Diagramas de Classes e O Paradigma da Orientação a Objetos usando UML

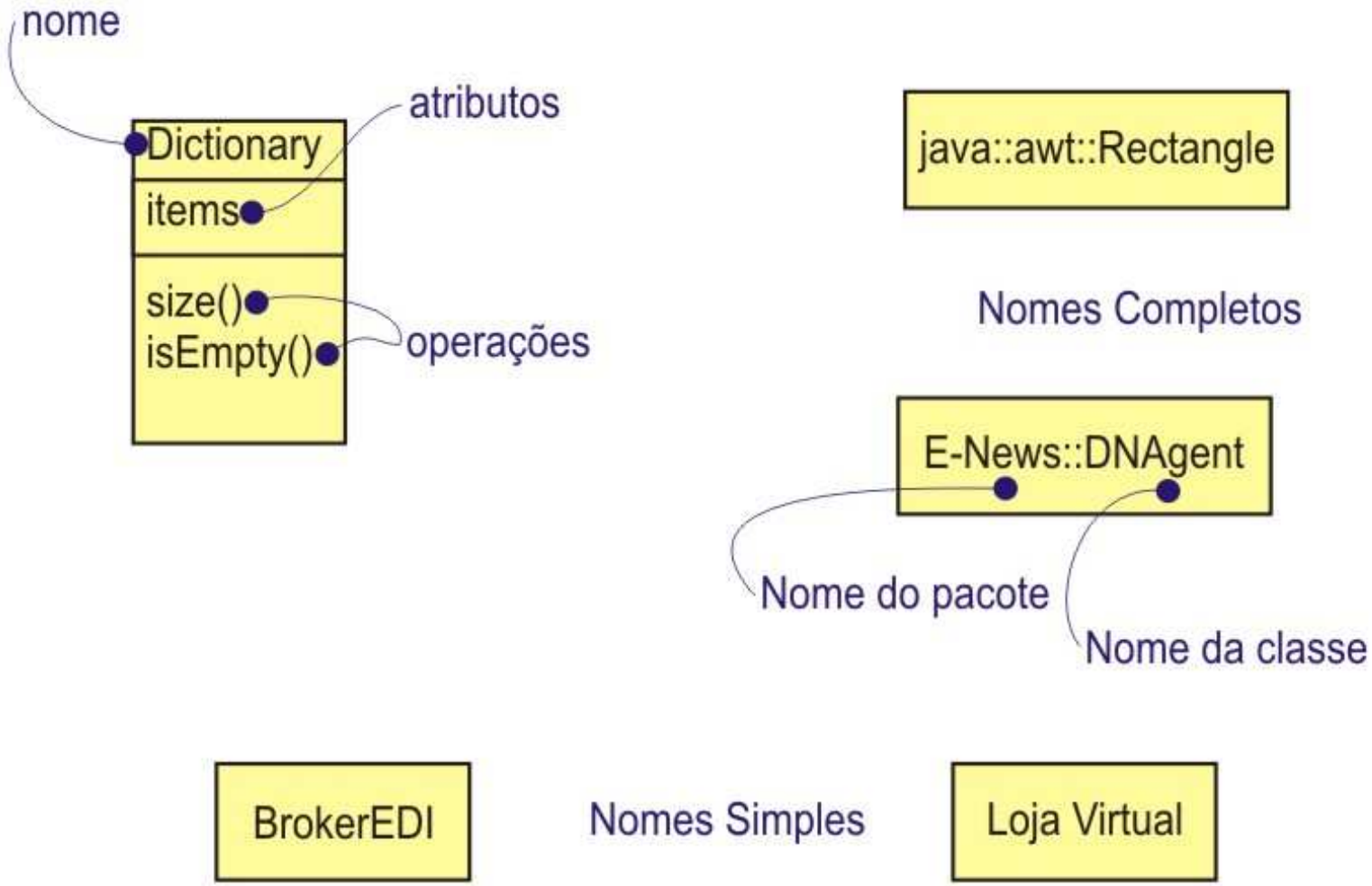
Prof. Ricardo A. Ramos

UML – Unified Modeling Language

