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**:
  - Criação de um objeto a partir de uma classe

Projeto Orientado a Objetos

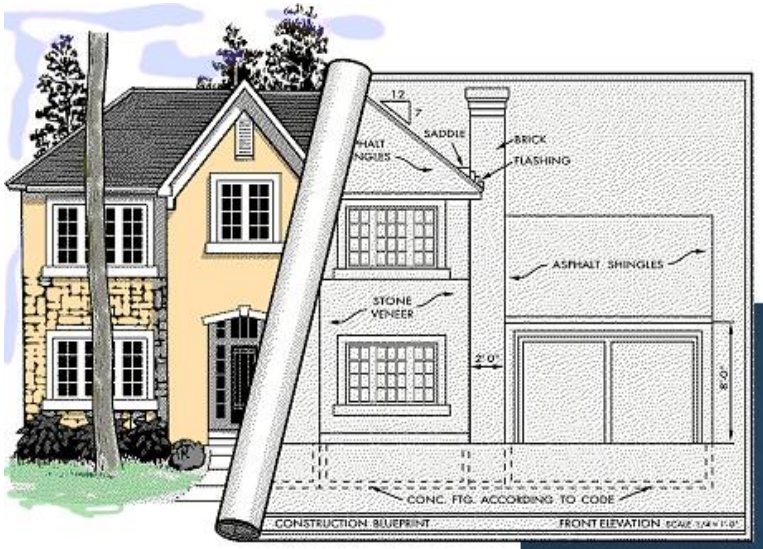
- Maneira natural de visualizar o software
  - Documentação e comunicação entre a equipe
- Modela o software semelhante ao mundo real - usando objetos
- **Objetos** são modelados em termos de seus **atributos** e comportamento (**métodos**)

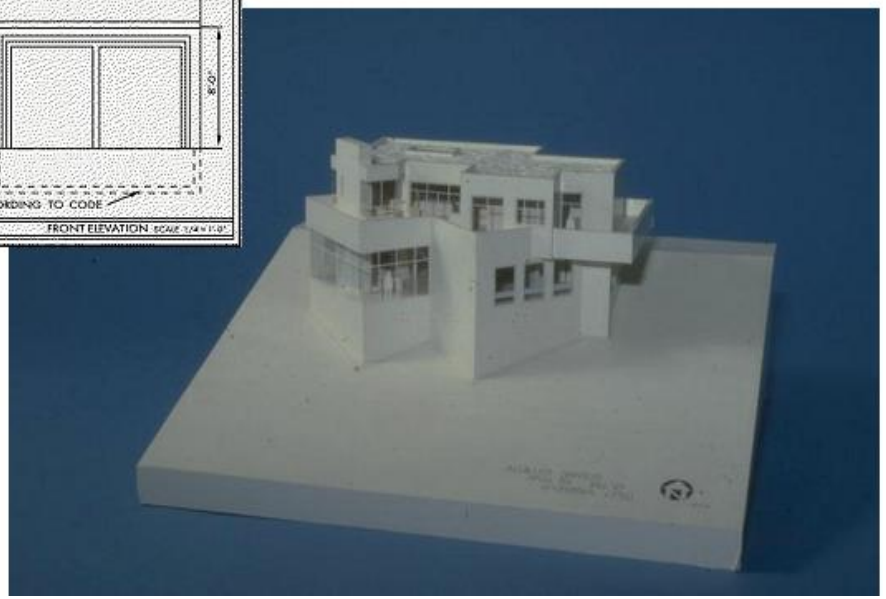
Por que projetar?

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





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
  - Mas usaremos apenas um sub-conjunto 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.

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, IMB, 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

Alguns Diagramas UML

- Diagramas Estruturais (Estáticos)
  - Diagrama de Classes
  - Diagramas de Objetos
  - Diagrama de Caso de Uso
  - Diagrama de Componentes, etc.
- Diagramas Dinâmicos
  - Diagrama de Seqüência
  - Diagrama de Estados
  - Diagrama de Atividades
  - Diagrama de Colaboração, etc.

