UML : Diagrama de Classes

Profa. Karin Becker
Engenharia de Software N
Instituto de Informática - UFRGS

Diagrama de Classes

- Descreve a **estrutura estática** de um sistema mostrando:
  - os tipos objetos pertencentes ao sistema
  - os tipos de relacionamentos entre esses objetos
  - os atributos que caracterizam cada objeto
  - as operações que caracterizam cada objeto
- é um esquema que descreve muitas instâncias de objetos

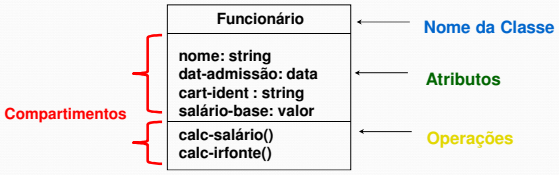
Diagrama de Classes

- Organiza elementos "classificadores"
  - Classe, Classe Parametrizada
  - Interface, Pacote, Objeto, etc
  - Associações
    - Binárias, ternárias, etc
    - Composição, generalização
    - Dependências (Estereótipos)
  - Restrições
    - Anotações
    - Expressões em OCL

Conceitos Básicos

Classe

- descrição de um conjunto de objetos que compartilham os mesmos atributos, operações, relações e semântica
  - Apenas nome é obrigatório



- outros compartimentos podem ser acrescentados (e.g. responsabilidades, exceções, eventos, etc)
  - Nem sempre disponível em ferramentas de modelagem

Classe

- O nome de uma classe distingue uma classe de outra
  - nome simples: nome sozinho
  - nome com caminho: o nome da classe é precedido pelo nome do pacote em que a classe existe.

Diagrama de Classes: Atributos

- Cada objeto de uma classe possui um estado, representado pelos valores associado a cada um dos atributos definidos para a classe

Sintaxe para Atributos:
[visibilidade] **nome** [[multiplicidade]] [:tipo]
[= valor inicial] [{propriedades}]
* atributos de classe são sublinhados

Exemplos:
nome
nome: String
idade: Inteiro = 0
cpf : Inteiro {frozen}
Endereço [0..2] : String
nroCorrentistas: Inteiro

Diagrama de Classes: Atributos

- Visibilidade
  - público (+)
  - protegido (#)
  - privado (-)
- Propriedades
  - changeable: não há restrições quanto à modificação do valor do atributo
  - frozen: o valor do atributo não pode ser alterado
  - addOnly: válida para atributos com multiplicidade superior a um, onde o valor atribuído a cada ocorrência de um atributo não pode ser alterado ou removido.

Diagrama de Classes: Operações

- Uma operação é a especificação de um serviço que pode ser requisitado a qualquer objeto da classe
- Distinção entre operações de classes e objetos:
  - **Operação de objeto**: atua sobre um objeto (instância);
  - **Operação de classe**: atua sobre a classe (conjunto de objetos).
    - Exemplos: criação de um novo objeto da classe, pesquisa sobre os objetos da classe, etc.

Diagrama de Classes: Operações

Sintaxe para Operações:

[visibilidade] **nome** [(lista-de-parâmetros)] [:tipo-retorno] {[propriedades]}

* operações de classe são sublinhadas

Exemplos:

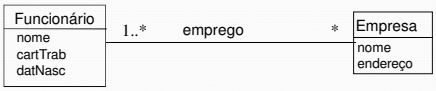
pagar
pagar(valor)
pagar(valor: numérico): numerico
lerTemperatura() : numerico
calcÁrea(Lado:double, Altura:double) : double
valorPadrao(): Inteiro

Diagrama de Classes: Relacionamentos

- Básico
  - Associação
- Avançado
  - Agregação
  - Generalização/Especialização
  - Dependência

Associação

- Uma associação é uma relação estrutural que descreve um conjunto de ligações, onde uma ligação é uma conexão entre objetos
- bidirecional



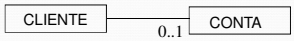
Associação: cardinalidade

- Cardinalidade especifica quantas instâncias de uma classe se relacionam com uma dada instância da outra classe.
  - Número: estabelece o número exato de objetos relacionados
    - Ex. 2
  - Intervalo de Valores: define a multiplicidade mínima e máxima
    - Ex.: 1..5
  - Máxima Ilimitada :
    - Ex.: 0..*, 0..n, n

