PROGRAMAÇÃO ORIENTADA A OBJETOS

INTRODUÇÃO AO UML

PROF. BRUNO GÓIS MATEUS

(BRUNOMATEUS@UFC.BR)

ÍNDICE

- Introdução
- Unified Modeling Language
- Diagrama de classe

INTRODUÇÃO

INTRODUÇÃO

- Quando se fala de orientação a objetos, um porção de termos relacionados surgem:
  - Análise orientada a objetos
  - Design orientado a objetos
  - Análise e Design orientada a objetos
  - Programação orientada a objetos

ANÁLISE ORIENTADA A OBJETOS

- O que precisa ser feito ?
- Processo de olhar para o problema e identificar objetos e suas interações
- Como resultado temos um conjunto de requisitos
- Exemplo de um website, os visitantes precisam:
  - *revisar* seu **histórico**
  - *concorrer* a uma vaga de **emprego**
  - *navegar, comparar, e comprar* **produtos**

DESIGN ORIENTADA A OBJETOS

- Como deve ser feito ?
- Processo de converter os requisitos em uma especificação de implementação
  - Objetos devem ser nomeados
  - Comportamentos devem ser definidos
  - Iterações entre objetos

PROGRAMAÇÃO ORIENTADA A OBJETOS

- Converter o projeto em um programa que faz exatamente o esperado

NA PRÁTICA

- Em um mundo ideal essas três etapas de desenvolvimento ocorrem de forma separada
- Na prática, é quase impossível isso acontecer
  - No momento de projetar você encontra falhas na análise
  - Quando está programando você percebe que o projeto não está claro
- Por essas razões, atualmente são utilizados o modelos de **desenvolvimento iterativos**

UNIFIED MODELING LANGUAGE

UML

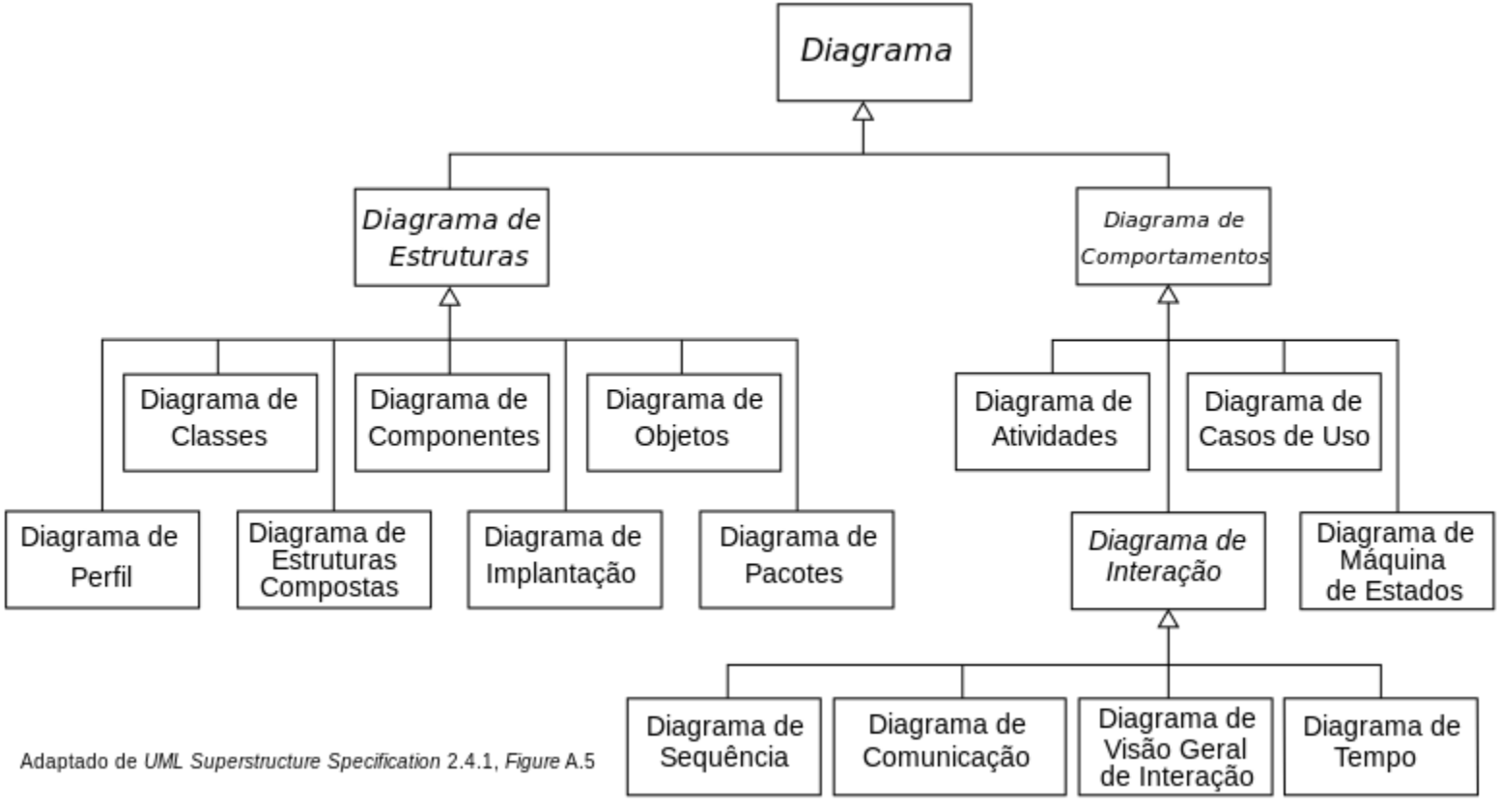
- A **Unified Modeling Language (UML)**, é um linguagem de modelagem utilizada na engenharia de software
- Foi criada para prover uma maneira padrão de visualização de projeto de software
- Foi criada em 1994-1995 por Grady Booch, Ivar Jacobson e James Rumbaugh
- O nascimento da UML está fortemente ligado a orientação a objetos