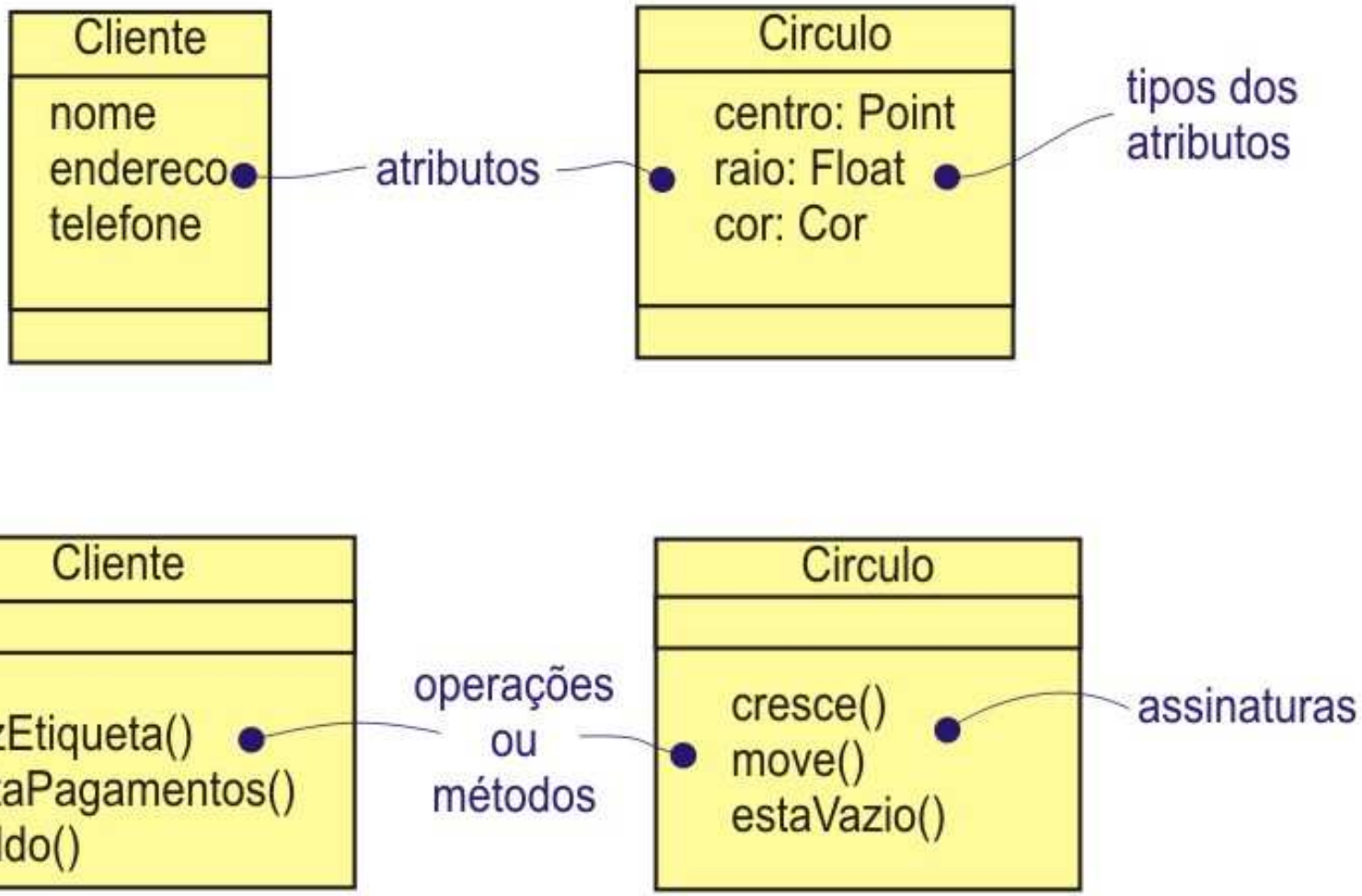
- É uma linguagem para especificação, construção, visualização e documentação de sistemas de software;

