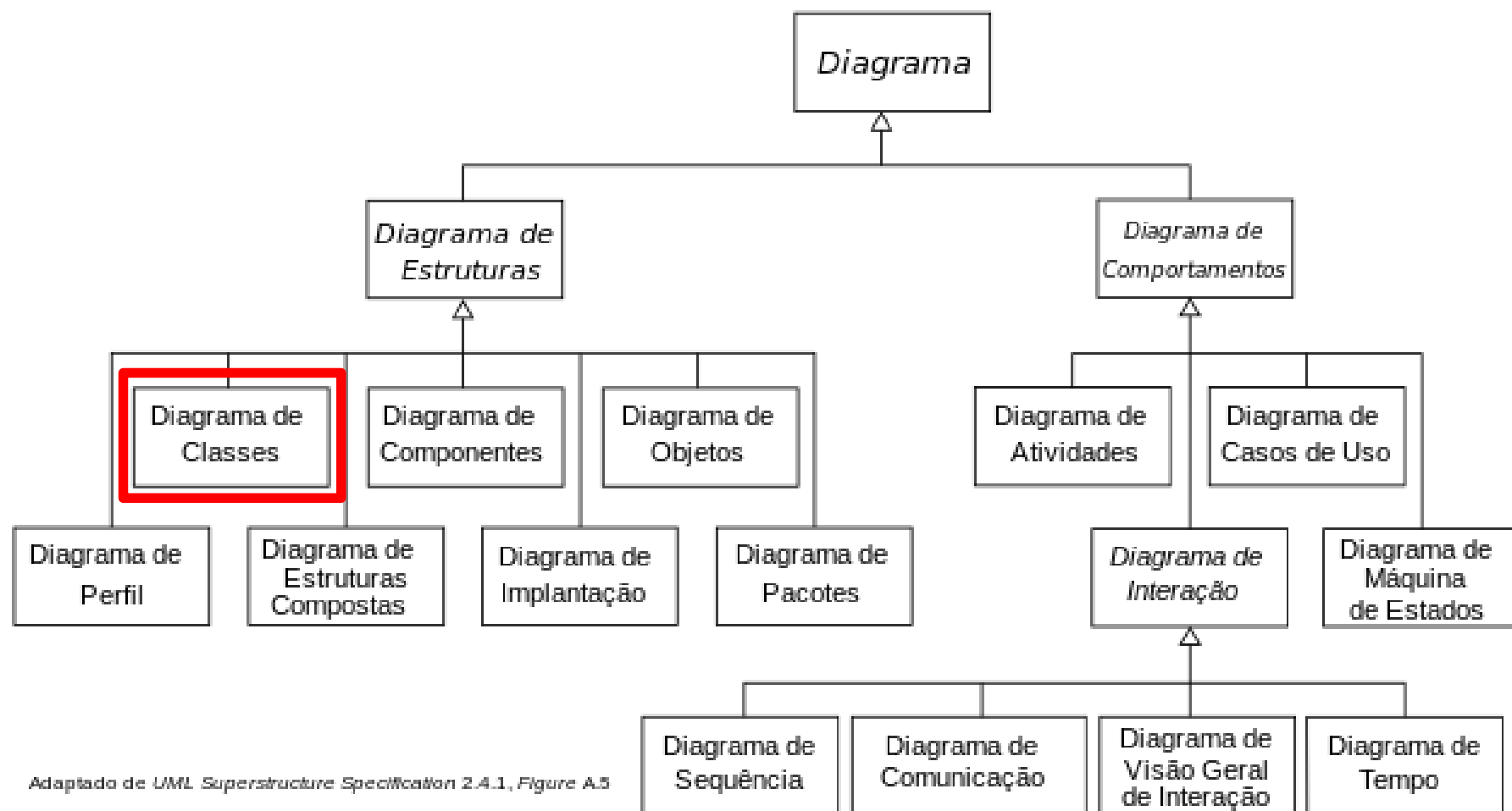
Profº. Tiago Sanches da Silva

Diagrama de Classes

A UML é uma linguagem-padrão para a elaboração da estrutura de projetos de software. Ela poderá ser empregada para a visualização, a especificação, a construção e a documentação de artefatos que façam uso de sistemas complexos de software.