UML

- Permite que desenvolvedores visualizem os produtos de seus trabalhos em diagramas padronizados
- A UML também especifica significados, isto é, semântica
- Os objetivos da UML são:
  - Especificação
  - Documentação
  - Estruturação para sub-visualização
  - Maior visualização lógica do desenvolvimento completo de um sistema de informação

UML

- A UML 2 possui 14 tipos de diagramas



Adaptado de *UML Superstructure Specification* 2.4.1, *Figure* A.5

(https://commons.wikimedia.org/wiki/File%3AUML_diagrams_

UML

- Podemos ainda dividir a UML em visões
  - Visão estática
    - Define os conceitos chaves da aplicação: **as classes e seus relacionamentos**
  - Visão dinâmica
    - Define a interação entre os objetos

DIAGRAMA DE CLASSE

DIAGRAMA DE CLASSE

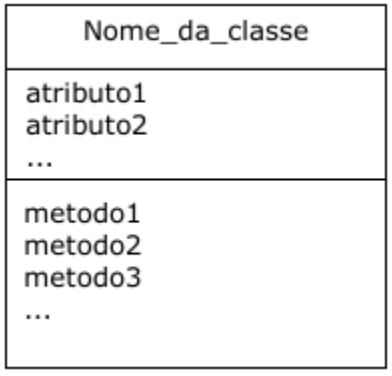
- É um diagrama estático que descreve a estrutura do sistema
  - Mostra quais são as classes do sistema, seus atributos e métodos
  - Mostra também o relacionamento entre as classes
- É o diagrama central da modelagem orientada a objetos

DIAGRAMA DE CLASSE

- Elementos de um diagrama de classes
  - Classes
  - Relacionamentos
    - Associação
      - Agregação
      - Composição
    - Generalização
    - Dependência

CLASSES

- Graficamente são representadas por retângulos



- Devem receber os nomes de acordo com o domínio do problema
- Em geral adotamos um padrão de nomenclatura
  - Substantivos singulares com a primeira letra maiúscula

