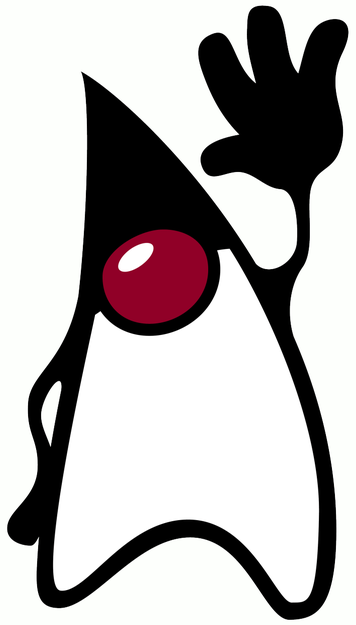
UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Introdução a UML

QXD0007 - Programação Orientada a Objetos

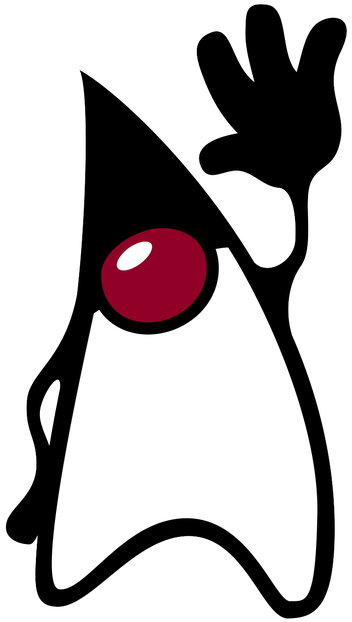


Prof. Bruno Góis Mateus (brunomateus@ufc.br)

Conteúdo

- Introdução
- Unified Modeling Language
- Diagrama de classes

Introdução



Introdução

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

Introdução

Análise orientada a objetos

- Foco em: “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 (usuários) precisam:
  - *revisar* seu **histórico**
  - *concorrer* a uma vaga de **emprego**
  - *navegar*, *comparar*, e *comprar* **produtos**

Introdução

Design orientado a objetos

- Foco em: “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

Introdução

Programação orientada a objetos

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

Introdução

Na prática

- Em um mundo ideal essas três etapas 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

Unified Modeling Language

Introdução

- A **Unified Modeling Language (UML)**, é um linguagem de modelagem utilizada na engenharia de software
- Criada para prover uma maneira padronizada 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

Unified Modeling Language

- É uma linguagem padronizada para modelar sistemas de software.
- Por que usar?
  - Facilita a comunicação entre desenvolvedores.
  - Auxilia no planejamento e documentação.
  - Permite visualizar a estrutura e o comportamento do sistema antes de codificar.

Unified Modeling Language

- É uma linguagem padronizada para modelar sistemas de software.
- Por que usar?
  - Facilita a comunicação entre desenvolvedores
  - Auxilia no planejamento e documentação
  - Permite visualizar a estrutura e o comportamento do sistema antes de codificar

Unified Modeling Language

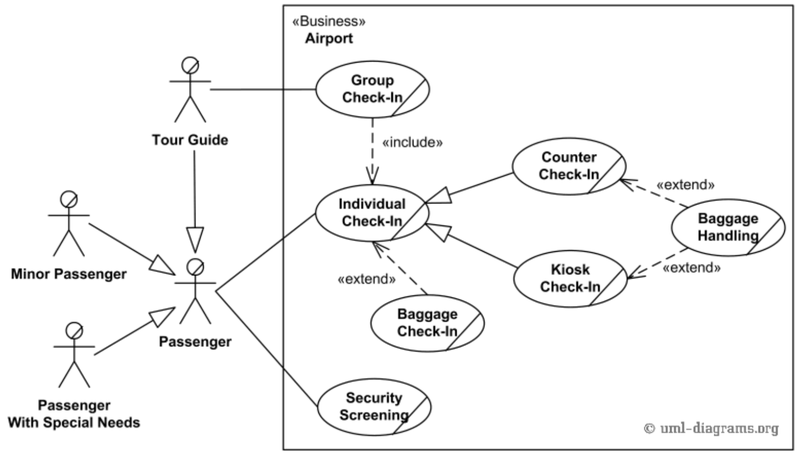


Diagram de caso de uso de check-in e verificação de segurança em um aeroporto [2]

