T É C N I C A 3

MODELAGEM CONCEITUAL

# GENERALIZAÇÃO/ESPECIALIZAÇÃO, AGREGAÇÃO E COMPOSIÇÃO

## Generalização/Especialização

Herança é o termo em orientação a objetos que se refere à criação de novas classes a partir de existentes onde os atributos e operações comuns são agrupados para fins de reutilização de código. Observe as duas classes abaixo:

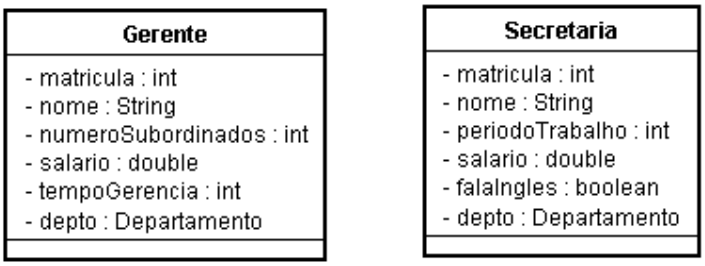



Figura 1 – classes independentes Gerente e Secretaria

Perceba que há muitos atributos comuns entre elas e isso demanda muito trabalho para definir e alterá-los. Além disso, o que estas duas classes têm em comum ? Elas fazem parte de um sistema de controle de pessoal e representam os funcionários de uma empresa. Com base nesses fatos, aplicaremos a técnica da generalização para eliminar as duplicidades e deixar o modelo mais elegante.

### 1- Criação da superclasse

Para implementar a generalização, criamos uma terceira classe generalista para agrupar os membros comuns de Gerente e Secretaria (matricula, nome, salario e depto). Essa classe deve receber um nome que também seja uma generalização de gerente e secretária e, por isso, escolhemos “Funcionario”.

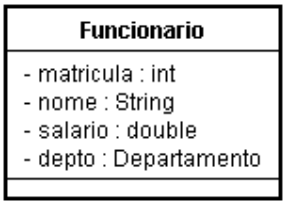



Figura 2 – superclasse Funcionario

### 2- Retirar elementos comuns das subclasses

A classe pai, também chamada de superclasse, ou ancestral, é quem possui os atributos comuns. A classe filha, ou subclasse, é a classe que herda os membros da superclasse e ainda possui os seus específicos (por isso, a subclasse tende a ser maior que a

superclasse). Assim, o passo seguinte é retirar os atributos comuns das duas subclasses, deixando apenas os específicos:

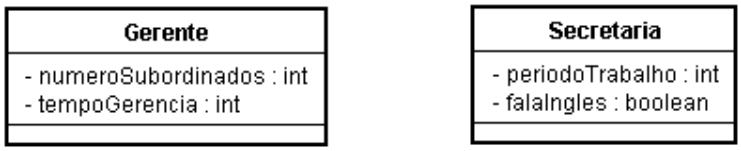



Figura 3 – subclasses com atributos comuns removidos

**3- Fazer o relacionamento de generalização entre subclasses e superclasse**

Criamos um relacionamento de generalização (cuja notação é uma reta com um triângulo na ponta) entre Gerente e Funcionário e outro entre Secretária e Funcionário. A ponta da seta sempre deve apontar para a superclasse.

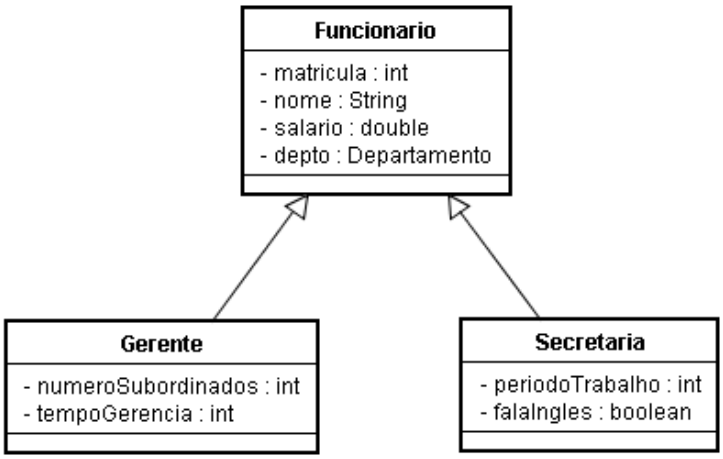



Figura 4 – notação dos relacionamentos

Outra forma de representação pode ser vista na figura abaixo. Para fazer isso na ferramenta JUDE, selecione um relacionamento, clique na ponta da seta, arraste e solte sobre a ponta da outra seta.