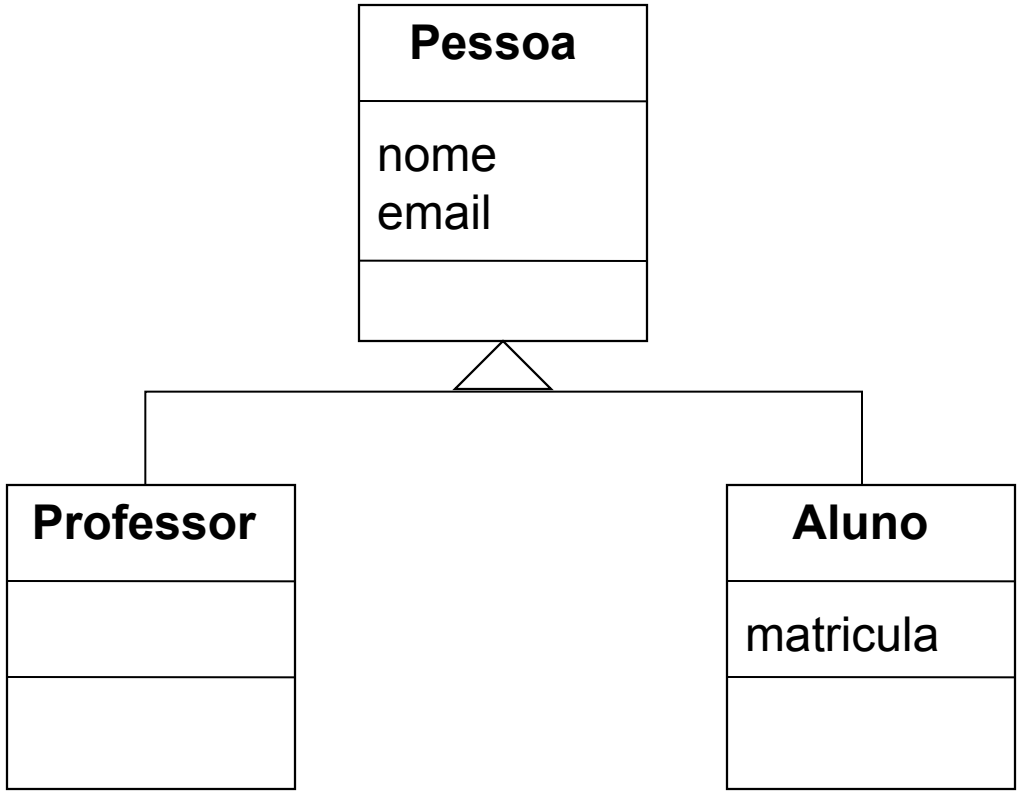
Alguns Diagramas UML

- Diagramas Estruturais (Estáticos)
  - **Diagrama de Classes**
  - Diagrama de Caso de Uso
  - Diagramas de Objetos
  - Diagrama de Componentes, etc.
- Diagramas Dinâmicos
  - Diagrama de Seqüência
  - Diagrama de Estados
  - Diagrama de Atividades
  - Diagrama de Colaboração, etc.

