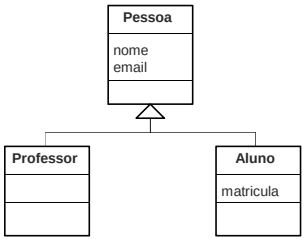
DCC / ICEx / UFMG

Diagrama de Classes

Eduardo Figueiredo

http://www.dcc.ufmg.br/~figueiredo

Primeiro Diagrama de Classes



Outro Diagrama de Classes

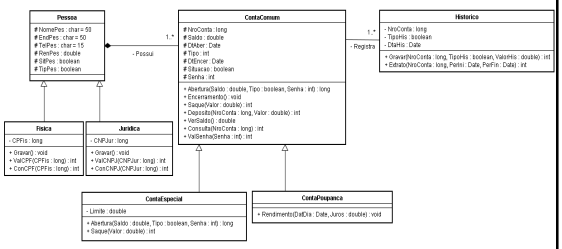


Diagrama de Classes

- Serve de apoio para a maioria dos outros diagramas
- Define a estrutura das classes do sistema
  - Apresenta uma visão estática de como as classes estão organizadas
- Estabelece como as classes se relacionam

Diagrama de Classes

- É 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 relacionamentos entre classes

Atributos

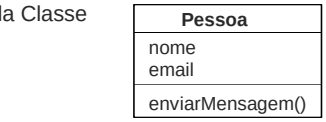
- Permite a identificação de cada objeto de uma classe
- Os valores dos atributos podem variar de instância para instância
- Atributos devem conter o tipo de dados a ser armazenado
  - Byte, boolean, int, double, char, String, etc.

Métodos

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

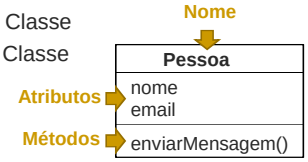
Representação de uma Classe

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



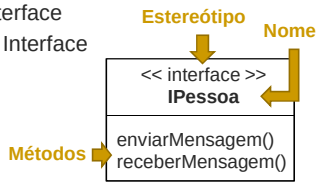
Representação de uma Classe

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



Representação de uma Interface

- Uma interface é semelhante a uma classe, mas não tem atributos
- Uma interface possui
  - Nome da Interface
  - Métodos da Interface
- Estereótipo
  - interface

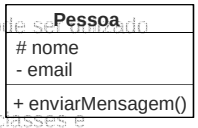


Tipos de Visibilidade

- Pública (+)
  - O atributo ou método pode ser utilizado por qualquer classe
- Protegida (#)
  - Somente a classe, sub-classes e classes amigas (ex. pacote) têm acesso
- Privada (-)
  - Somente a própria classe terá acesso

Tipos de Visibilidade

- Pública (+)
  - O atributo ou método pode ser utilizado por qualquer classe
- Protegida (#)
  - Somente a classe, sub-classes e classes amigas (ex. pacote) têm acesso
- Privada (-)
  - Somente a própria classe terá acesso



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 geralmente implementadas como chamadas de métodos
  - Nome = o nome do método
  - Informação = a lista de parâmetros
  - Requisitante = o método/objeto que realizou a chamada

Relacionamentos

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

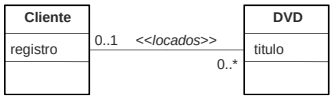
Associações

- 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 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*

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 as partes
  - Ou, as partes não existem sem o todo

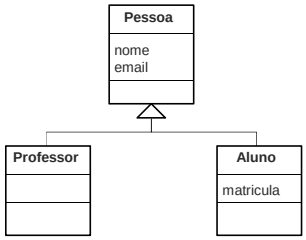
Representação da Composição

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

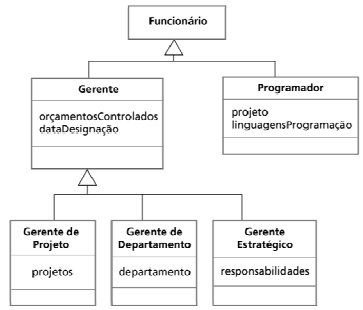
Especialização / Generalização

- Identificar super-classe (geral) e sub-classes (especializadas)
  - Semântica “é um”
  - Tudo que a classe geral pode fazer, as classes específicas também podem
- Atributos e métodos definidos na classe-mãe são **herdados** pelas classes-filhas

Especialização / Generalização



Especializações de Funcionário



Vantagens da Herança

- O gráfico de herança é uma fonte de conhecimento sobre o domínio do sistema
- É um mecanismo de abstração usado para classificar entidades
- Mecanismo de reuso em vários níveis
  - Como projeto e programação

Problemas com Herança

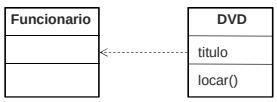
- Classes de objetos não são auto-contidas
  - Não podem ser compreendidas sem referência às suas super-classes
- Reusar gráficos da fase de análise pode ser ineficiente
  - Os gráficos de herança na análise, projeto e implementação têm diferentes funções (devem ser refinados)

Dependência

- Tipo menos comum de relacionamento
- Identifica uma ligação fraca entre objetos de duas classes

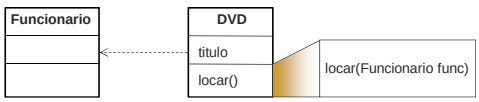
Dependência

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



Dependência

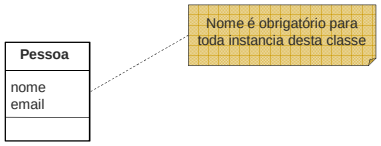
- Representado por uma reta tracejada entre duas classes
- Uma seta na extremidade indica o dependente



Notas

- Informativos
  - Têm função de comentários em classes, métodos ou atributos
  - Informa restrição de funcionalidade
  - Indicar condições para os relacionamentos, etc.

Notas



Bibliografia

- G. Booch, J. Rumbaugh, I. Jacobson. **UML, Guia do Usuário**. 2ª Ed., Editora Campus, 2005.
  - Capítulos 4, 8 e 9
- M. Fowler. **UML Essencial**, 2a Edição. Bookmann, 2000.
  - Capítulos 4 e 6