Em outras palavras, na área de Engenharia de Software, a Linguagem de Modelagem Unificada (do inglês, **UML - Unified Modeling Language**) permite representar um sistema de forma padronizada (com intuito de facilitar a compreensão e pré-implementação).


UML possui 15 tipos de diagramas, divididos em duas grandes categorias: Estruturais (7 diagramas) e Comportamentais (8 diagramas). Sete tipos de diagramas representam informações estruturais, e os outros oito representam tipos gerais de comportamento



Adaptado de UML Superstructure Specification 2.4.1, Figure A.5

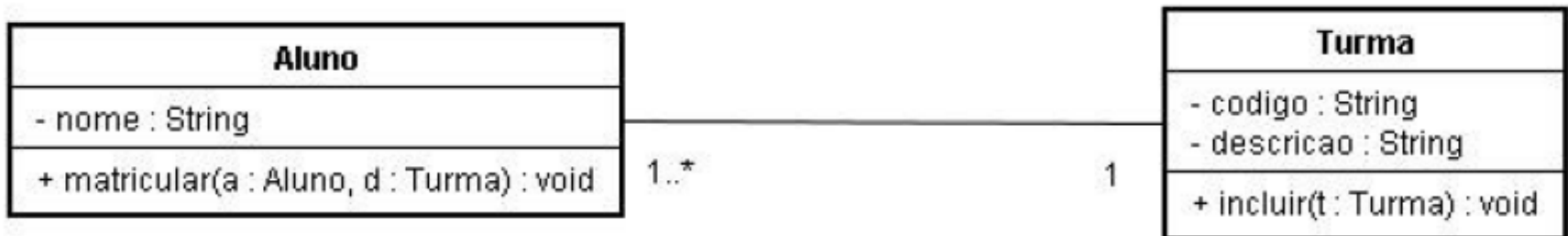
Diagrama de Classes

O Diagrama de Classes apresenta como as classes interagem entre si e qual a responsabilidade de cada classe na realização das operações solicitadas pelos atores.

- **Nome**
 1. Sempre deve ser iniciado com letra maiúscula
 - **Atributos**
 1. Visibilidade ou nível de encapsulamento
 2. Nome (deve ser iniciado com letra minúscula)
 3. Tipo de dados
 - **Operações ou métodos**
 1. Visibilidade ou nível de encapsulamento
 2. Nome (deve ser iniciado com letra minúscula)
 3. Lista de parâmetros (se houver)
 4. Tipo de retorno
- 

- Associações entre si
 1. Multiplicidade

Exemplo



Visibilidade ou nível de encapsulamento

- - private (privado)
- # protected (protegido)
- + public (público)
- (em branco) default (pacote)

Nome

- Demonstram as características dos objetos

Tipo de dados

- São os mesmos tipos usados em Java: String, boolean, int, float, double, Date, etc...

Visibilidade ou nível de encapsulamento

- Os mesmos usados para os atributos

Nome

- O nome do método deve expressar a ação que realiza, por exemplo `incluirAluno()`. Não deve possuir espaços e nem começar com dígitos

Lista de parâmetros

- Deverá vir entre parênteses e separados por vírgula.

Tipo de retorno

- Informa que tipo de dado o método deverá retornar após a sua execução.
- Se o método não retornar nada, deverá ser usada a palavra “void” no tipo de retorno.

Associação entre classes

Para representar o fato de que objetos podem se relacionar uns com os outros, utiliza-se a associação.

Representa que duas classes possuem uma ligação (link), significando por exemplo que elas "conhecem uma a outra".

Associação entre classes

Representada através de um segmento de reta ligando as classes cujos objetos se relacionam.

Nome da Associação

- Quando usado, deverá ser escrito junto à linha que representa a associação, normalmente um verbo (não é obrigatório).