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

Meu Primeiro Diagrama



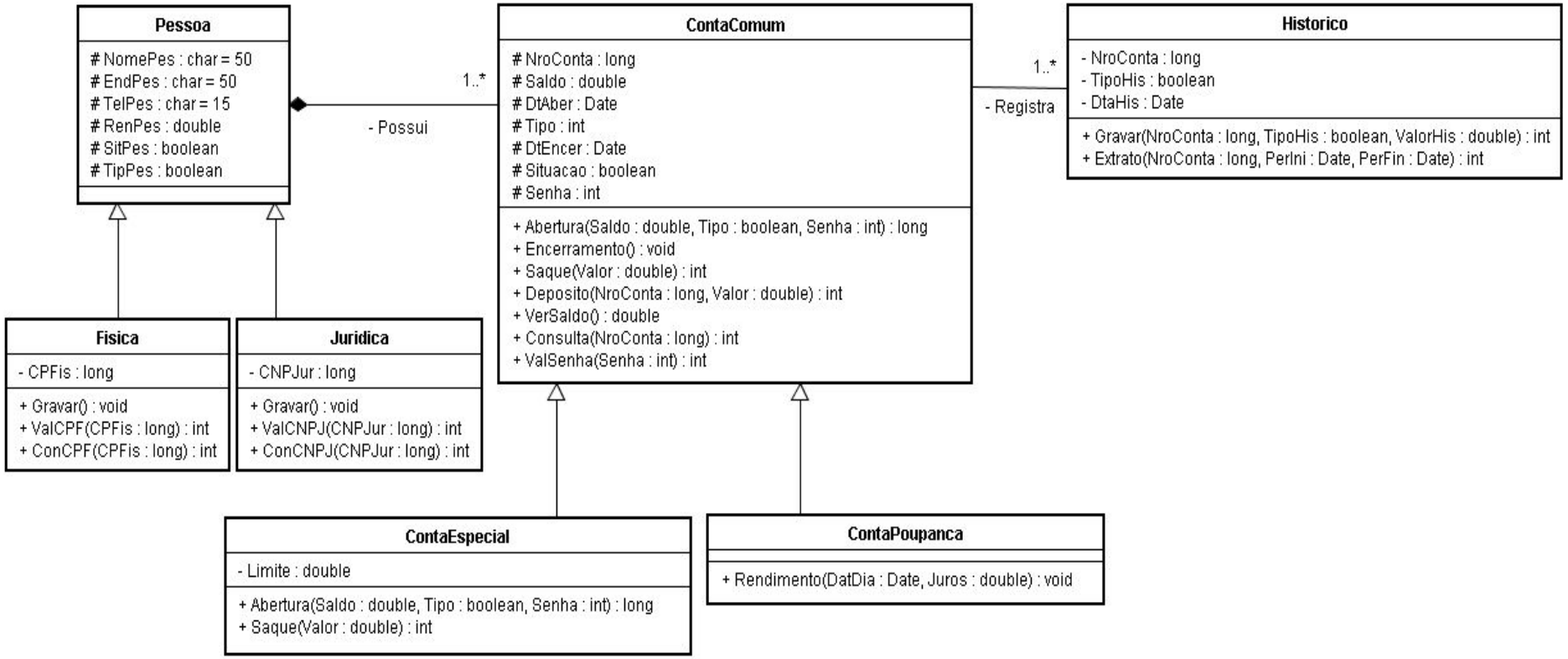

Pessoa
NomePes : char = 50
EndPes : char = 50
TelPes : char = 15
RenPes : double
SitPes : boolean
TipPes : boolean
1..*
- Possui
ContaComum
NroConta : long
Saldo : double
DtAber : Date
Tipo : int
DtEncer : Date
Situacao : boolean
Senha : int
+ Abertura(Saldo : double, Tipo : boolean, Senha : int) : long
+ Encerramento() : void
+ Saque(Valor : double) : int
+ Deposito(NroConta : long, Valor : double) : int
+ VerSaldo() : double
+ Consulta(NroConta : long) : int
+ ValSenha(Senha : int) : int
1..*
- Registra
Historico
- NroConta : long
- TipoHis : boolean
- DtaHis : Date
+ Gravar(NroConta : long, TipoHis : boolean, ValorHis : double) : int
+ Extrato(NroConta : long, PerIni : Date, PerFin : Date) : int
Fisica
- CPFis : long
+ Gravar() : void
+ ValCPF(CPFis : long) : int
+ ConCPF(CPFis : long) : int
Juridica
- CNPJur : long
+ Gravar() : void
+ ValCNPJ(CNPJur : long) : int
+ ConCNPJ(CNPJur : long) : int
ContaEspecial
- Limite : double
+ Abertura(Saldo : double, Tipo : boolean, Senha : int) : long
+ Saque(Valor : double) : int
ContaPoupanca
+ Rendimento(DatDia : Date, Juros : double) : void

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

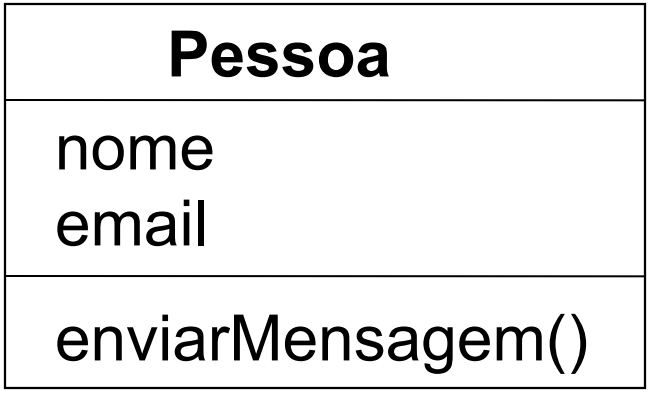
- Permite a identificação de cada objeto de uma classe
- Os valores dos atributos podem variar de instância para instância
- Atributos devem conter o tipo de dados a ser armazenado
  - Byte, boolean, int, double, char, String, etc.