- É a união da sintaxe gráfica de vários métodos, com vários símbolos removidos e vários adicionados;

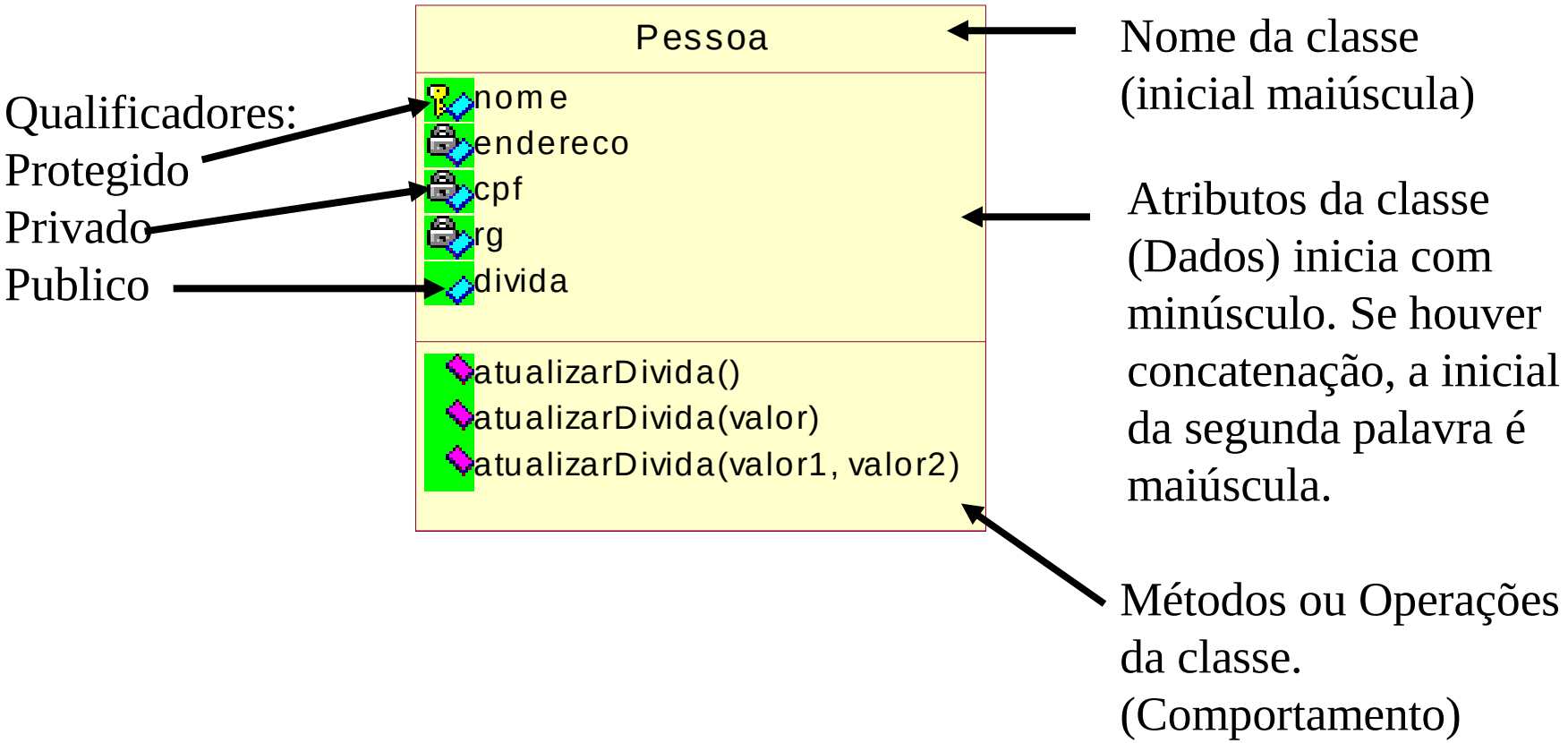
UML - Classes


nome
atributos
Dictionary
items
size()
isEmpty()
operações
java::awt::Rectangle
Nomes Completos
E-News::DNAgent
Nome do pacote
Nome da classe
BrokerEDI
Nomes Simples
Loja Virtual