Multiplicidades

- Cada associação em um diagrama de classes possui duas multiplicidades, uma em cada extremo da linha de associação.

Navegabilidade ou direção de leitura

- Indica como a associação deve ser lida

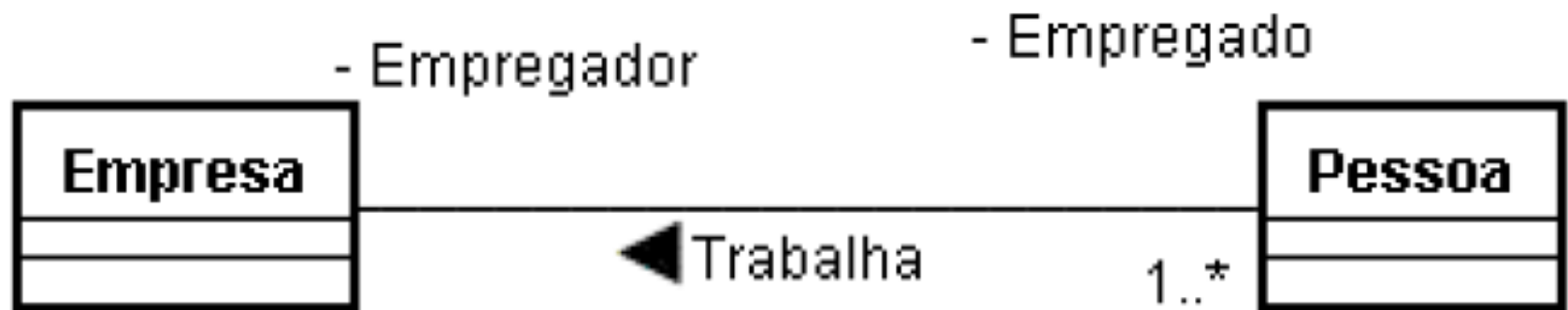


Associação entre classes

Opções de multiplicidade

Nome	Simbologia
Apenas Um	1..1 (ou 1) (ou em branco)
Zero ou Muitos	0..* (ou *)
Um ou Muitos	1..*
Zero ou Um	0..1
Intervalo Específico	$l_i..l_s$