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

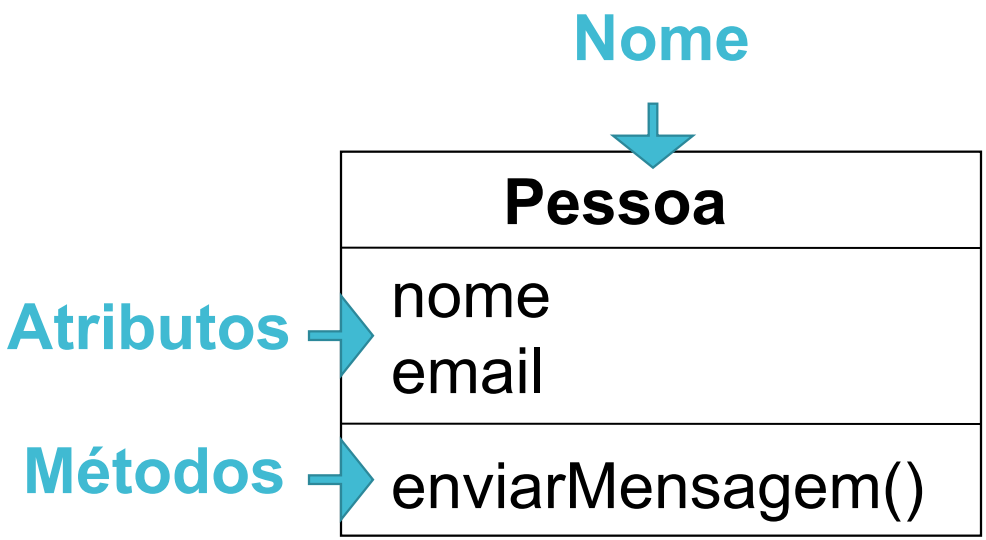
Representação de uma Classe

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



Representação de uma Classe

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

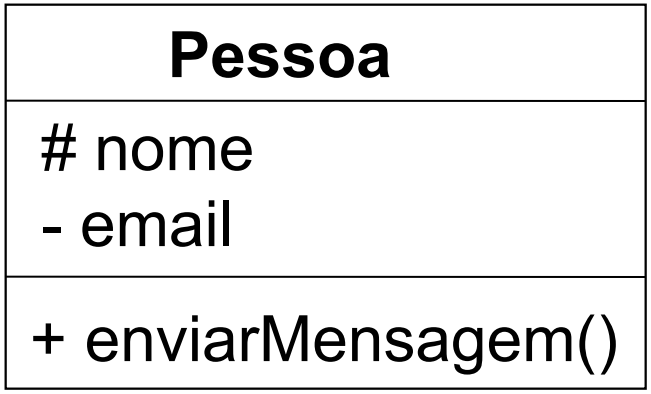


Tipos de visibilidade

- 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



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 método que realizou a chamada

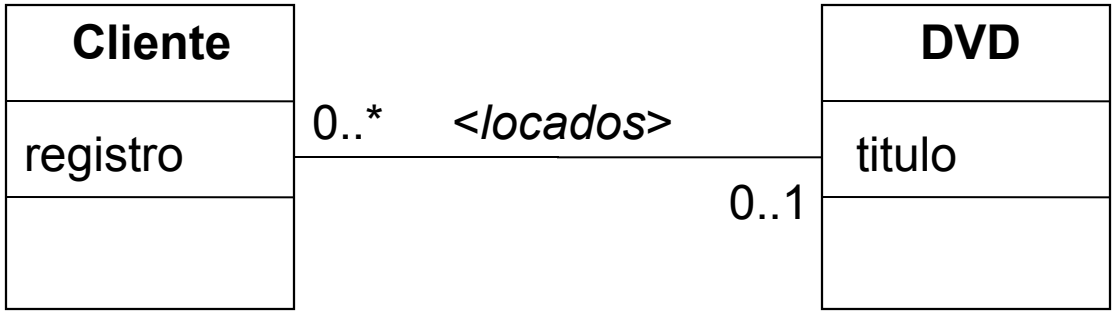
Associações

- 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



Agregação

- Tipo especial de associação
- Demonstra que as informações e 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*