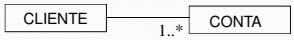
Associação: cardinalidade

- (1-1): cliente tem sempre 1 (e somente 1) conta

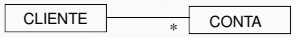
- (0-1): cliente pode ter 1 (e no máximo 1) conta



- (1-N): cliente tem sempre 1 conta, podendo ter mais

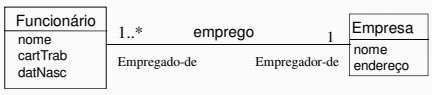


- (0-N): cliente pode não ter 1 conta, podendo ter mais que uma



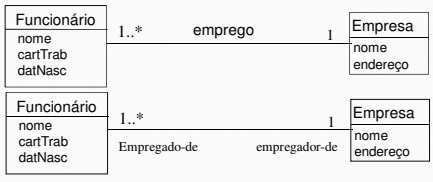
Associação: nome e papel

- Papel (opcional)
- nome relacionamento (opcional)



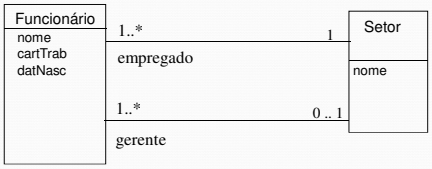
Associação: nome e papel

- Papel (opcional)
- nome relacionamento (opcional)



Associação: nome e papel

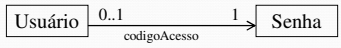
- Escolha a opção que der melhor clareza ao que quer modelar

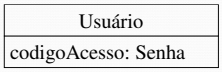


- Obs: as ferramentas CASE costumam usar nome dos papéis na geração de código

Navegação de Associações

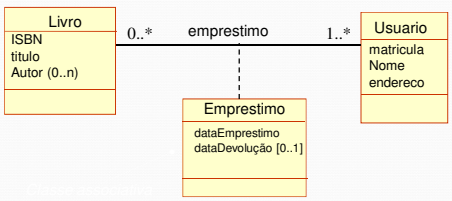
- por definição, a navegação entre classes associadas é bidirecional
- por conveniência, a navegação pode ser restringida a uma única direção
- definição a ser postergada até o projeto detalhado





Classe Associativa

- uma associação para a qual seja necessário expressar propriedades de uma classe (e.g. atributos, operações)



Exercício : Venda de Produtos

A Firma Limpex firma vende produtos de limpeza, e deseja melhor controlar os produtos que vende, seus clientes e suas vendas.

- Cada produto é caracterizado por um código, nome do produto, categoria, e seu preço. A categoria é uma classificação criada pela própria firma (ex. detergente, sabão em pó, sabonete), e somente são vendidos produtos pertencentes a categorias previamente definidas.
- A firma possui informações sobre todos seus clientes. Cada cliente é caracterizado por seu código interno, nome do cliente, endereço, um ou mais telefone, e o seu limite de crédito.
- Registra-se a informação de toda e qualquer venda feita a um cliente. Cada venda possui um número, e guarda-se a data da venda. Cada venda envolve pelo menos um produto, e para cada produto, indica-se a quantidade deste pedida.
- Com estas informações, deseja-se poder representar a nota fiscal ao lado

Limpex S.A.
Comendador Oliveira 27
CGC. 7654321/09

Controle Interno
Pedido 98765

Data: 14/6/99

Código: C-1234
Nome: Joao da Silva
Endereço: Anita Garibaldi 8765, Porto Alegre, 90345-678, RS
Telefone: 051-2345678

Cód.	Descrição	Qt.	Total
123	Limpa-Tudo	1	10,00
345	Detergentex	2	4,00
678	Escovex	3	3,00
		Total	17,00