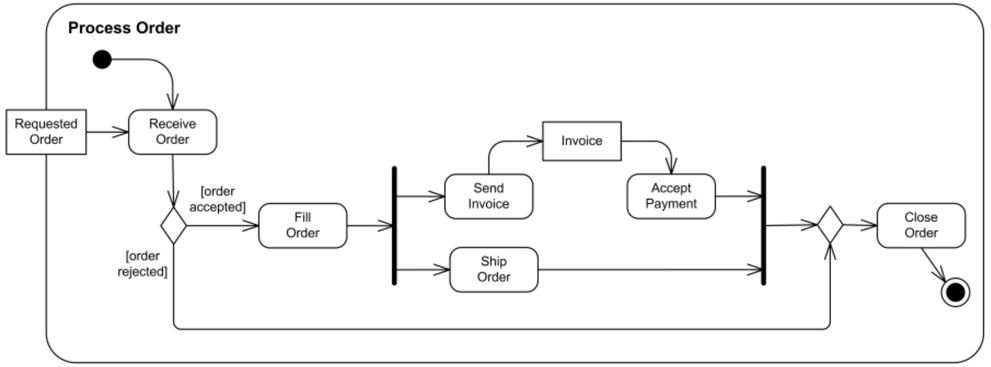


Diagram de atividades de um fluxo do processo de compra [3]

