

Design de Software

Diagrama de Classes

Baseado no material do Prof. Eduardo Figueiredo - UFU



Tópicos da Aula

- Conceitos de programação orientada a objetos
- Projeto orientado a objetos com UML
- Diagrama de Classes

Pensar Orientado a Objetos

- Onde quer que você olhe no mundo real, você vê objetos
 - Pessoas, animais, plantas, carros, etc.
- Humanos pensam em termos de objetos
 - Portanto, POO é alto nível, i.e., mais próximo dos humanos que dos computadores

Características de Objetos

- Classificação
 - Animados: possuem vida, se movem...
 - Inanimados: não se movem por conta própria
- Objetos possuem **atributos**
 - Tamanho, forma, cor, peso, etc.
- Objetos exibem **comportamentos**
 - Uma bola rola, um avião voa
 - Uma pessoa anda, fala, pensa, etc.

Classe de Objetos

- Uma classe é um “esqueleto” para criação de objetos
 - Como a planta é um “esqueleto” para criação de casas

Definições

- **Objeto:**
 - Entidade que descreve uma realidade
- **Classe:**
 - Abstração que define objetos
- **Instância:**
 - Concretização de um objeto a partir de uma classe

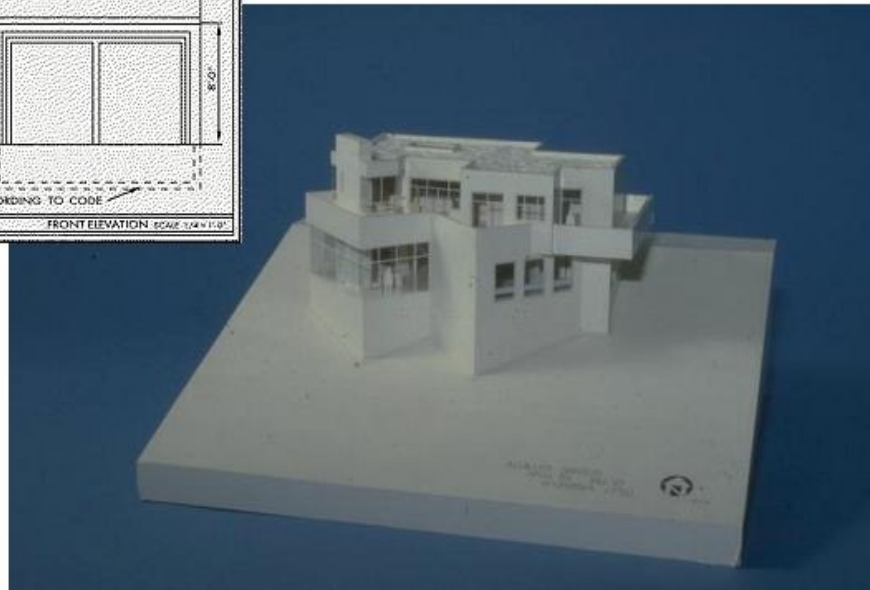
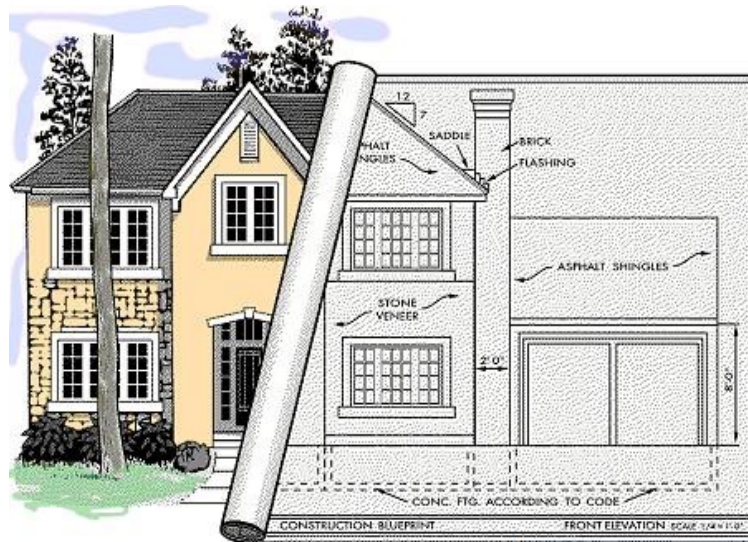
Projeto Orientado a Objetos

- Maneira natural de visualizar o software
 - Documentação e comunicação entre a equipe
- Modela o software semelhante ao mundo real - usando objetos
- **Objetos** são modelados em termos de seus dados (**atributos**) e seu comportamento (**métodos**)

Por que projetar?