Exercício : BROLIWOOD

BROLIWOOD possui diversos estúdios cinematográficos, cada um caracterizado por um nome, nome do(s) sócio(s), data de fundação, e o faturamento do ano anterior. Estes estúdios produzem filmes que possuem um nome, o número de meses que levou sendo feito, o ano de lançamento, custo total de produção do filme, e, quando for lançado, faturamento. Em cada filme atuam atores, que possuem um nome artístico, uma nacionalidade, idade, sexo, e um conjunto de tipos de papéis para o qual seu tipo físico é aconselhável (ex: avó, mocinha jovem, galã com idade avançada, adolescente). Estes tipos de papéis não são pré-definidos, constitutindo uma lista preenchida a critério de cada ator para fins de casting. Em cada filme onde atua, um ator ganha um cachê, e desempenha um personagem que possui um nome. Estúdios podem existir mesmo que ainda não tiverem produzido um filme, mas só são considerados atores que já atuaram em pelo menos um filme. Só são registrados filmes já produzidos.

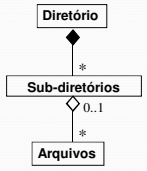
Conceitos Avançados

Conceitos Avançados

- Associação
  - Agregação
  - Especialização/generalização
  - Dependência
- Classificadores
  - Classe abstrata
  - interface

Agregação

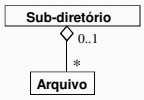
- Indica que o relacionamento tem uma semântica (significado) especial
  - Todo-parte
- Faz mais sentido em modelos conceituais do que em modelos de projeto



- **Agregação é transitiva**
  Se A faz parte de B e B faz parte de C então A faz parte de C
- **Agregação é não-simétrica**
  Se A faz parte de B então B não faz parte de A

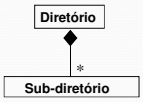
Agregação Simples

- A existência de um objeto componente não depende da existência do objeto agregado
  - questionável do ponto de vista semântico do mecanismo de abstração agregação
  - "placebo semântico"
- Um objeto componente pode ser compartilhado com outro objeto agregado



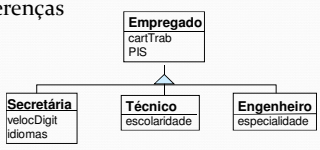
Agregação de Composição

- Quando o todo é criado, as suas partes com multiplicidade não fixa podem ser criadas posteriormente
- Quando uma parte é criada, a sua existência deve ser coincidente com a existência do todo, a não ser que seja explicitamente removida antes da eliminação do todo
- Quando o todo é eliminado, as suas partes também devem ser eliminadas



Especialização/generalização

- permite modelar aspectos semelhantes entre classes, preservando suas diferenças
- Herança
- Generalizar
- Especializar

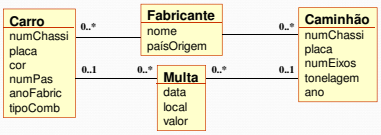


- Usado tanto na modelagem conceitual, quanto em etapas (avançadas) de projeto (e de implementação)

Generalizar

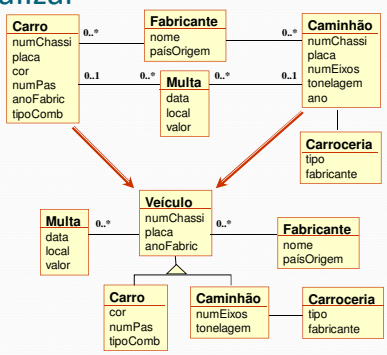
- Identificar classes com propriedades semelhantes
- Definir uma nova classe com as propriedades comuns
- As classes originais tornam-se subclasses da nova classe e herdam as propriedades desta
  - Atributos
  - Operações
  - relacionamentos
- Manter nas classes originais as propriedades não comuns

Generalizar



Quais os problemas desta modelagem?
Poderia ser melhorada através da generalização?

Generalizar

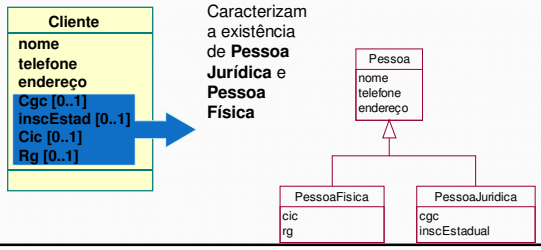


Especializar

- Definir uma ou mais subclasses a partir de uma classe existente
- Adicionar propriedades específicas da nova subclasse
  - Atributos
  - Operações
  - relacionamentos
- Propriedades comuns ficam ligadas à superclasse
- Pode existir mais de um tipo de especialização com base em diferentes características.
  - Cada hierarquia de generalização/especialização deve abranger uma única característica.