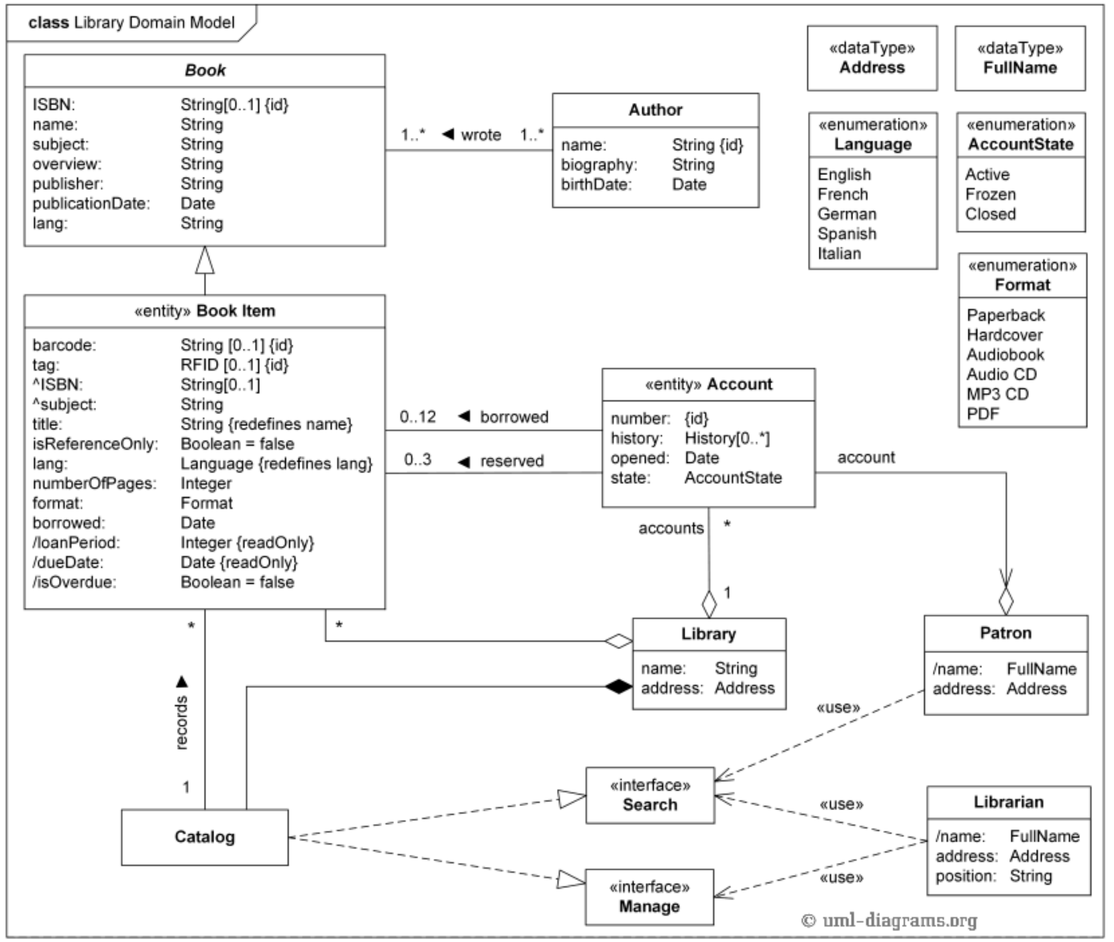
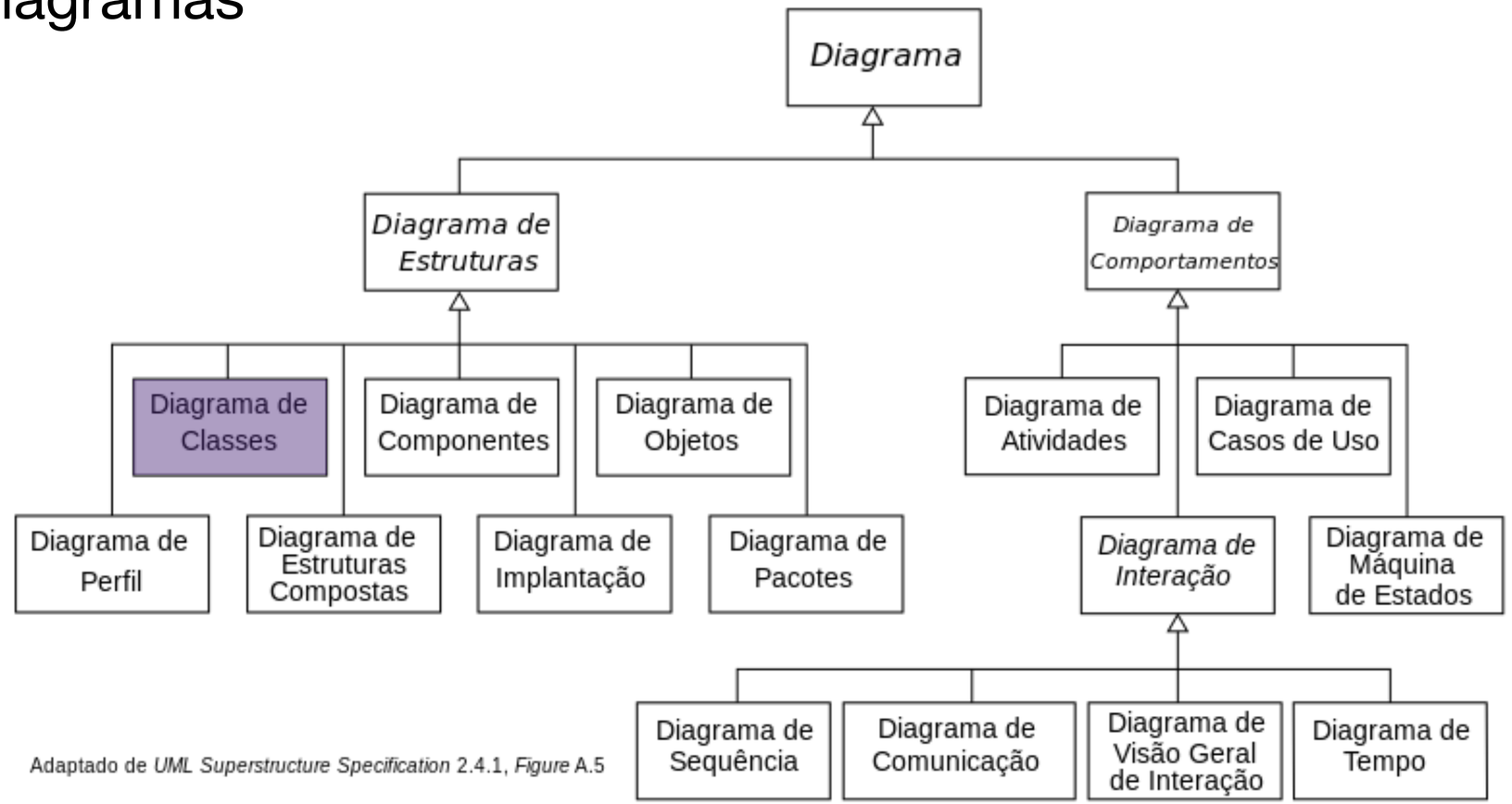


Diagrama de classes para um sistema de biblioteca [1]

Unified Modeling Language

- A UML possui 14 diagramas



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

Diagrama de classe

Diagrama de classe

- Representação visual da estrutura estática de um sistema
- Mostra as classes, seus atributos, seus métodos, e os relacionamentos entre elas.
- Útil para:
  - Planejar como as partes do sistema interagem
  - Entender o código antes de escrevê-lo
- É o diagrama central da modelagem orientada a objetos