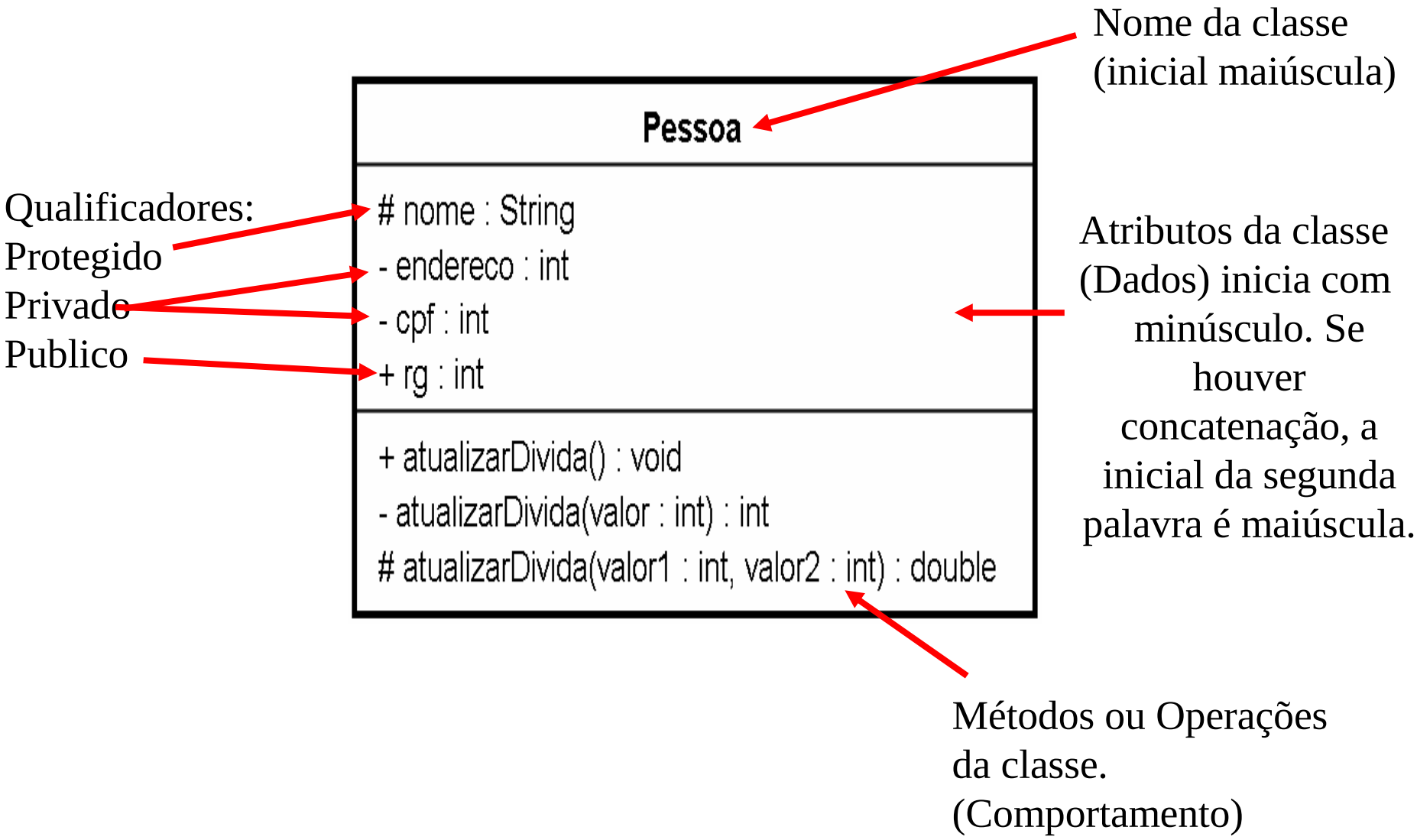
UML - Classes


Cliente
nome
endereco
telefone
atributos
Circulo
centro: Point
raio: Float
cor: Cor
tipos dos
atributos
Cliente
fazEtiqueta()
listaPagamentos()
saldo()
operações
ou
métodos
Circulo
cresce()
move()
estaVazio()
assinaturas