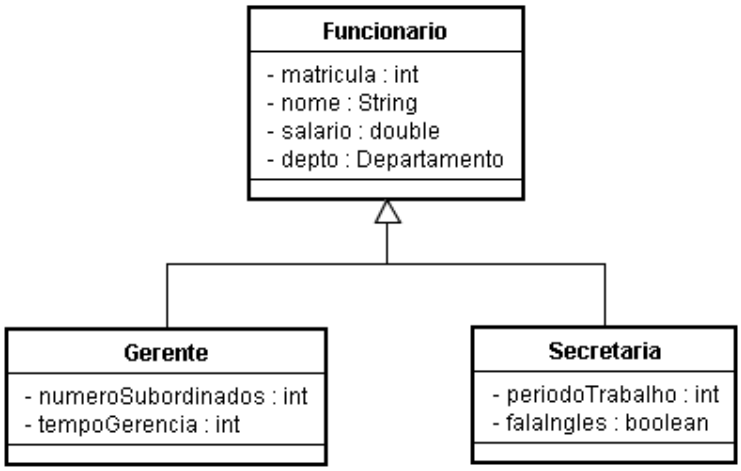



Figura 5 – outra forma de notação de duas generalizações

**4- Validar os relacionamentos**

Ao contrário do relacionamento de associação, a generalização não precisa de um nome, pois implicitamente o relacionamento deve ser lido como "é um". Isso inclusive serve para conferir se o já o relacionamento está válido ou se o nome da classe generalista foi escolhido corretamente. Assim, as seguintes perguntas devem ter uma resposta afirmativa para que nosso relacionamento seja válido:

- Gerente "é um" funcionário ? - Resposta: SIm
- Secretária "é um" funcionário ? - Resposta: Sim

**5- Definir superclasse como abstrata (se for o caso)**

Muitas vezes definimos uma classe que não representa verdadeiramente algo do mundo real que precise ser instanciado (usado em programas). É o caso da classe Funcionário criada neste exemplo. Na prática, objetos do tipo Funcionario nunca serão usados, pois instanciaremos apenas objetos a partir de Gerente e Secretaria. Funcionário é apenas uma classe que surgiu da necessidade de uma generalização. Classes assim são chamadas de abstratas. Na UML, para indicar que uma classe é abstrata, seu nome deve ficar em itálico. Para fazer isso no JUDE, selecione a classe e mude seu atributo *Abstract* para *true*.

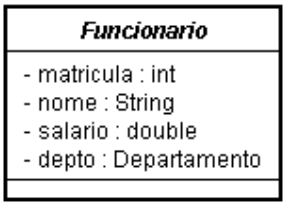



Figura 6 - classe abstrata

## Agregação

É um tipo especial de associação onde tenta-se demonstrar que as informações de um objeto precisam ser complementadas pelas informações contidas em um ou mais objetos de outra classe, demonstrando um relacionamento todo-parte entre elas. É interessante ressaltar que para ser agregação, as classes podem viver independentemente. Vejamos um exemplo onde temos as classes Time e Jogador. Um time pode ser composto de nenhum ou vários jogadores. Um jogador pode estar contido no máximo em um time.


Figura 7 - classes Time e Jogador

Poderíamos apenas fazer uma associação entre Time e Jogador e extrair as cardinalidades, mas queremos dar um enfoque que as informações do time são completadas pelos seus jogadores, ou seja, que os jogadores fazem parte do time. Entretanto, se eu excluir o time, os jogadores continuam existindo. Assim, utilizamos a notação da agregação que é uma reta com um losango na ponta ligada à classe do lado "todo" (que contém):

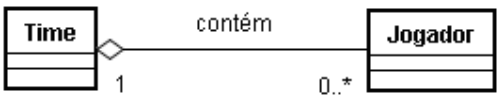



Figura 8 - agregação entre Time e Jogador

## Composição

A composição é uma variação da agregação onde se tenta representar um vínculo mais forte entre os objetos todo-parte ao ponto de um não existir sem o outro. Vejamos o exemplo entre as classes Revista e Edição. Uma revista deve ser composta de no mínimo um artigo, mas pode conter vários. Um artigo obrigatoriamente deve pertencer a uma e somente uma revista.


Figura 9 - classes Revista e Artigo

Poderíamos fazer apenas uma associação entre Revista e Artigo, mas queremos demonstrar que revista e artigos formam um único conjunto e que um não existe sem o outro. O símbolo da composição diferencia-se graficamente da agregação por utilizar um losango preenchido que, da mesma forma, deve ficar do lado do objeto -todo:

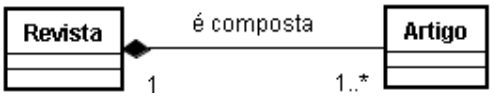



Figura 10 – composição entre Revista e Artigo

Uma revista pode ter vários artigos, mas se a revista for excluída, teremos que excluir também os artigos. Não faz sentido ter um objeto artigo que ele esteja vinculado a uma revista. Sua única razão de existir é "compôr" a revista.