Diagrama de classe

Principais elementos

Classes

Relacionamentos

Associação

Agregação

Composição

Generalização

Dependência

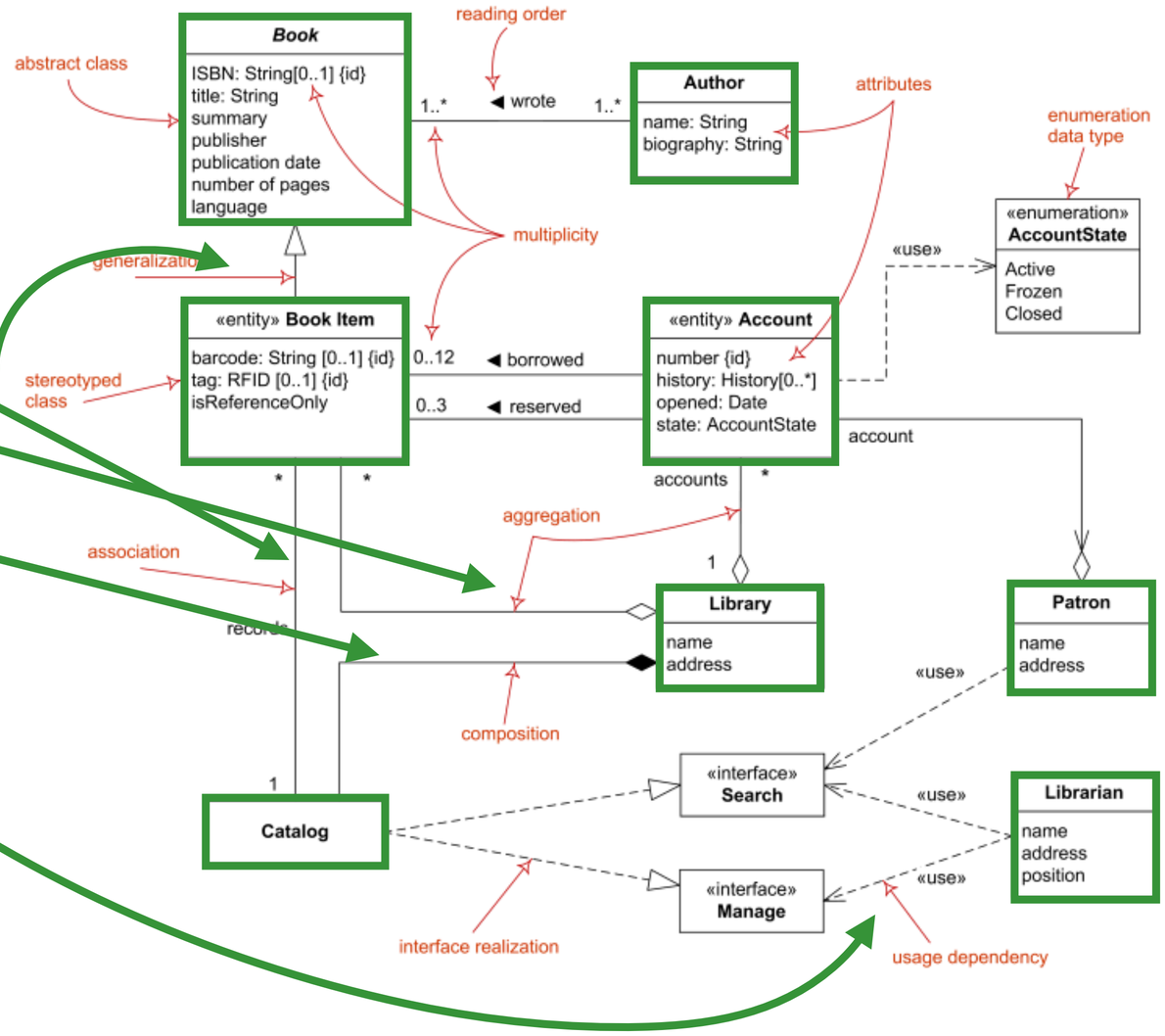
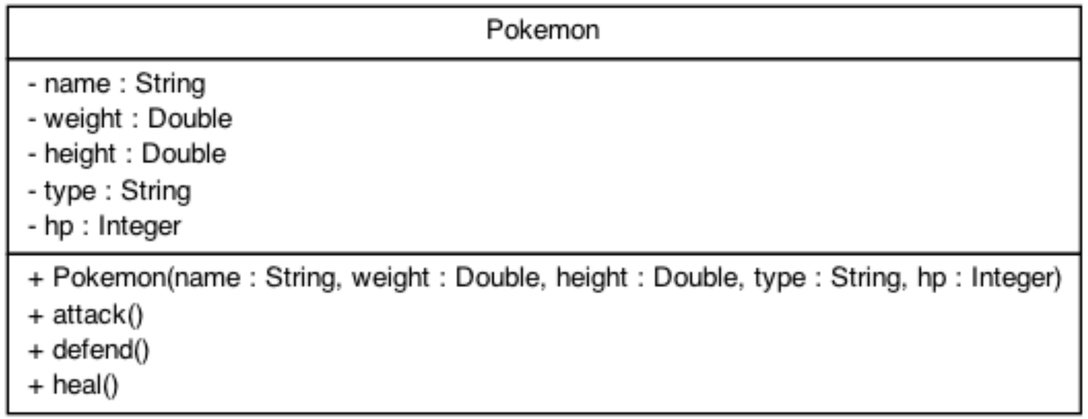