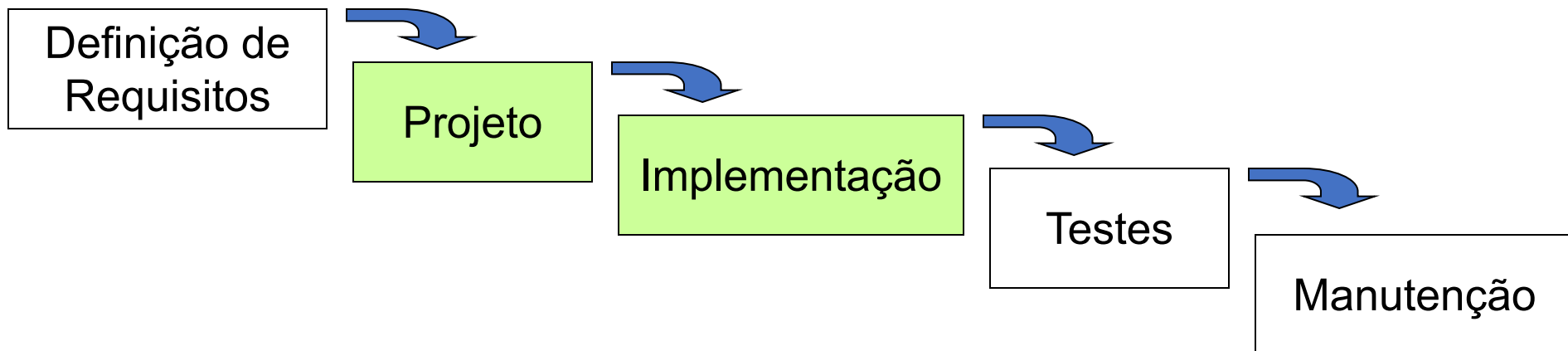
- Tão essencial quanto ter uma planta antes da construção de uma casa
 - Melhora a comunicação entre os membros da equipe
 - A equipe entende melhor o sistema
 - Permite analisar o sistema sobre vários aspectos
 - Facilita a programação e a manutenção
 - Diminui a possibilidade de erros

Projetar é Fundamental



Fases do Desenvolvimento

- Modelo Cascata
 - Define atividades seqüenciais
 - Outras abordagens são baseadas nesta ideia



A Linguagem UML

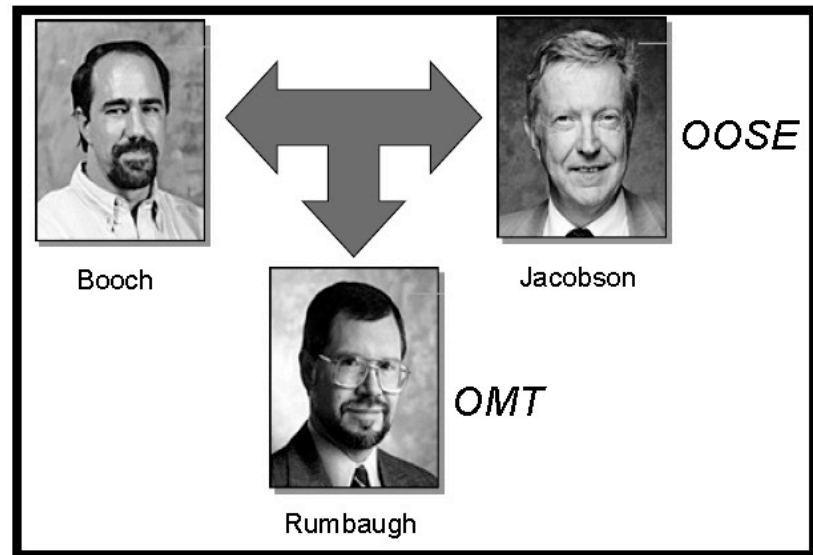
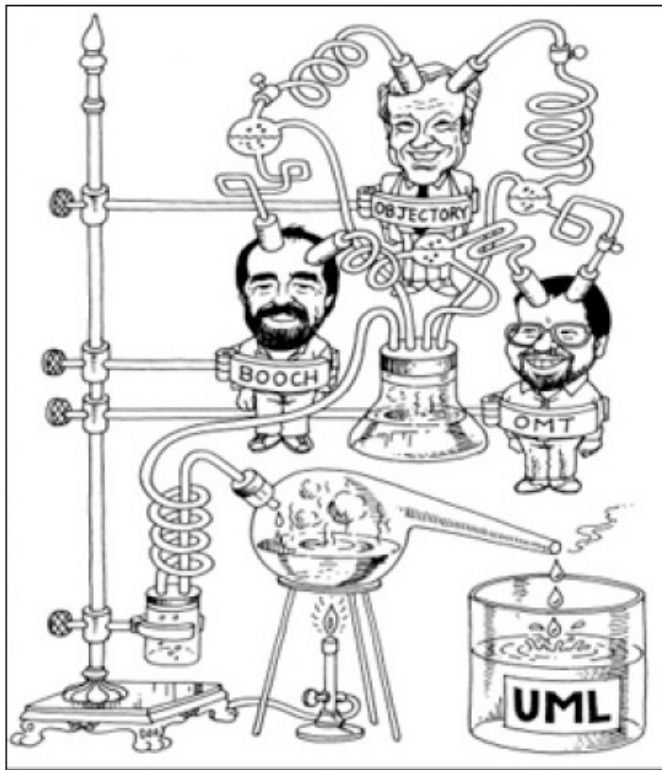
A Linguagem UML

- UML (Linguagem de Modelagem Unificada)
- É uma notação gráfica (visual) para projetar sistemas OO
 - Não é uma linguagem de programação
- Define diagramas padronizados
- É extensível
- É complexa
 - Usaremos apenas um subconjunto da UML

De onde surgiu?

- Da união de três metodologias de modelagem
 - Método de Booch - Grady Booch
 - Método OMT - Ivar Jacobson
 - Método OOSE - James Rumbaugh.