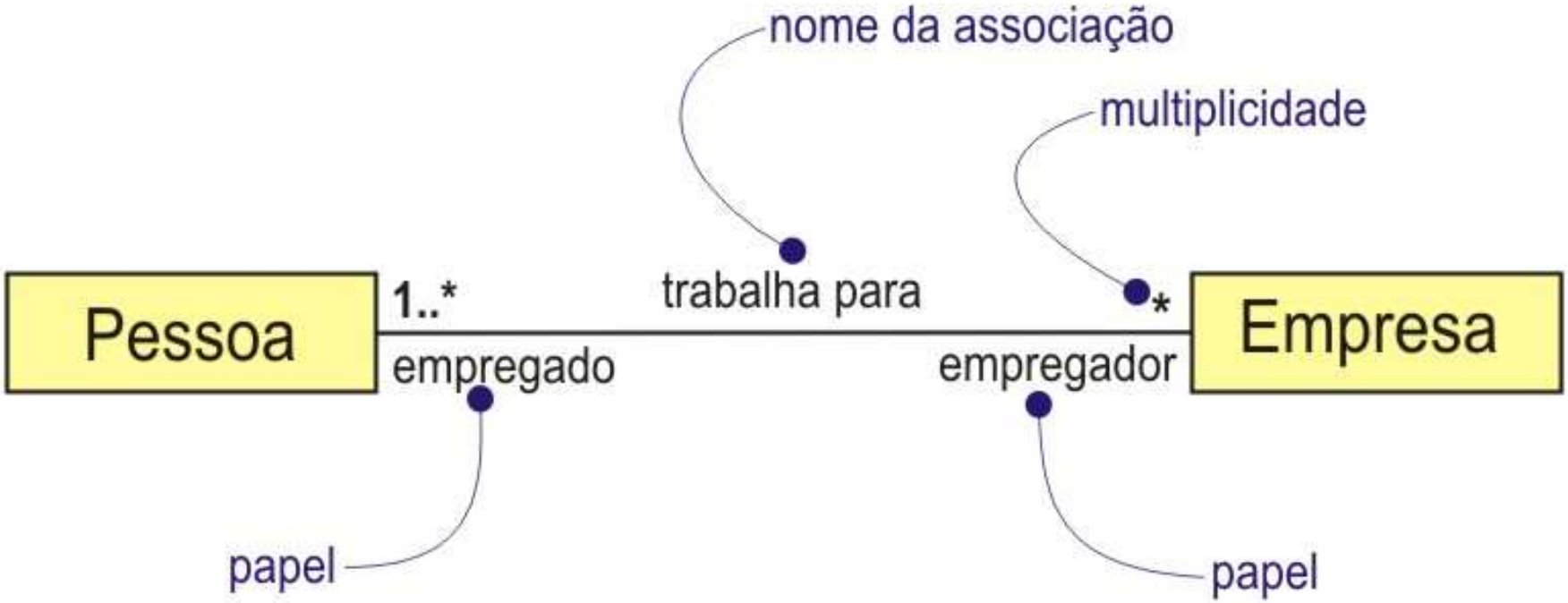
UML – Classe (no Rational Rose)



UML – Classe (no astah)