Diagrama de classe

Classes

- Graficamente são representadas por retângulos
- Devem ser nomeadas de acordo com o domínio do problema
- Em geral adotamos substantivos singulares com a primeira letra maiúscula



Nome da classe

Diagrama de classe

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: +
  - Protegido: #
  - Privado: -
  - Pacote (package ou default): ~

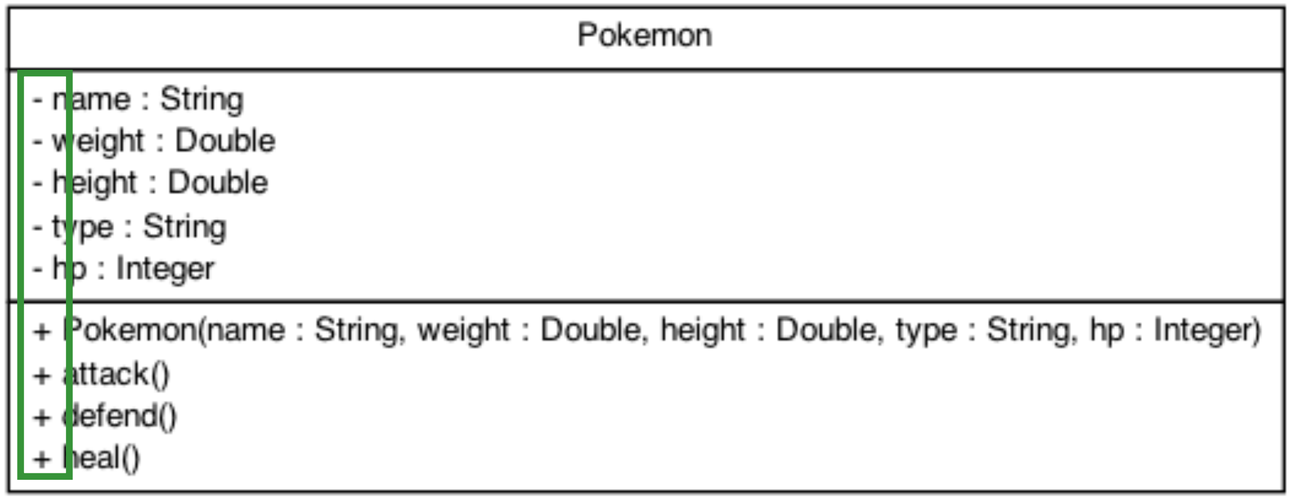


Diagrama de classe

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

Diagrama de classe

Relacionamentos

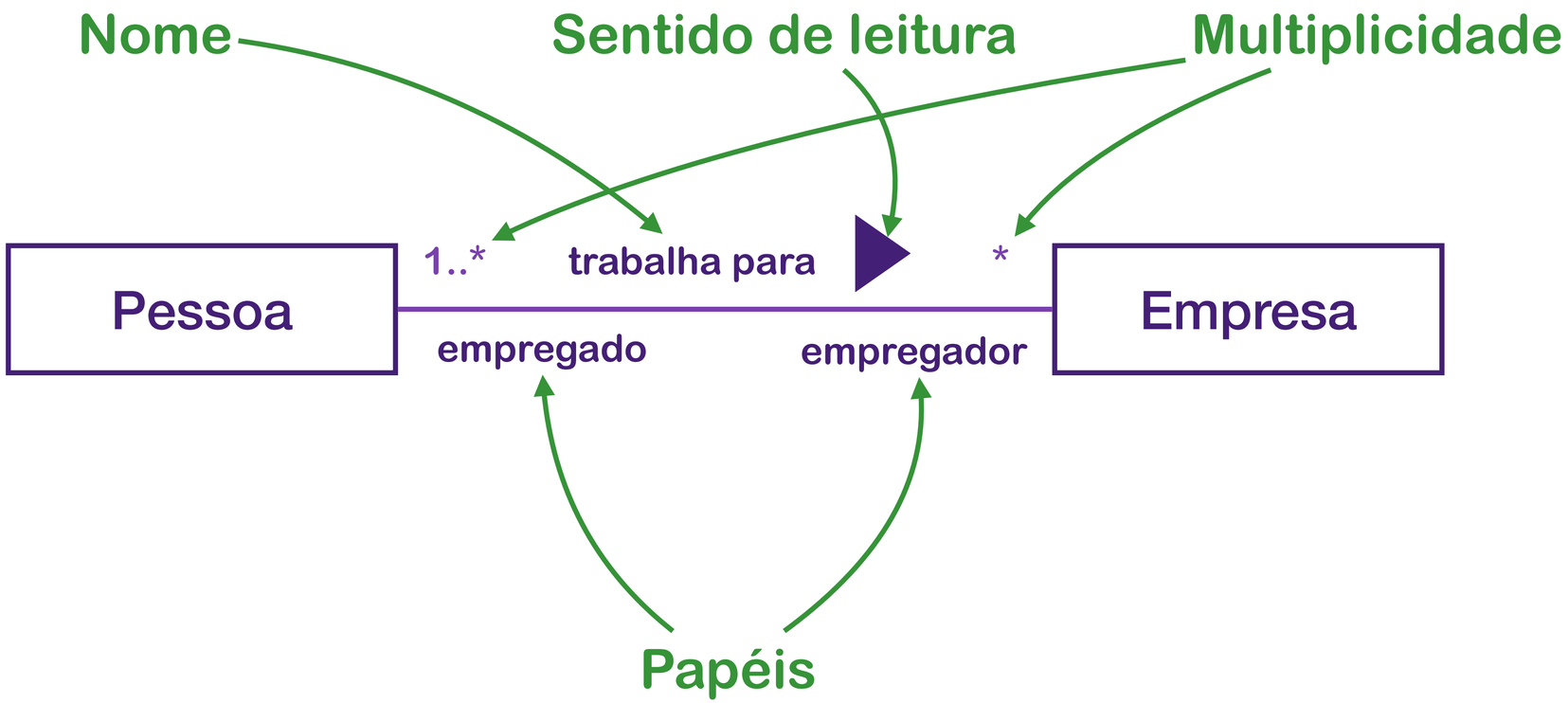
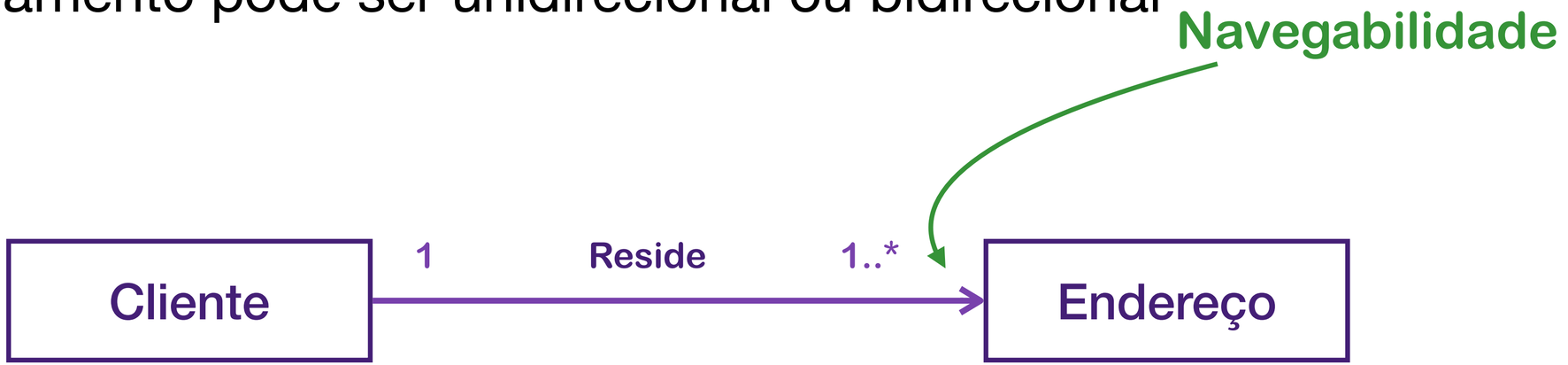