Fundadores da UML



História da UML

- 1994: Booch, Jacobson e Rumbaugh começaram a unificar suas notações
- 1996: Primeira versão (beta) da UML foi liberada
- 1996/97: Grandes empresas formaram a “UML Partners”
 - HP, IBM, Microsoft, Oracle, etc.
- 1997: UML foi adotada pela a OMG (Object Management Group) como linguagem padrão de modelagem

UML Define Diagramas

- Tipos Principais de Diagramas
 - Estrutural
 - Comportamental
- Objetivos
 - Visualizar o sistema
 - Especificar estrutura e/ou comportamento
 - Guiar e documentar das decisões

Diagramas UML

- Diagramas Estruturais (Estáticos)
 - Diagrama de Classes
 - Diagramas de Objetos
 - Diagrama de Caso de Uso
 - Diagrama de Componentes
- Diagramas Comportamentais (Dinâmicos)
 - Diagrama de Sequência
 - Diagrama de Estados
 - Diagrama de Atividades
 - Diagrama de Colaboração

Diagramas UML

- Diagramas Estruturais (Estáticos)

- Diagrama de Classes
- Diagramas de Objetos
- Diagrama de Caso de Uso
- Diagrama de Componentes

Por enquanto, vamos trabalhar apenas com diagramas de classes

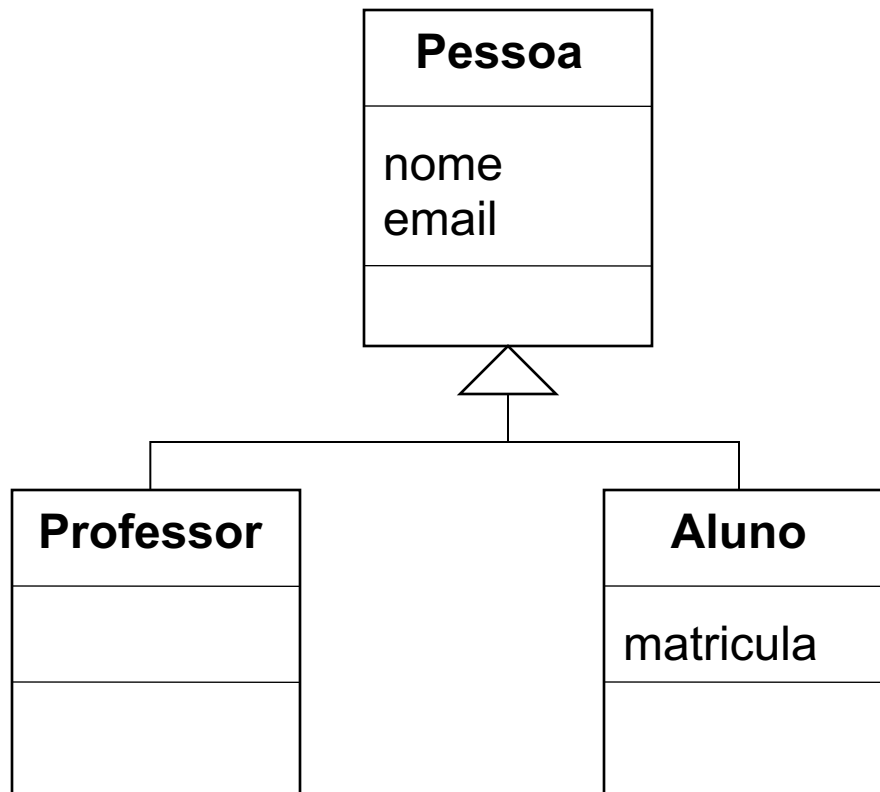
- Diagramas Comportamentais (Dinâmicos)

- Diagrama de Sequência
- Diagrama de Estados
- Diagrama de Atividades
- Diagrama de Colaboração

Diagrama de Classes

- Diagrama mais utilizado da UML
- Serve de apoio para a maioria dos outros diagramas
- Define a estrutura das classes do sistema
- Estabelece como as classes se relacionam