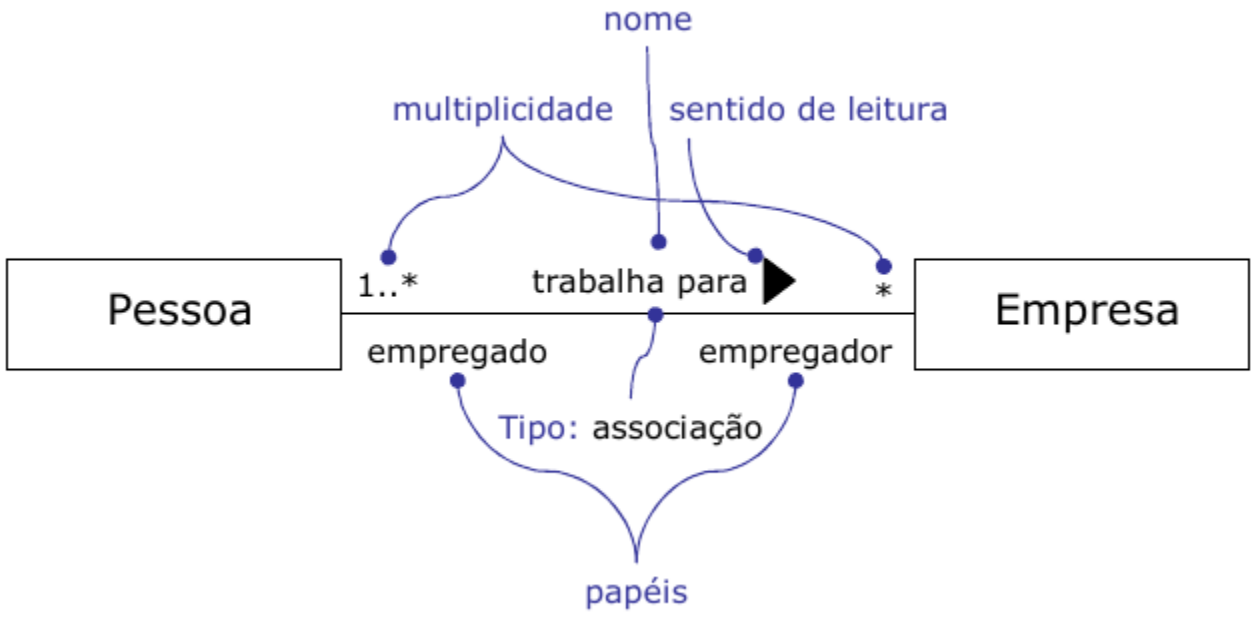
CLASSES

- Atributos
  - Representam o conjunto de características dos objetos daquela classe
- Métodos
  - Representam o conjunto de operações (comportamentos) que a classe fornece
- Devemos ainda definir a visibilidade dos atributos e métodos
  - Público: **+** visível em qualquer classe de qualquer pacote
  - Protegido: **#** visível para classes do mesmo pacote
  - Privado: **-** Visível somente para classe

RELACIONAMENTOS

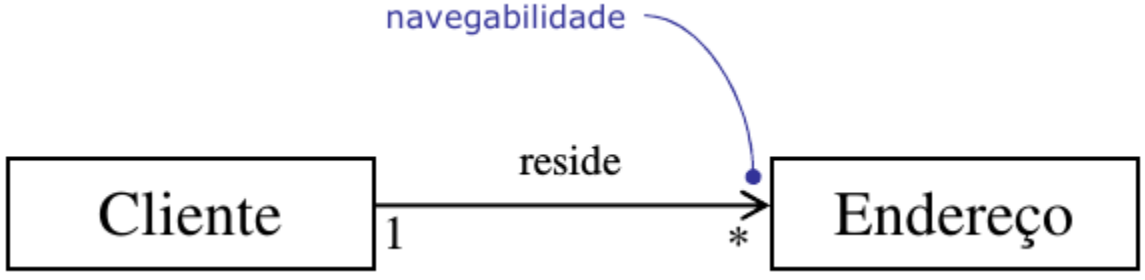
- Os relacionamentos possuem:
  - **Nome:** descrição dada ao relacionamento
  - **Sentido de leitura**
  - **Navegabilidade:** indicada por uma seta no fim do relacionamento
  - **Multiplicidade:**0..1, 0..*, 1, 1..*
  - **Tipo:** associação, generalização e dependência
  - **Papéis:** desempenhados por classes em um relacionamento