Diagrama de classe

Relacionamentos

- Um relacionamento pode ser unidirecional ou bidirecional



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

Diagrama de classe

Associação

- Indica que os objetos de uma classe estão vinculadas a objetos de outra classe
- É representada por uma linha sólida conectando duas classes
- Nomeando um relacionamento
  - Usa-se um verbo que permita obter frase da forma:
    classe1 + verbo + classe2

Diagrama de classe

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

Diagrama de classe

Exemplo

Diagrama de classe

Associação

- Envolvem objetos de uma mesma classe

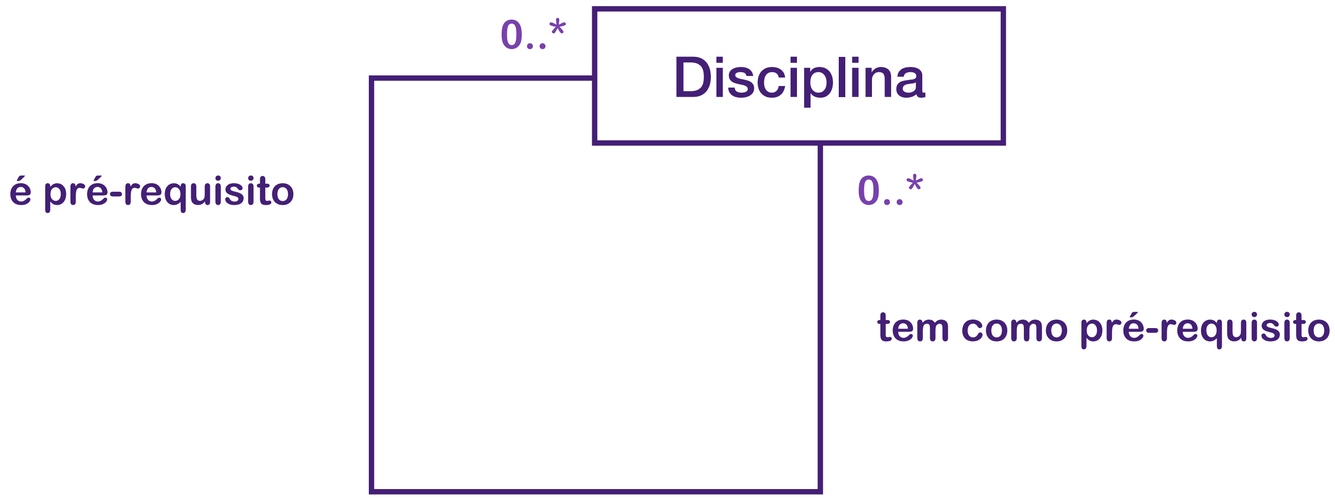


Diagrama de classe

Agregação

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

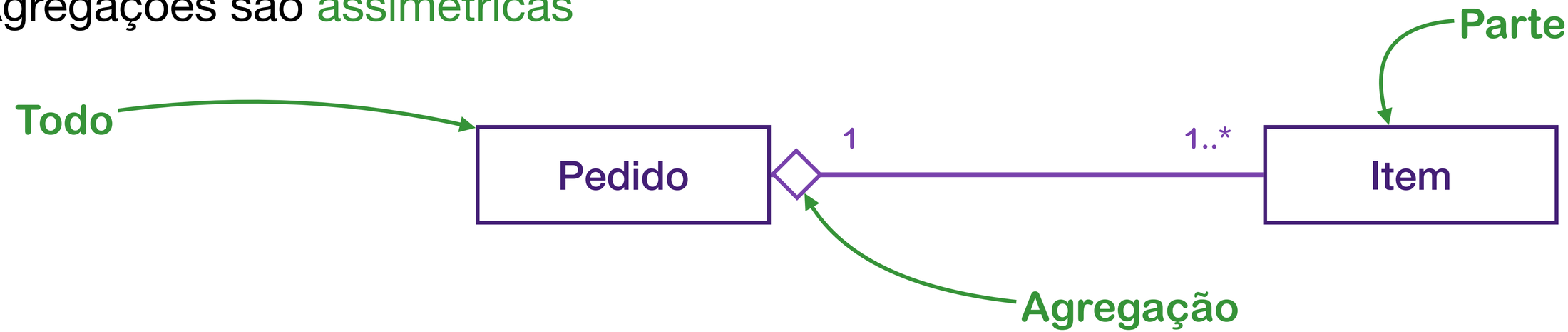


Diagrama de classe

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

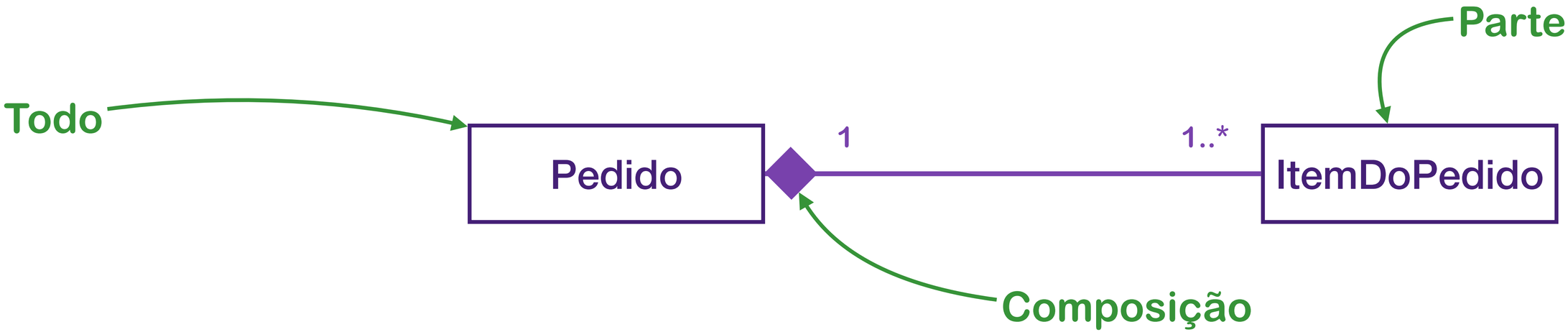


Diagrama de classe

Generalização

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

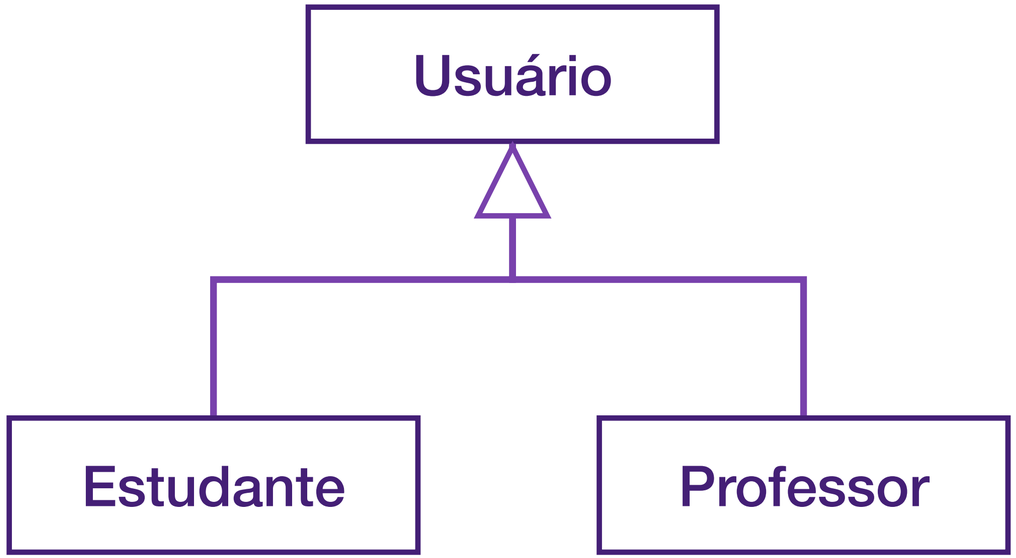
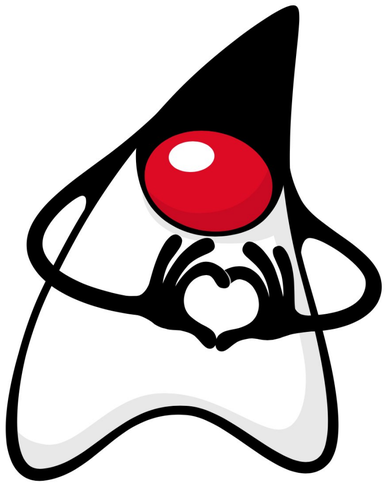


Diagrama de classe

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

Por hoje é só



Mão na massa

NOT VISIBLE