Diagrama de Classes – Sistema Acadêmico



Outro Diagrama de Classes

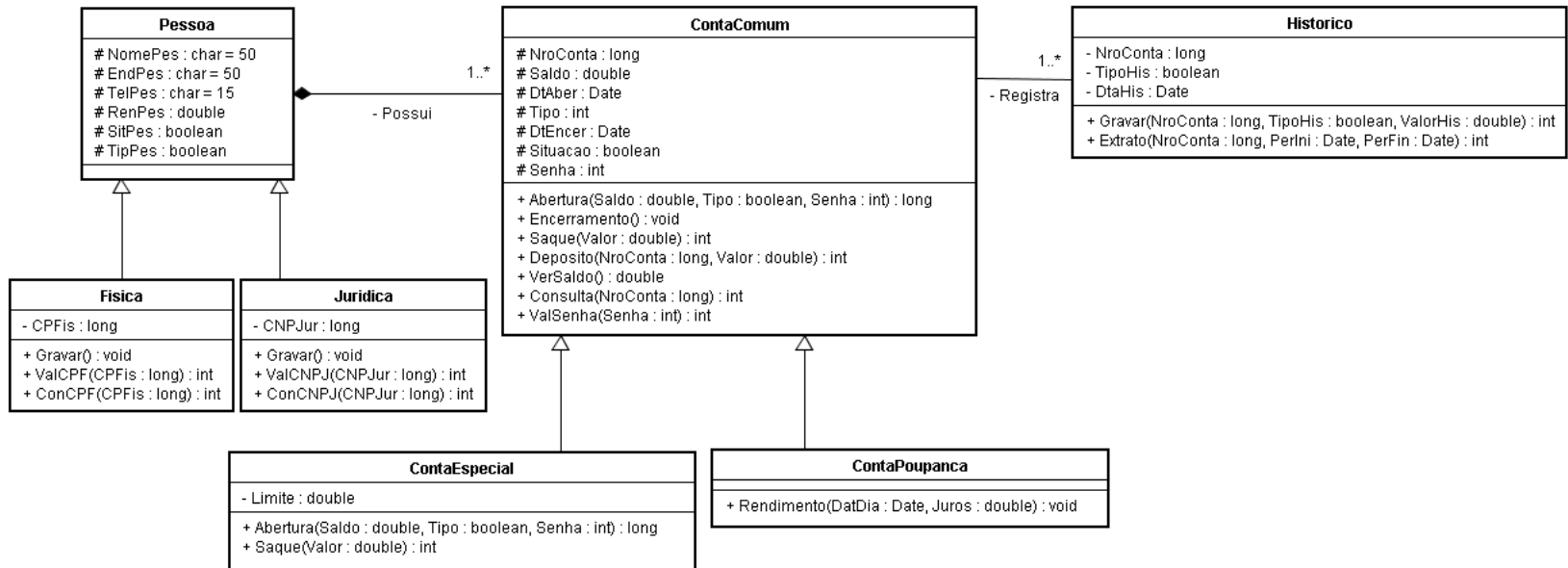


Diagrama de Classes

A estrutura do projeto

Diagrama de Classes

- O mais importante e o mais utilizado diagrama da UML
- Permite a visualização das classes que compõem o sistema
- Representa
 - Atributos e métodos de uma classe
 - Os relacionamento entre classes.

Diagrama de Classes

- Apresenta uma visão estática de como as classes estão organizadas
- Preocupação com a estrutura lógica

Atributos

- Permite a identificação de cada objeto de uma classe
- Os valores dos atributos podem variar de instância para instância
- Atributos devem conter o tipo de dados a ser armazenado
 - bool, int, float, string, etc.

Métodos

- São apenas declarados neste diagrama
 - Diagrama de Classes não define a implementação
- Outros diagramas permitem modelar o comportamento interno dos métodos
 - Diagrama de Seqüência
 - Diagrama de Atividades

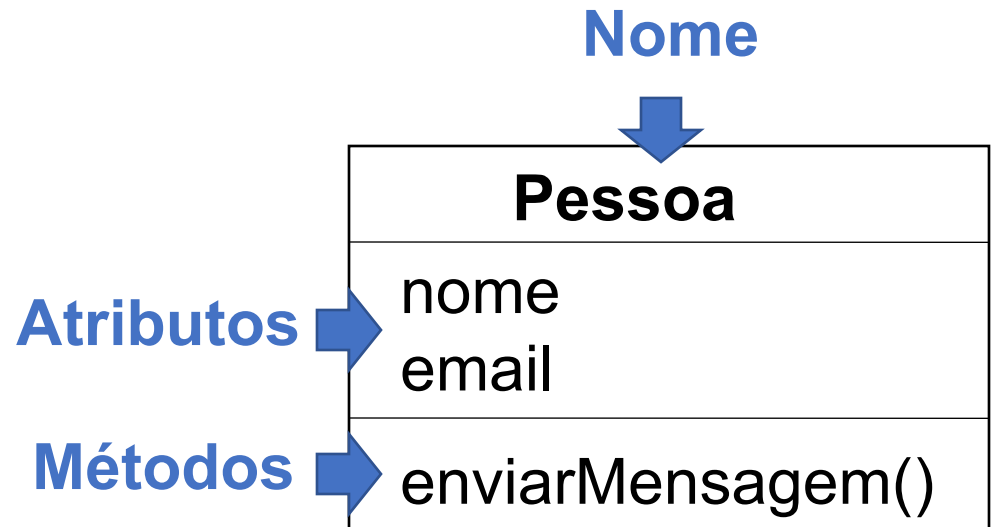
Representação de uma Classe

- Uma classe é representada por um retângulo com três divisões:
 - Nome da Classe
 - Atributos da Classe
 - Métodos da Classe

Pessoa
nome email
enviarMensagem()

Representação de uma Classe

- Uma classe é representada por um retângulo com três divisões:
 - Nome da Classe
 - Atributos da Classe
 - Métodos da Classe



Tipos de visibilidade

- Pública (+)
 - O atributo ou método pode ser utilizado por qualquer classe
- Protegida (#)
 - Somente a classe ou sub-classes terão acesso
- Privada (-)
 - Somente a classe terá acesso

Tipos de visibilidade

- Pública (+)
 - O atributo ou método pode ser utilizado por qualquer classe
- Protegida (#)
 - Somente a classe ou sub-classes terão acesso
- Privada (-)
 - Somente a classe terá acesso

Pessoa
nome - email
+ enviarMensagem()

Relacionamento

- Classes possuem relacionamentos entre elas
 - Compartilham informações
 - Colaboram umas com as outras
- Principais tipos de relacionamentos
 - Associação
 - Agregação / Composição
 - Herança
 - Dependência

Comunicação entre Objetos (I)

- Conceitualmente, objetos se comunicam através da troca de mensagens.
- Mensagens definem:
 - O nome do serviço requisitado
 - A informação necessária para a execução do serviço
 - O nome do requisitante.