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

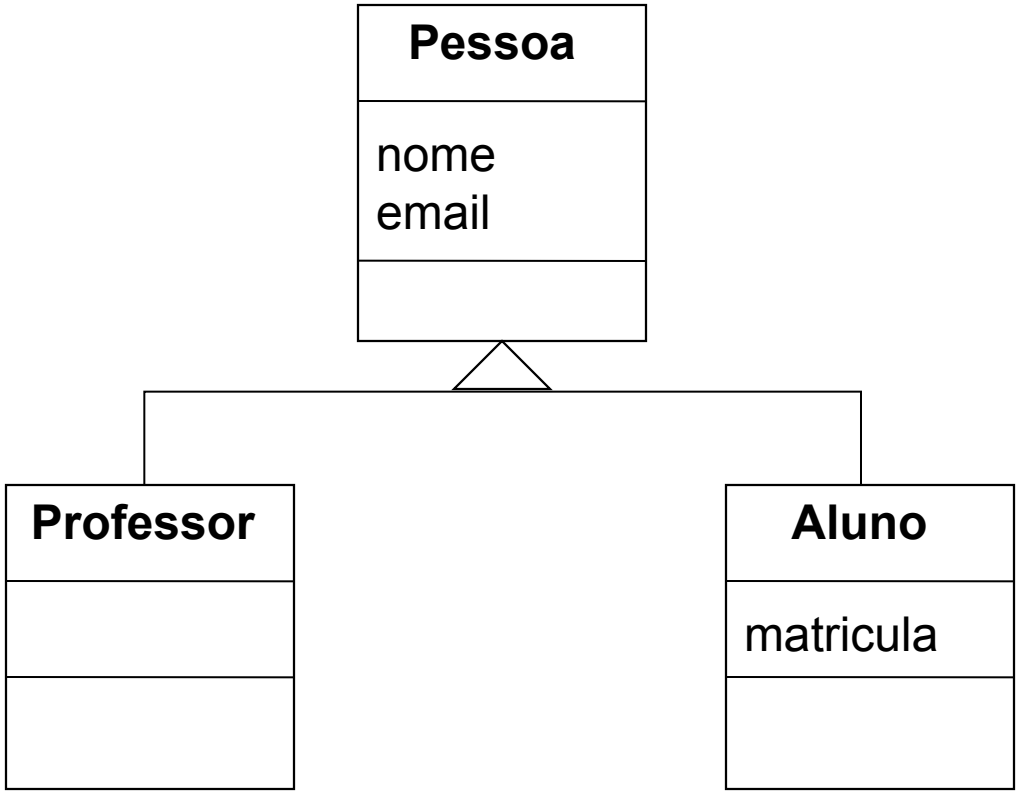
Representação da Composição

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

Especialização / Generalização

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

Especialização / Generalização

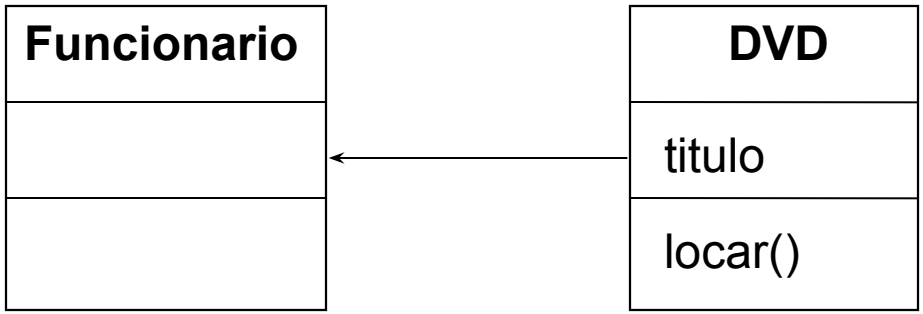


Dependência

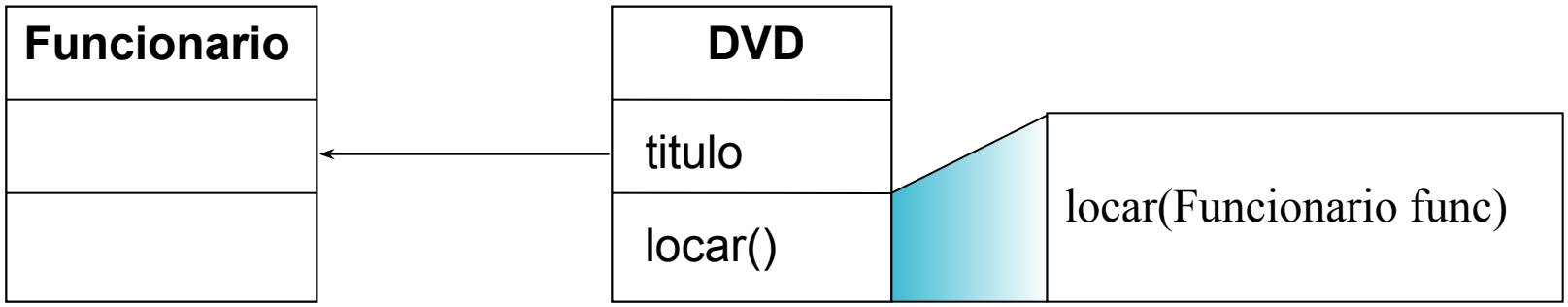
- Tipo menos comum de relacionamento
- Identifica um baixo grau de dependência de uma classe em relação a outra

Dependência

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