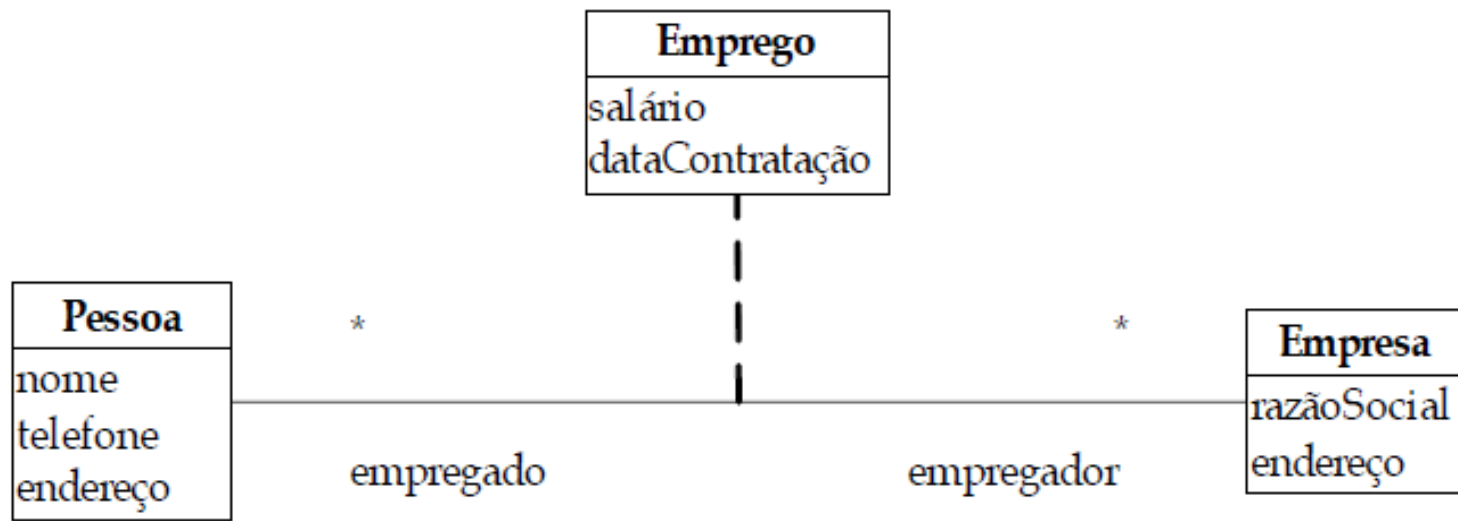
Associação entre classes



Classe Associativa

É uma classe que está ligada a uma associação, ao invés de estar ligada a outras classes.

É normalmente necessária quando duas ou mais classes estão associadas, e é necessário manter informações sobre esta associação (histórico).



Agregação

É um caso especial de associação e, consequentemente, multiplicidades, nome da associação e papéis, podem ser usados normalmente.

Utilizada para representar conexões que guardam uma relação todo-parte entre si.

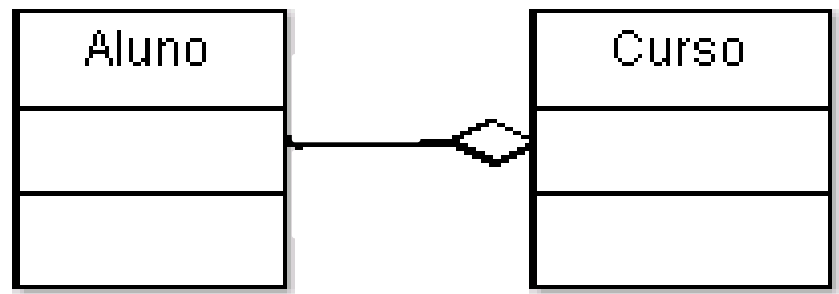
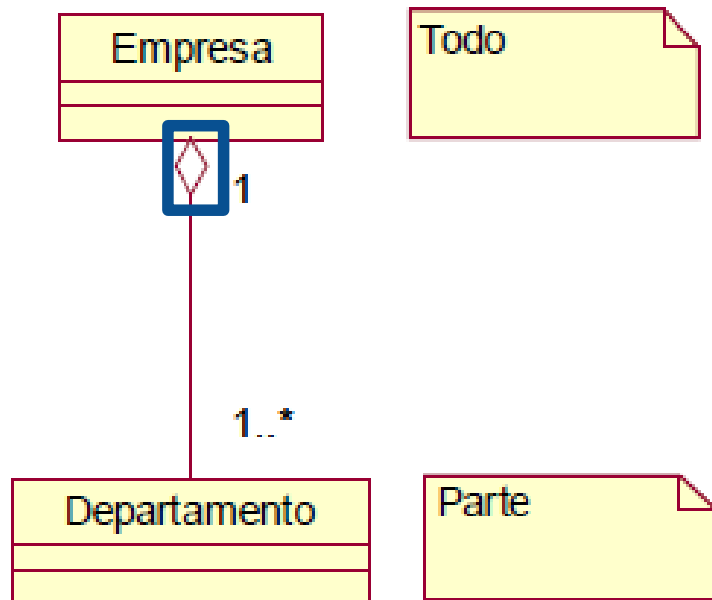
Em uma agregação, um objeto está contido no outro, ao contrário de uma associação.

Onde se puder usar uma agregação, uma associação também poderá ser utilizada.

Representado com um losango na classe agregadora.