Comunicação entre Objetos (II)

- Na prática, mensagens são implementadas como chamadas de métodos
 - Nome = o nome do método
 - Informação = a lista de parâmetros
 - Requisitante = o objeto que realizou a chamada

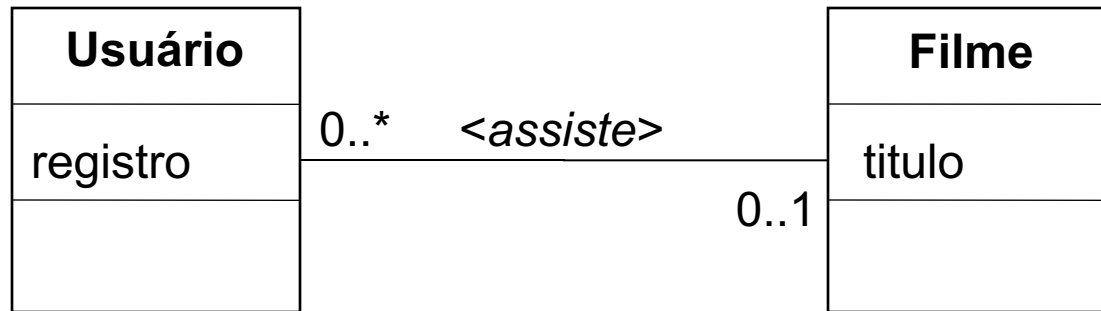
Associações

- Descreve um vínculo entre duas classes
 - Chamado **Associação Binária**
- Determina que as instâncias de uma classe estão de alguma forma ligadas às instâncias da outra classe

Multiplicidade

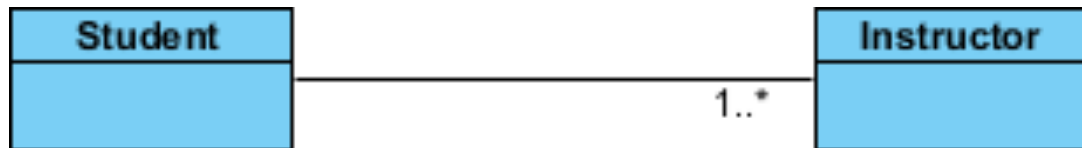
0..1	No máximo um. Indica que os Objetos da classe associada não precisam obrigatoriamente estar relacionados.
1..1	Um e somente um. Indica que apenas um objeto da classe se relaciona com os objetos da outra classe.
0..*	Muitos. Indica que podem haver muitos objetos da classe envolvidos no relacionamento
1..*	Um ou muitos. Indica que há pelo menos um objeto envolvido no relacionamento.
3..5	Valores específicos.

Representação de Associação

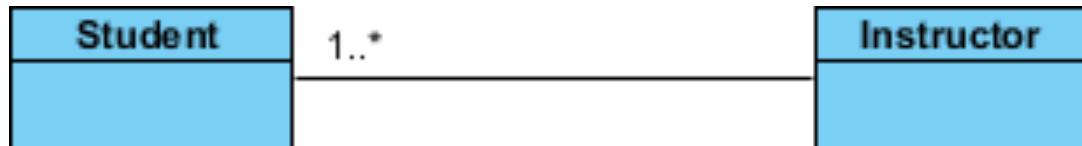


Representação de Associação

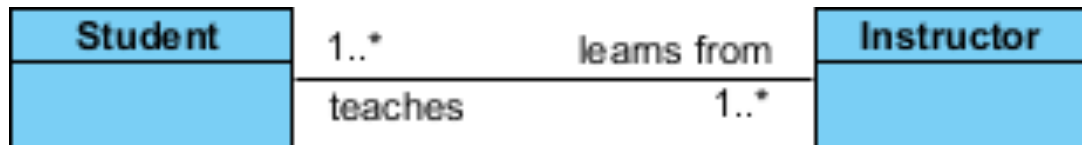
- Um único aluno pode se associar a vários professores:



- Cada Instrutor possui um ou mais Alunos:



- Podemos indicar a associação usando nomes de funções:

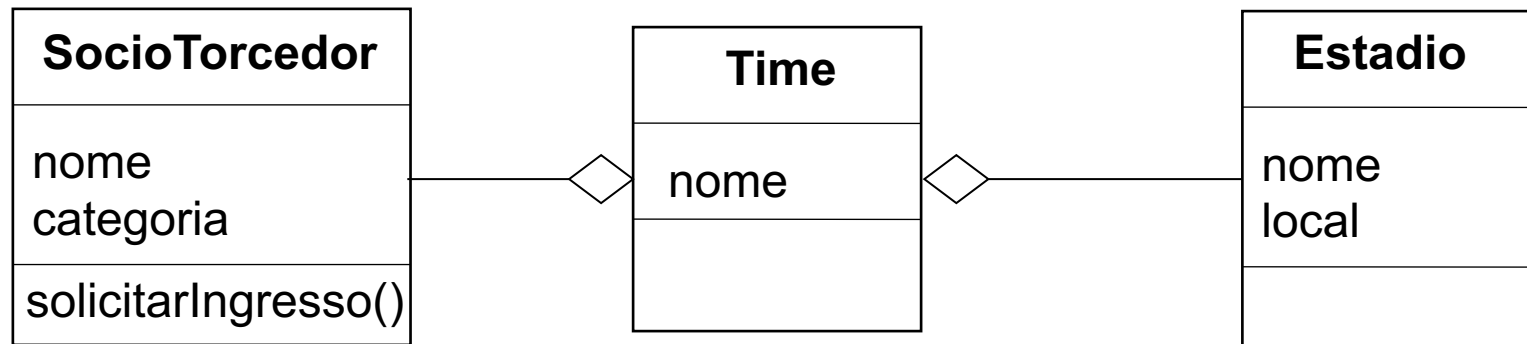


Agregação

- Tipo especial de associação
- Demonstra que as informações de um objeto precisam ser complementadas por um objeto de outra classe
- Associação Todo-Parte
 - objeto-todo
 - objeto-parte