RELACIONAMENTOS



RELACIONAMENTOS

- Um relacionamento pode ser unidirecional ou bidirecional



- O cliente sabe quais são seus endereços, mas o endereço não sabe a quais clientes pertence

ASSOCIAÇÃO

- É um relacionamento que indica que os objetos de uma classe estão vinculadas a objetos de outra classe
- Um associação é representada por uma linha sólida conectando duas classes

ASSOCIAÇÃO

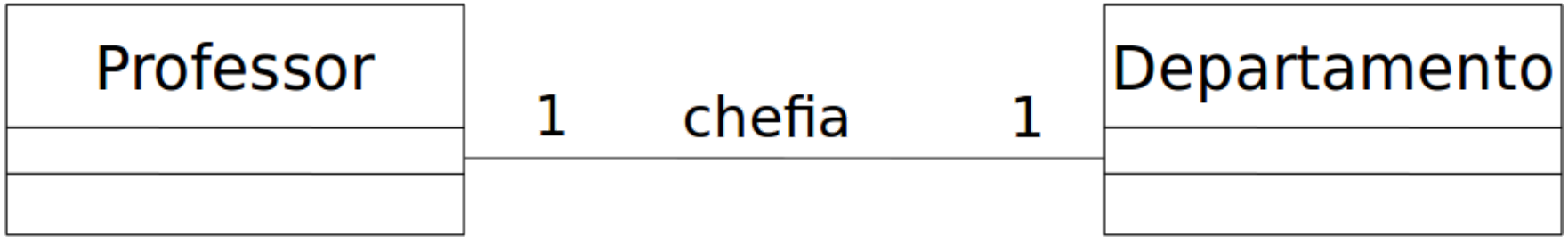
- Indicadores de multiplicidade
  - 1
  - 1..*
  - 0..*
  - *
  - 0..1
  - m..n

EXEMPLO

- Um projeto tem, alocados a ele, de 0 a N funcionários
- Um funcionário está alocado a um único projeto

ASSOCIAÇÃO

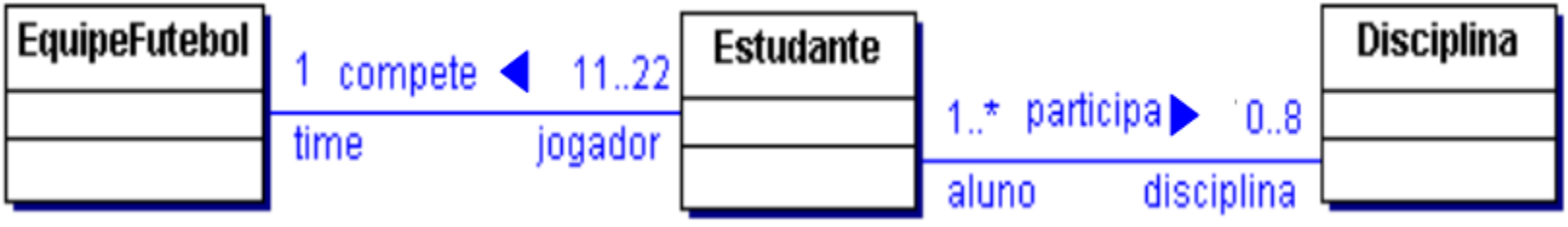
- Nome do relacionamento
  - Usa-se um verbo que permita obter frase da forma: **class1 + verbo + classe2**