Especializar

- Atributos opcionais podem indicar a necessidade de especialização de uma classe

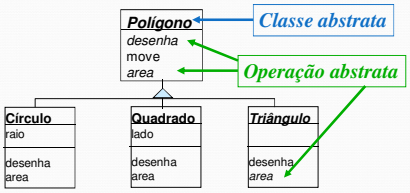


Exercício : Aeroclube

Um aeroclube tem interese em manter o controle sobre todos os seus sócios, bem como sobre as aulas práticas realizadas pelos seus alunos.
• Os sócios dividem-se em pilotos, instrutores e alunos de pilotagem. Todos sócios são caracterizados pelo número de matrícula, nome, endereço e idade. Os pilotos possuem um número de brevê. Os instrutores são pilotos com formação adicional de instrutor, devendo ser registrados o nome do curso de sua formação e a data de obtenção do diploma. A escola só ministra cursos básicos, e portanto não há pilotos ou instrutores que são alunos de cursos avançados.
· Alunos são sócios que seguem uma formação teórica-prática visando obtenção de um breve. Para estes alunos, deseja-se unicamente registrar todos os seus vôos para contabilização de horas necessárias à obtenção do brevê.Para cada vôo de um aluno, registra-se a data, instrutor (deve ser um dos instrutores do clube), data, hora de saída e de chegada, bem como o parecer do instrutor sobre o voo.

Classe Abstrata

- Não pode ser instanciada
- projetada para servir como geradoras de outras classes (fatoramento de interface comum)
- pode ter implementações parciais



Exemplo: Java.Util

- AbstractCollection
  - This class provides a skeletal implementation of the Collection interface, to minimize the effort required to implement this interface
- AbstractList
  - This class provides a skeletal implementation of the List interface to minimize the effort required to implement this interface backed by a "random access" data store (such as an array).
- Vector, ArrayList

http://download.oracle.com/javase/1,5,0/docs/api/java/util/AbstractCollection.html

Exemplo

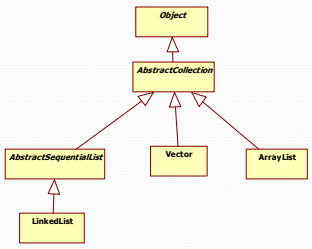
java.util
Class AbstractList<E>

java.lang.Object
  └ java.util.AbstractCollection<E>
    └ java.util.AbstractList<E>

All Implemented Interfaces:
Iterable<E>, Collection<E>, List<E>

Direct Known Subclasses:
AbstractSequentialList, ArrayList, Vector

interface in java.util

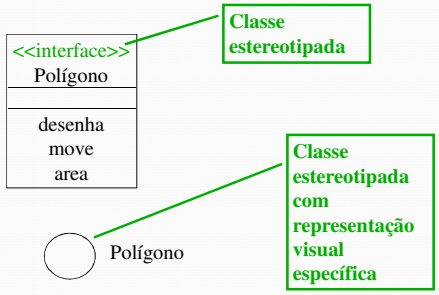


Interface

- coleção de operações usadas para especificar serviços de classe ou componente
  - operações
  - constantes
- não pode possuir variáveis nem implementação para operações
- pode estar ligada a outras interfaces/classes por dependência, generalização e associações
- Usado na definição de arquiteturas e em etapas (avançadas) de projeto

Interface

- Representada por uma classe estereotipada



Exemplo: Java.Lang (Comparable)

- Comparable: This interface imposes a total ordering on the objects of each class that implements it. This ordering is referred to as the class's natural ordering, and the class's compareTo method is referred to as its natural comparison method