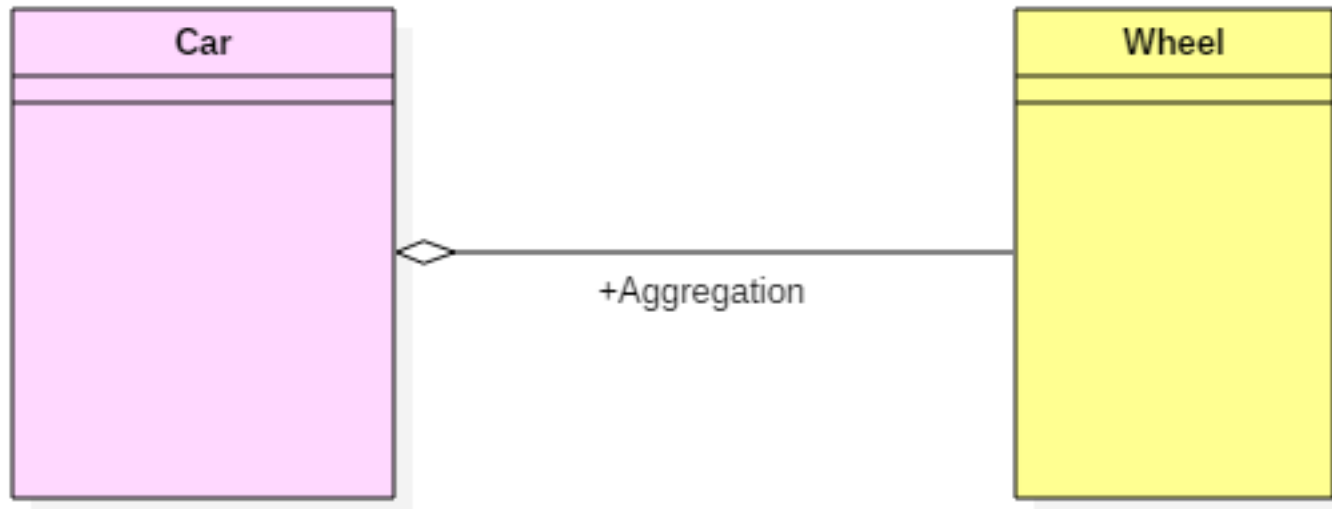
Representação de Agregação

- Um losango na extremidade da classe que contém os *objetos-todo*



Exemplo – Agregação

- Um carro precisa de uma roda para funcionar, mas nem sempre uma roda precisa de um carro
- Roda pode ser usada em bicicletas ou quaisquer outros veículos
- Aqui, o objeto roda é significativo mesmo sem o objeto carro

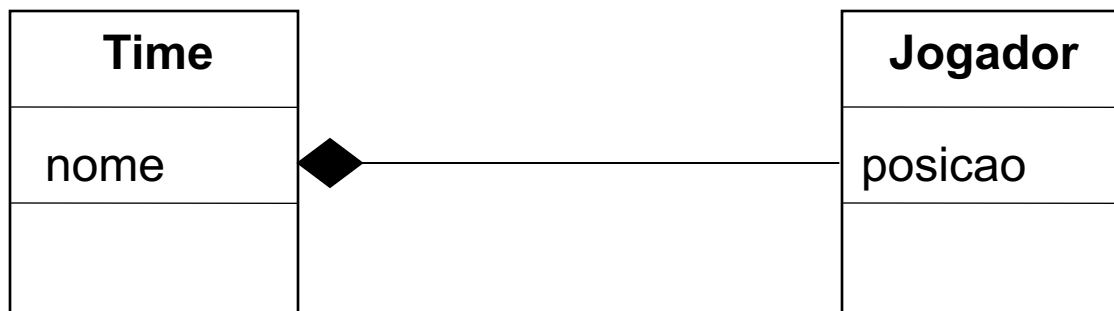


Composição

- Uma variação do tipo agregação
- Representa um vínculo mais forte entre objetos-todo e objetos-parte
- Objetos-parte **têm** que pertencer ao objeto-todo
 - O todo não existe (ou não faz sentido) sem a parte

Representação da Composição

- Um losango preenchido, e da mesma forma que na Agregação, deve ficar ao lado do objeto-todo