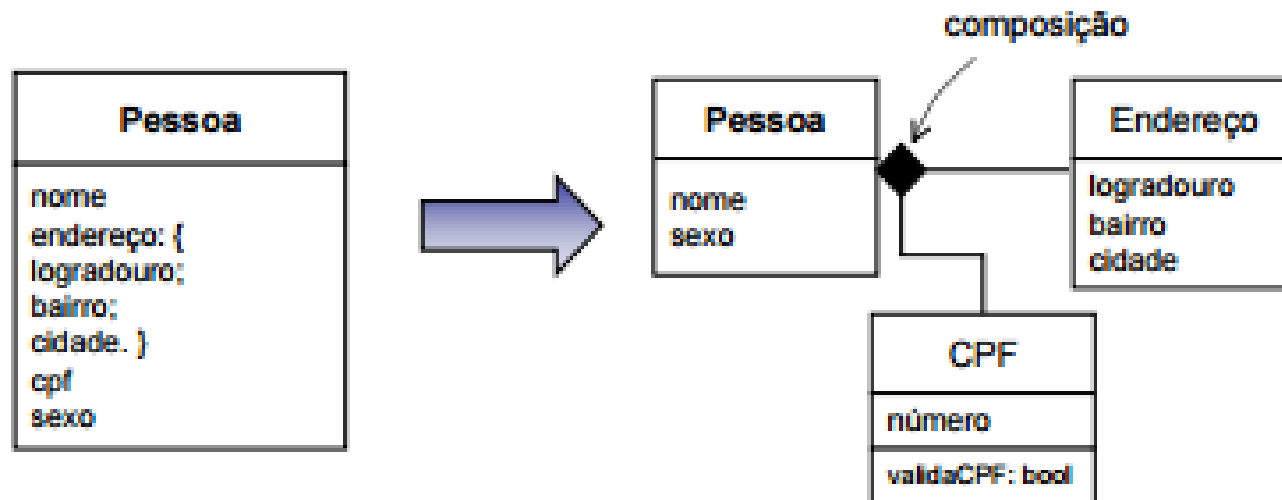
Agregação



Composição (“...pertence exclusivamente a ...”)

A Agregação Simples possui uma variante, a Composição. Essa variação adiciona um grau de importância semântica à relação. Ela define a relação de posse ou possessão. Isso significa que um objeto da classe pertence apenas e exclusivamente ao objeto da outra classe.

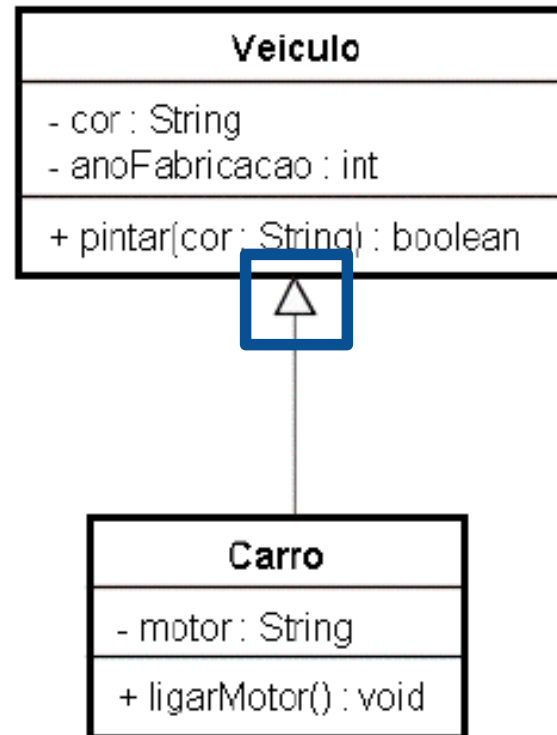
Como a classe mais importante (ou todo) é responsável pela disposição das partes. Isso significa que ela precisa gerenciar a criação e destruição das partes que a compõe.