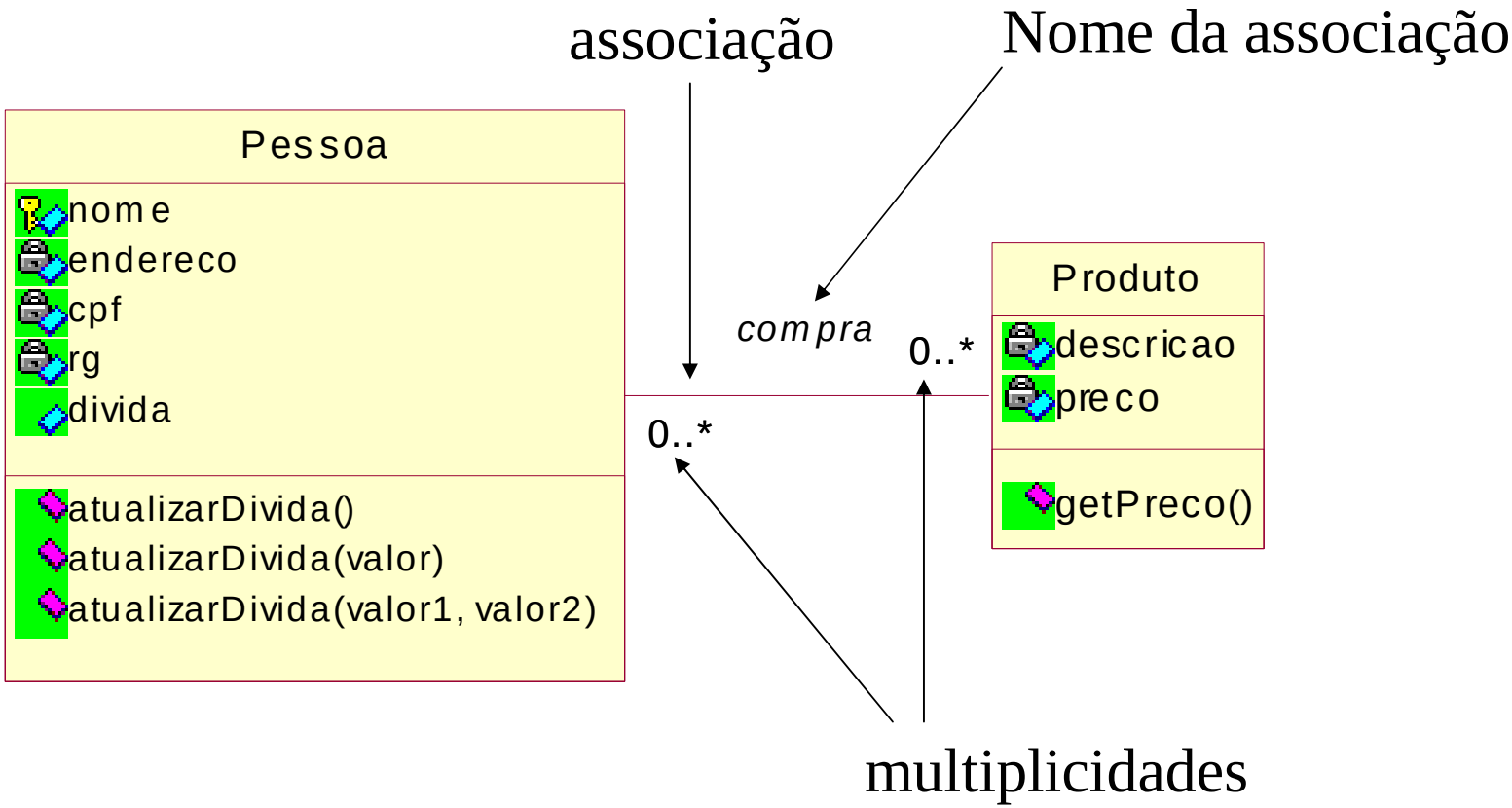


Relacionamentos - Associação



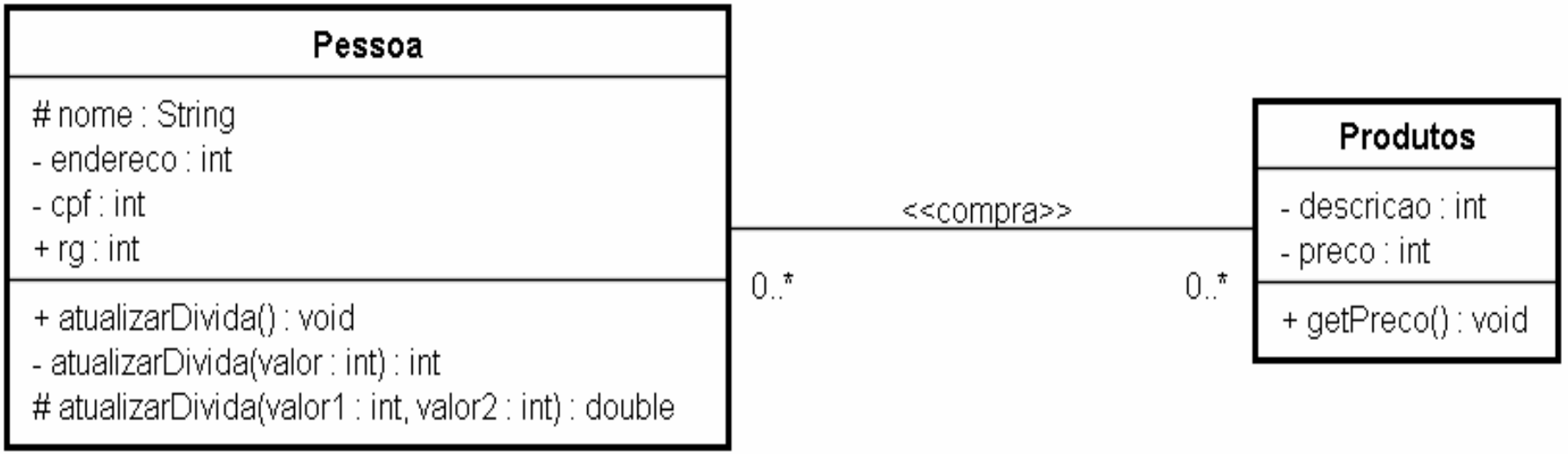