Exemplo – Composição

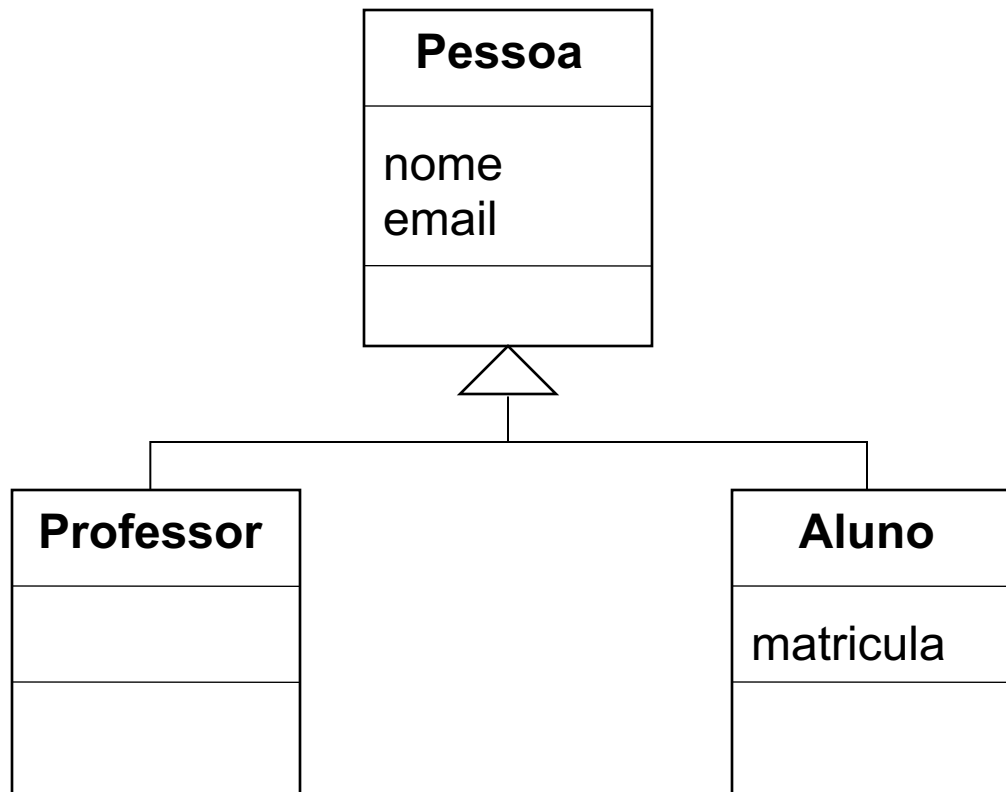
- Uma pasta pode conter muitos arquivos, enquanto cada arquivo possui exatamente uma pasta pai
- Se a pasta for excluída, todos os arquivos contidos também serão excluídos



Especialização / Generalização

- Identificar classes-mãe (gerais) e classes-filhas (especializadas)
- Atributos e métodos definidos na classe-mãe são **herdados** pelas classes-filhas

Especialização / Generalização



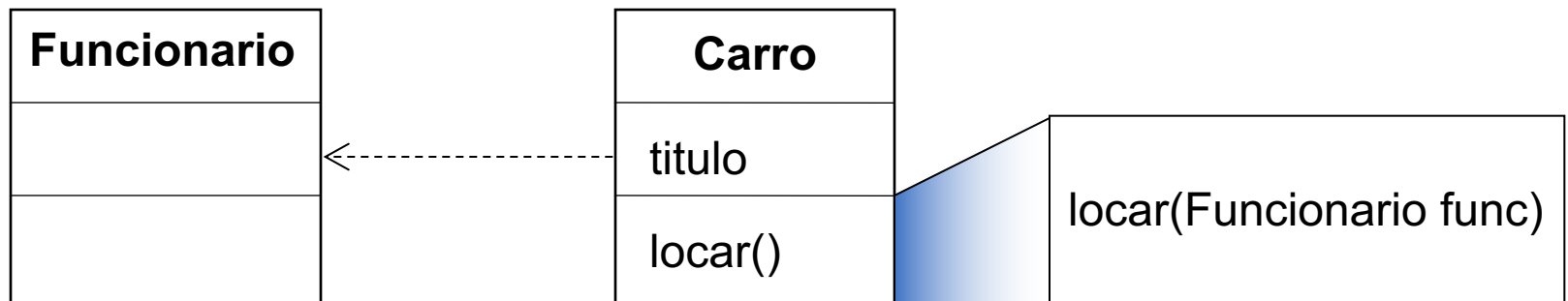
Dependência

- Representa um relacionamento genérico
- Indica que uma entidade usa ou depende de uma outra entidade



Dependência

- Representado por uma reta tracejada entre duas classes
- Uma seta na extremidade indica o dependente