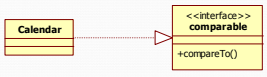
java.lang
Interface Comparable<T>

All Known Subinterfaces:
Delayed, Name, ScheduledFuture<V>

All Known Implementing Classes:
AuthenticatorRequestorType, BigDecimal, BigInteger, Boolean, Byte, ByteBuffer, Calendar, Character, CharBuffer, Charset, CollationKey, CompositeName, CompoundName, Date, Date, Double, DoubleBuffer, ElementType, Enum, File, Float, FloatBuffer, FormatterBigDecimalLayoutForm, FormSubmitEvent.MethodType, GregorianCalendar, IntBuffer, Integer, JTable.PrintMode, KeyRep.Type, LdapName, Long, LongBuffer, MappedByteBuffer, MemoryType, ObjectStreamField, Proxy.Type, Rdn, RetentionPolicy, RoundingMode, Short, ShortBuffer, SSLEngineResult.HandshakeStatus, SSLEngineResult.Status, String, Thread.State, Time, Timestamp, TimeUnit, URI, UUID

Interface

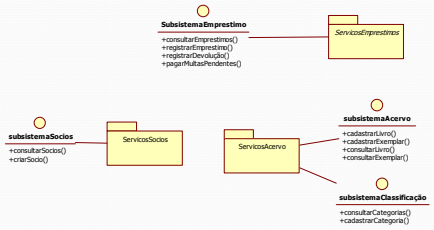
- Uma classe realiza uma interface
  - Notação : textual e associação "realize"



  - Notação: icônica Pirulito

Exemplo: Interface

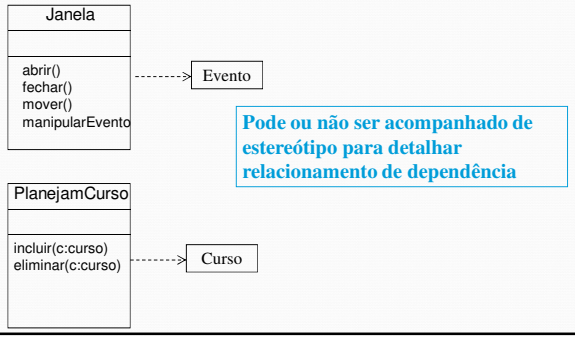
Exemplo Biblioteca com 3 subsistemas, cada uma definida externamente por suas interfaces



Dependência

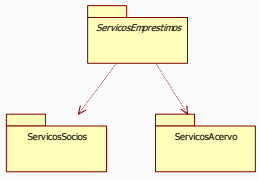
- Estabelece um relacionamento entre duas classes, uma independente e outro dependente
  - uma mudança na classe independente poderá afetar a classe dependente
  - sem semântica definida
  - Estereótipos
- utilizada quando se deseja representar a utilização de uma classe por outra classe
  - argumentos de operações
  - implementação de interfaces (“realização”)
  - uso no código de operações
  - etc

Dependência



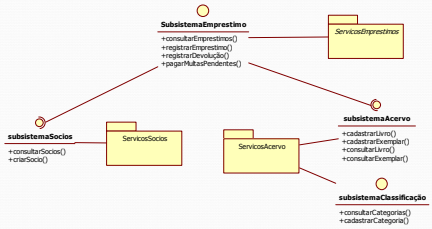
Exemplo: Dependência

Exemplo Biblioteca com 3 subsistemas, onde um deles é dependente dos demais



Exemplo: Dependência

Exemplo Biblioteca com 3 subsistemas, cujas interfaces têm dependências (notação socket&ball)



Para saber mais ...

- Fowler, M. ; Scott, K. UML Essencial, Bookman, 2005.
  Livro de referência sobre UML mas descreve apenas a notação e os modelos e não o processo de construí-los. Foca na UML 1.x.
- Ambler, S. , *The Elements of UML 2.0 Style* , Cambridge, 2005.
  Discute cada tipo de diagrama, com dicas de bom uso. Bom para iniciantes, mas se concentra na notação.
- Larman, Craig. Utilizando UML e Padrões - Uma Introdução à Análise e ao Projeto Orientados a Objetos, Bookman.
  Descreve passo a passo UM processo de Análise e Projeto Orientados a Objetos utilizando a notação UML. Aborda também o uso de padrões de projeto.
  As duas primeiras edições são mais objetivas e sucintas, a terceira é mais focada em desenvolvimento iterativo e ágil.
- http://www.uml.org/