Especialização/Generalização (extends)

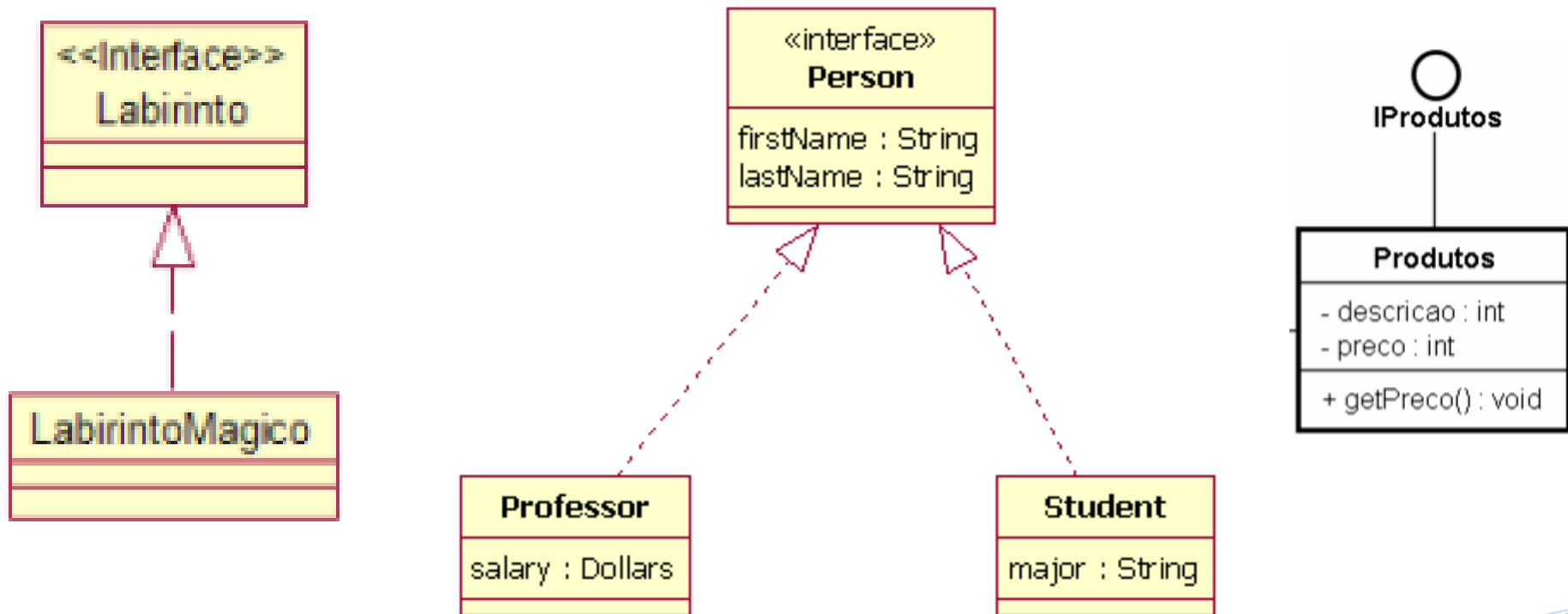
Seu objetivo é identificar classes-mãe, chamadas gerais e classes-filhas, chamadas especializadas.

Basicamente utilizado para representar Herança.