### Agregação x Composição x Associação

Imagine um cenário onde temos duas classes "A" e "B" e estamos na dúvida de qual relacionamento colocar entre elas. Inicie fazendo a seguinte pergunta:

1 - Se “A” for excluído, terei que excluir também o “B” ?
- Sim = utilize relacionamento de composição
- Não = pode ser agregação ou simplesmente uma associação. Vá para a pergunta 2

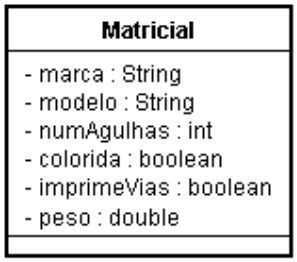
2 – “B” tem alguma utilidade sozinha?
- Sim = utilize uma associação comum
- Não = utilize relacionamento de agregação

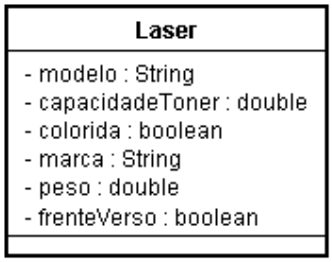
### Referências

GUEDES, Gilleanes. UML, uma abordagem prática. Ed. Novatec: São Paulo, 2004.
NOGUEIRA, Admilson. UML - Unified Modeling Language - Generalização, agregação, composição e dependência. Disponível na internet em http://www.linhadecodigo.com.br/Artigo.aspx?id=943. Acesso em 21/03/2009

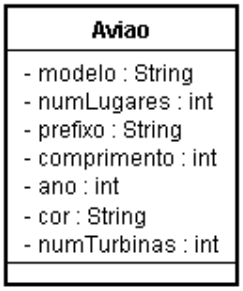
## Exercícios Propostos

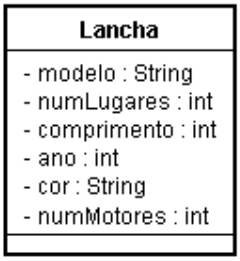
**1.** Usando seus conhecimentos de generalização/especialização, analise as duas classes abaixo e proponha um novo modelo que elimine as duplicações



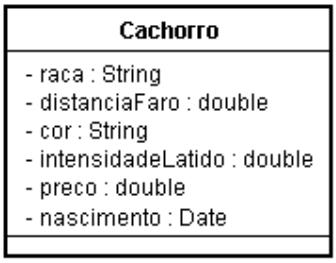




**2.** Usando seus conhecimentos de generalização/especialização, analise as três classes abaixo e proponha um novo modelo que elimine as duplicações

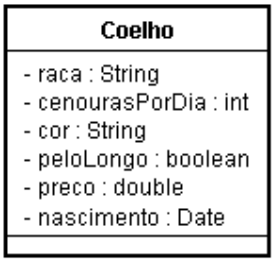




**3.** Usando seus conhecimentos de generalização/especialização, analise as duas classes abaixo e proponha um novo modelo que elimine as duplicações








**4.** Em um sistema foram identificadas duas classes: Pessoa e Coração. Uma pessoa deve ter um e somente um coração e um coração só pode pertencer a uma pessoa. Com base no enunciado acima, faça a representação das duas classes (atributos e métodos não são necessários) e seu relacionamento (com nome e cardinalidades).

**5.** Em um sistema para uma concessionária foram identificadas as classes Carro e Autopeça. Um carro pode possuir uma ou várias autopeças. Uma autopeça serve para apenas um carro. Com base no enunciado acima, faça a representação das duas classes (atributos e métodos não são necessários) e seu relacionamento (com nome e cardinalidades).

**6.** Em um sistema de vendas foram identificadas as classes Pedido e ItemPedido. Um pedido deve conter no mínimo um item ou vários. Um item deve obrigatoriamente pertencer a um único pedido. Com base no enunciado acima, faça a representação das

duas classes (atributos e métodos não são necessários) e seu relacionamento (com nome e cardinalidades).

**7.** Em um sistema para uma editora foram identificadas as classes Livro, Capítulo e Página. Um livro é composto nenhum ou vários capítulos. Um capítulo deve obrigatoriamente pertencer a um único livro. Um capítulo contém uma (no mínimo) ou mais páginas. Uma página não necessariamente precisa pertencer a um capítulo, mas se pertencer, pode ser a mais de um. Com base no enunciado acima, faça a representação das duas classes (atributos e métodos não são necessários) e seu relacionamento (com nome e cardinalidades).

**8.** Faça a representação do relacionamento entre as classes País, Estado e Cidade. Um pais é composto de vários estados (pelo menos um estado é requerido). Estados podem ter nenhuma ou várias cidades. Uma cidade é obrigada a pertencer a um, e somente um estado. Um estado obrigatoriamente tem que pertencer a um país