Relacionamentos – Associação

(no Rational Rose)

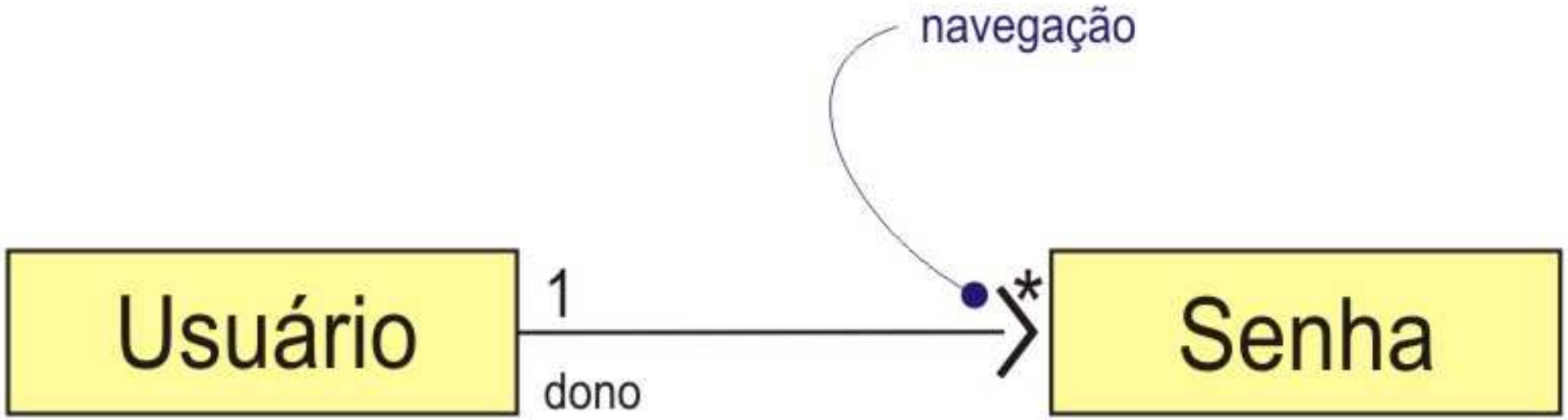


Relacionamentos – Associação

(no astah)