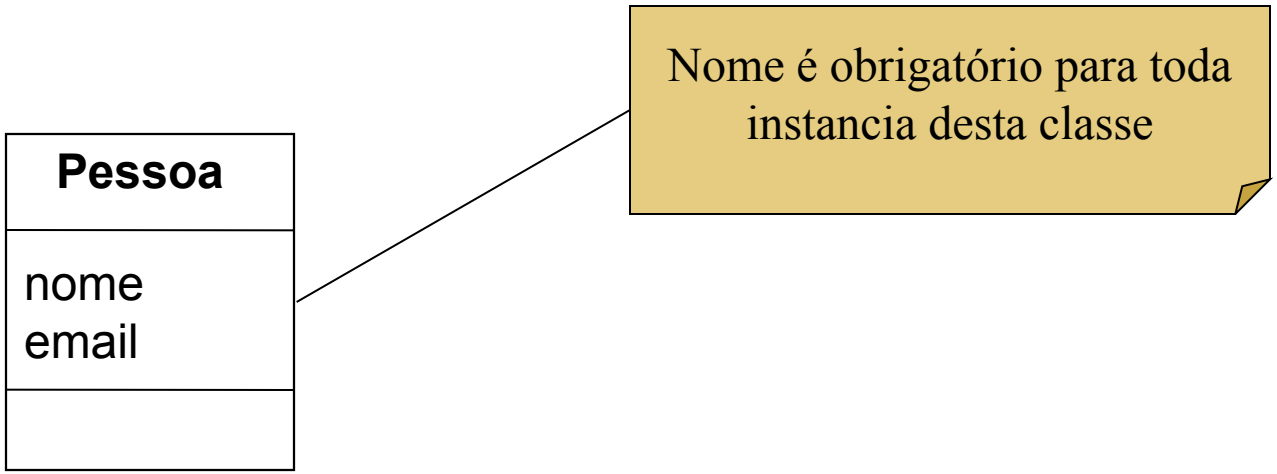
- Representado por uma reta tracejada entre duas classes
- Uma seta na extremidade indica o dependente



Notas

- Informativos
  - Algum comentário na classe, método ou atributo
  - Alguma restrição de funcionalidade
- Objetivo é informa como o objeto se comporta

Notas


Pessoa
nome
email
Nome é obrigatório para toda instancia desta classe

Referências

- DEITEL, H. M.; DEITEL P. J. **Java: Como Programar**, 6a. Edição. Pearson, 2005.
- BOOCH, G., RUMBAUGH, J., JACOBSON, I. **UML, Guia do Usuário**. Rio de Janeiro: Campus, 2000.