+ For Public

- For Private

For Protected

/ For Derived

~ For Package

 Generalization

 Inheritance

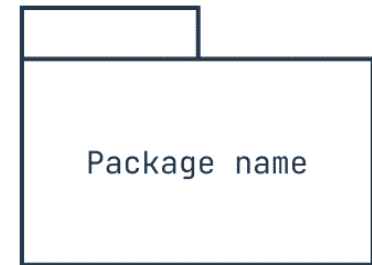
 Composition

 Aggregation

 Dependencies

 Properties

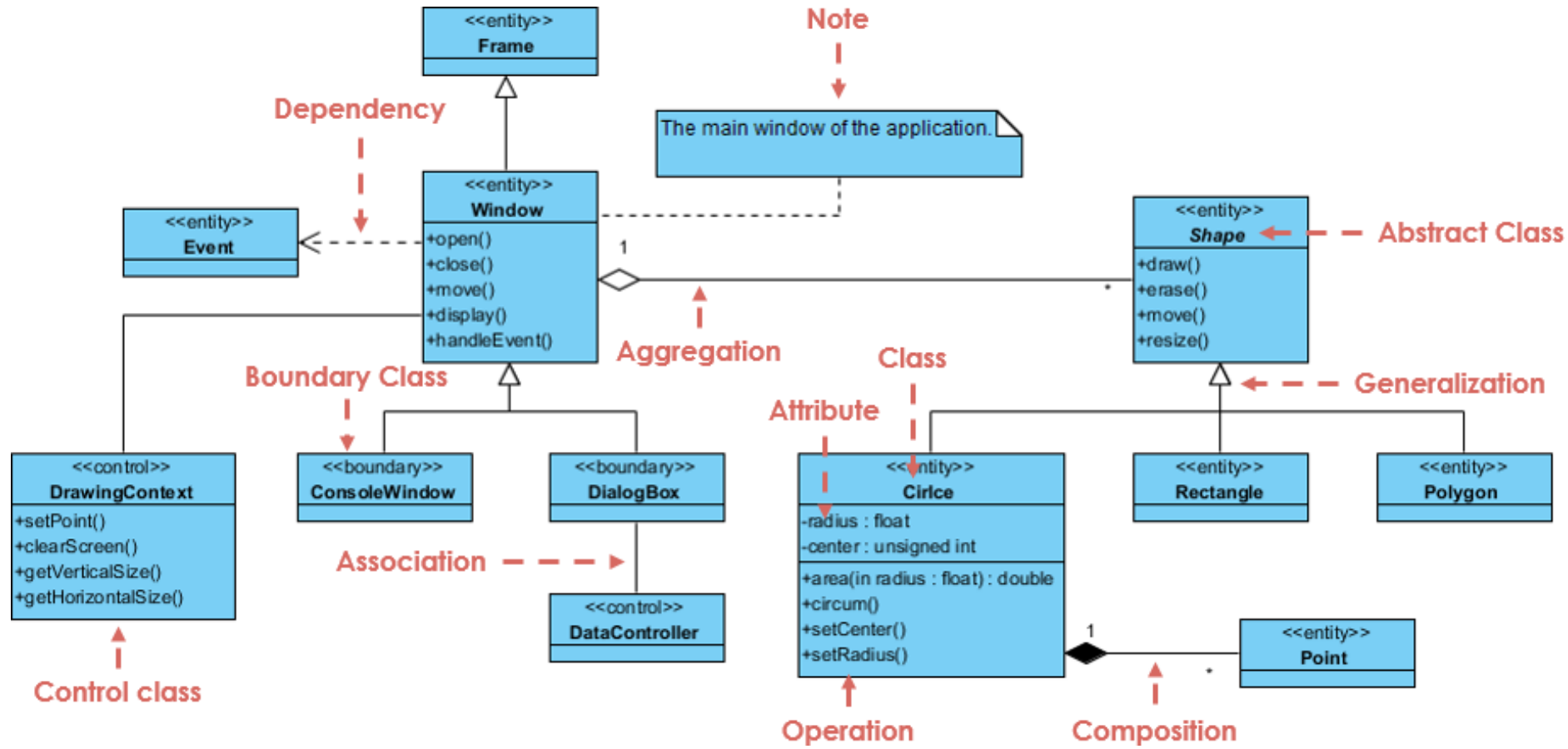
 Multiplicity



 Initial State

 Final State

Exemplo



Exercício: e-Commerce

Elabore um diagrama de classes para um sistema de e-commerce com as seguintes propriedades:

- **Cliente**

Esta classe deve ter atributos como id, nome, endereço e telefone. Deve ter operações como `fazer_pedido()` e `pagar()`

- Cliente pode realizar vários pedidos

- **Produto**

Atributos devem ser id, nome, categoria e preço. Deve ter operações como `adicionar_ao_carrinho()` e `comprar()`

- Produto pode estar presente em qualquer número de pedidos

- **Pedido:**

Deve ter id e preço total como atributos e `processar_pedido()` e `realizar_entrega()` como métodos

- Pedido deve pertencer a um cliente
- Pedido deve ser formado por um ou mais produtos

- **Pagamento:**

Atributos devem ser id, valor e método de pagamento e deve ter operações como `realizar_pagamento()` e `estornar()`

- Cada pagamento deve ser relativo a um único pedido

Exercício: Hospital (parte 1)

Elabore um diagrama de classes para um sistema hospitalar com as seguintes características:

- Uma Pessoa pode estar associada a diferentes Hospitais, e um Hospital pode empregar ou atender a várias Pessoas. Uma pessoa pode ter nome, sobrenome, data de nascimento, gênero e telefone.
- Uma pessoa pode ser um Paciente ou um Funcionário. Um paciente possui data de internação enquanto Funcionário possui data de admissão.
- Um Hospital tem nome, endereço e telefone. Um hospital possui vários departamentos. Cada departamento pode possuir vários funcionários.

Exercício: Hospital (parte 2)

Um funcionário pode se especializar em:

- Funcionário Operacional
- Funcionário Administrativo
- Funcionário Técnico

Um Funcionário Operacional pode ser (se especializar em) Médico ou Enfermeiro. Um médico deve ter como atributo sua especialidade. O Funcionário Operacional pode estar associado a um paciente quando o atende.

Um Funcionário Administrativo pode se especializar em Atendente. Um atendente, por sua vez, pode se especializar em Recepcionista.

Um Funcionário Técnico pode se especializar em Técnico e Tecnólogo. Um tecnólogo por sua vez pode ser um Tecnólogo Cirúrgico.