  - Ex: professor chefia departamento

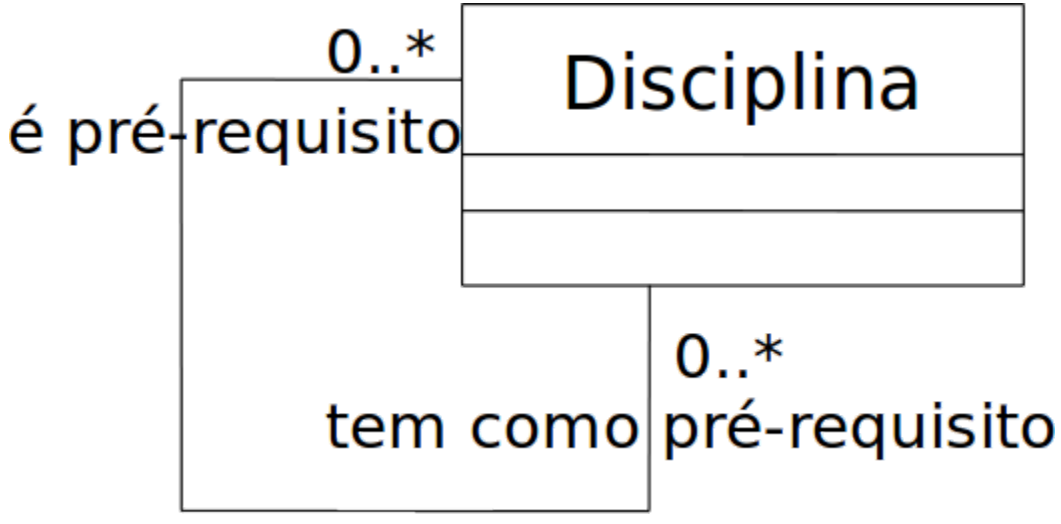
EXEMPLO

- Requisitos:
  - Um **Estudante** pode ser:
    - Um **aluno** de uma **disciplina**
    - Um **jogador** do **time de futebol**
  - Cada disciplina deve ser cursada por no mínimo um aluno
  - Um aluno pode cursar até 8 disciplinas



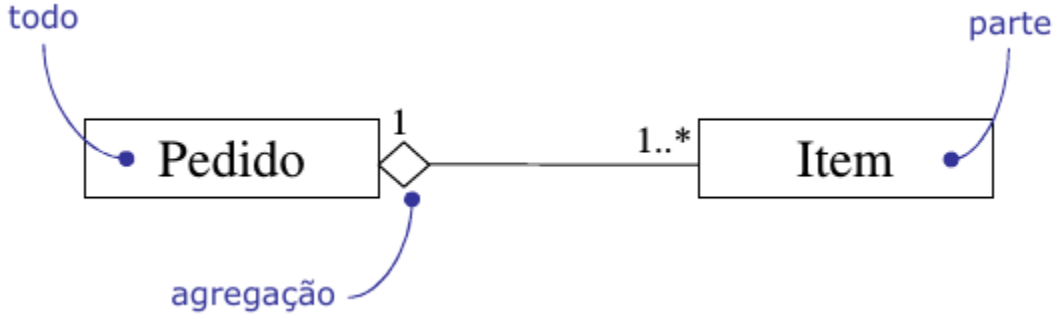
RELACIONAMENTOS REFLEXIVOS

- Envolvem objetos de uma mesma classe



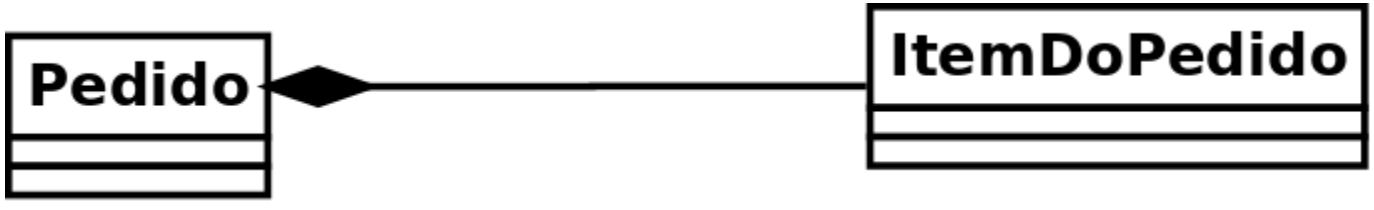
AGREGAÇÃO

- Tipo especial de **associação**
- Indica uma relação de *"todo-parte"*
- Em uma agregação um objeto está contido no outro
- Agregações são assimétricas