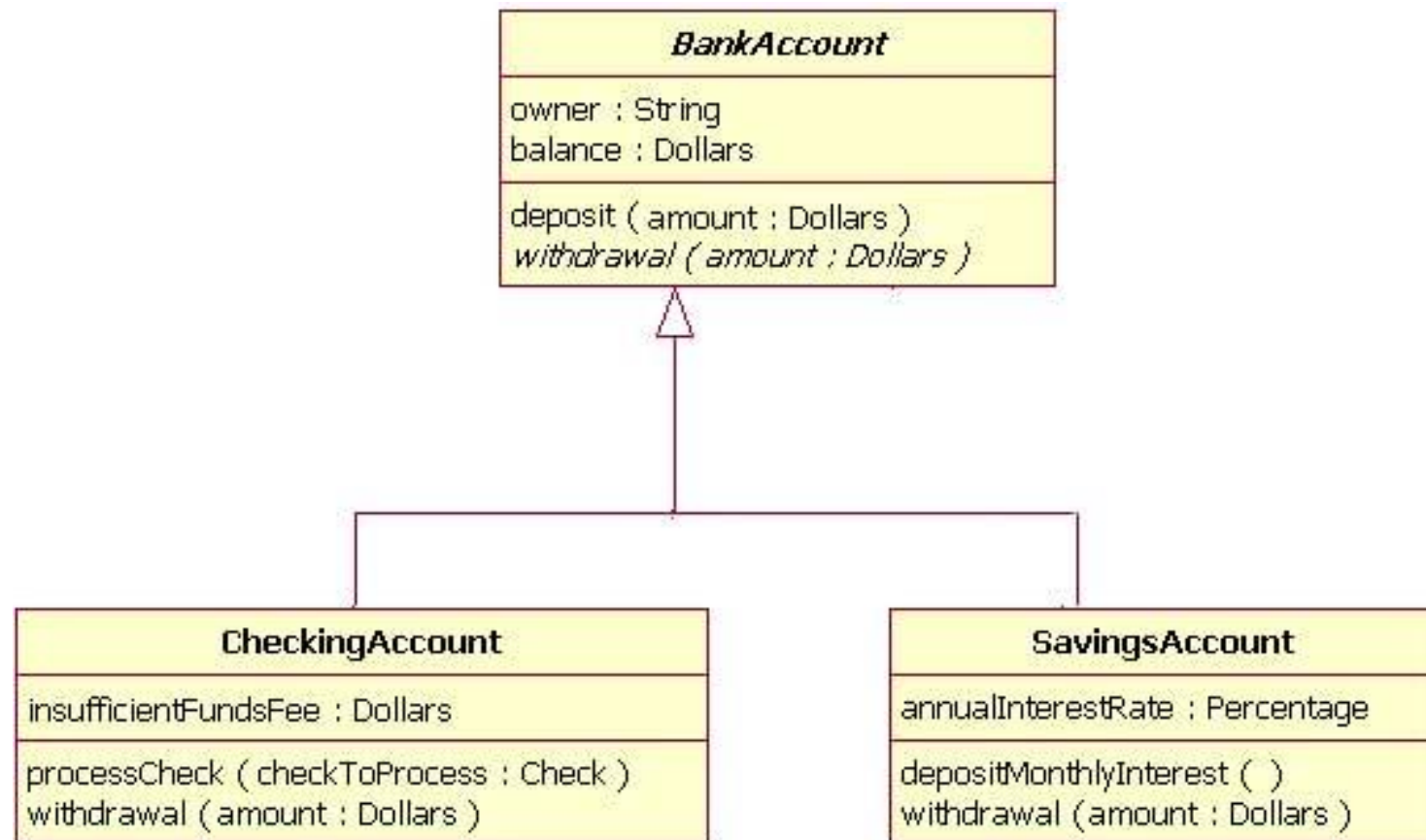
Interface (implements)

Utilizado para indicar que uma classe implementa uma interface. Representação similar a herança só que utilizando a linha tracejada. Também é utilizado <<interface>> no topo do retângulo.



Classes abstratas

São representadas como classes concretas, porém seus nomes são escritos em *ITALICO*.



Resumo relação entre classes



Dependência: a classe precisa saber sobre a outra classe para usar os objetos desta classe. Por exemplo, um método da sua classe pode necessitar utilizar algum método da classe **Math** do Java.

Exercício

Crie o diagrama de classes dos exercícios já feitos em sala:

- Conta bancaria
- Concessionária
- Exercício Interface

Procure por tutoriais de como utilizar o software StarUML, exemplos:

[http://staruml.sourceforge.net/docs/user-guide\(en\)/ch05_2.html](http://staruml.sourceforge.net/docs/user-guide(en)/ch05_2.html)

<https://www.youtube.com/watch?v=RPSkYWra8nQ>



Perguntas?

Referências

- Deitel
- Modelagem Visual de objetos com UML: Lucelia