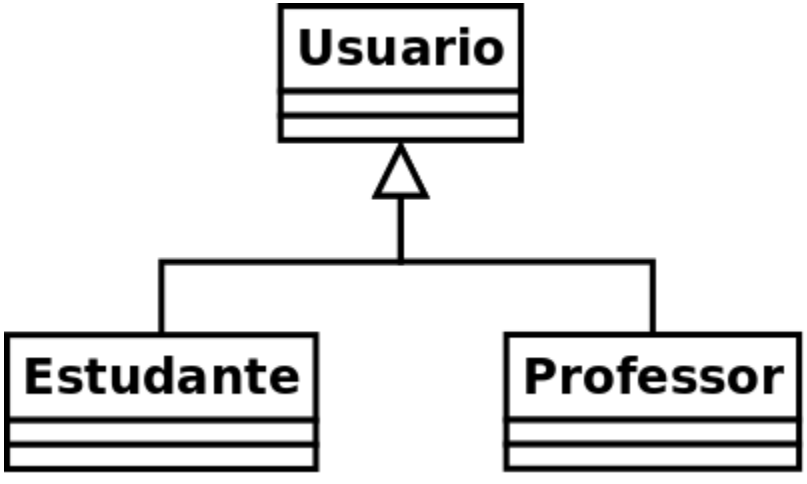
COMPOSIÇÃO

- Uma forma especial de agregação
- Usado quando as partes, para a sua existência, dependem da existência do todo
- O relacionamento é tão forte que as partes não pode existir independentes



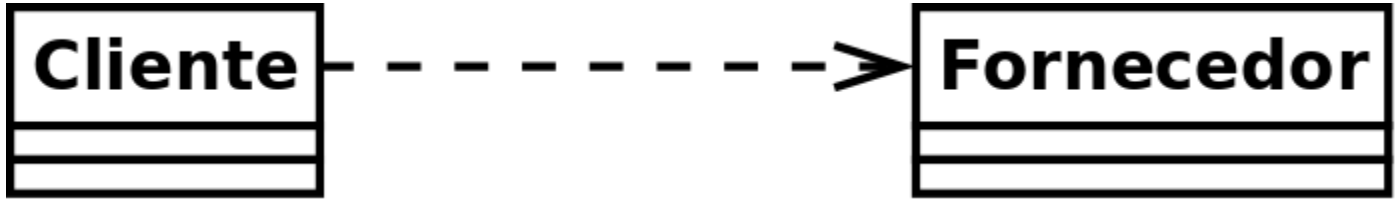
GENERALIZAÇÃO

- Generalização entre duas classes coloca-as numa hierarquia
- Representa o conceito de herança de uma classe derivada de uma classe base



DEPENDÊNCIA

- É um relacionamento direto que mostra que um ou vários elementos UML dependem de de outro elemento
- Também chamado de relacionamento **fornecedor/cliente**
- Uma modificação no **fornecedor** pode resultar em mudanças no **cliente**



  - Utiliza uma classe do fornecedor como um parâmetro para uma de suas operações
  - Utiliza uma classe do fornecedor como uma variável local para uma de suas operações
  - Envia uma mensagem para uma classe do fornecedor

O QUE VEM POR AÍ

Encapsulamento, modificadores de acesso e atributos de classe

Material inspirado no material da Laboratório de Engenharia de Software da PUC Rio (http://www.les.inf.puc-rio.br/wiki/images/7/7f/Aula1-diagrama_classes.pdf)