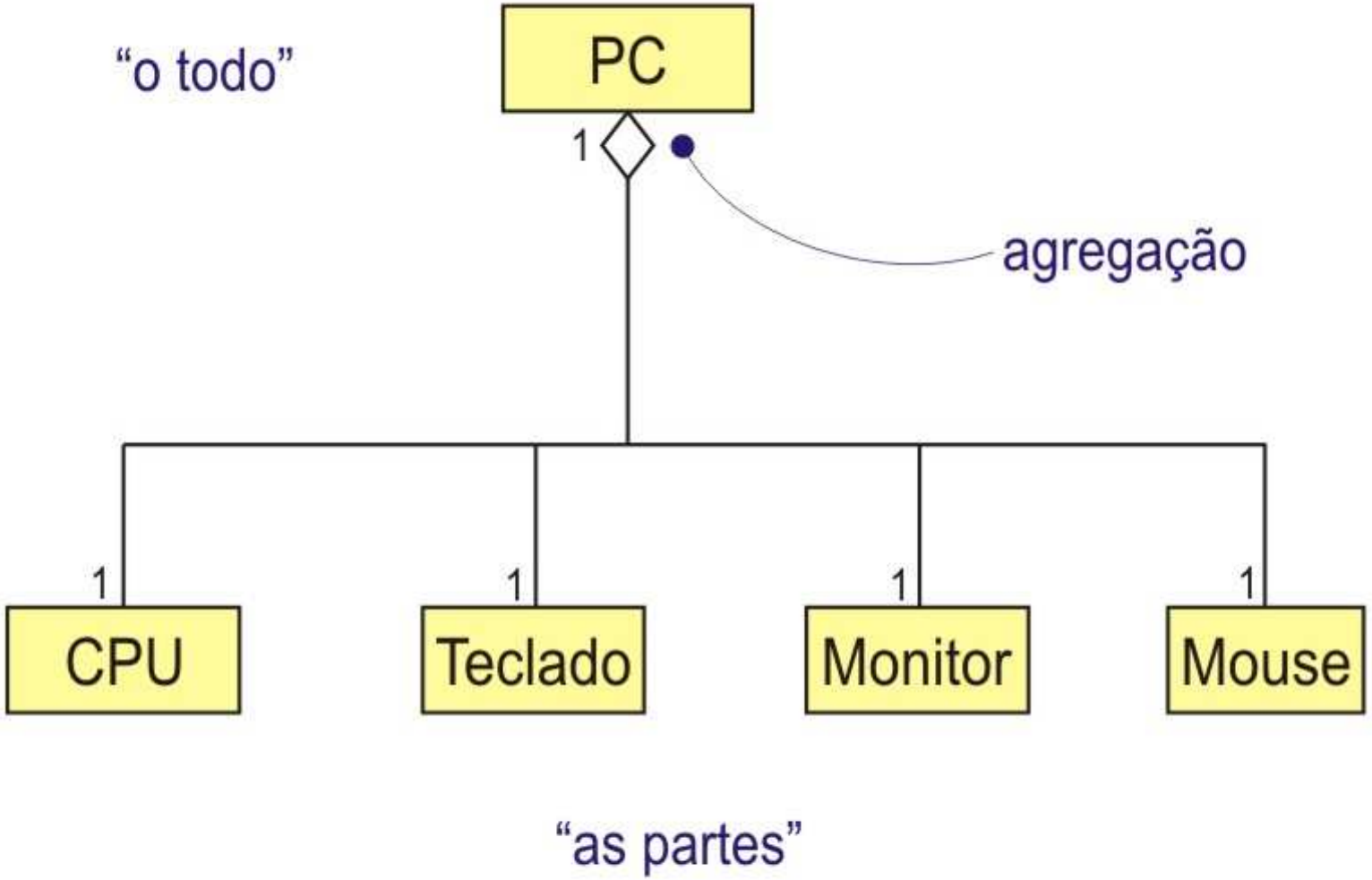
Associação com navegação



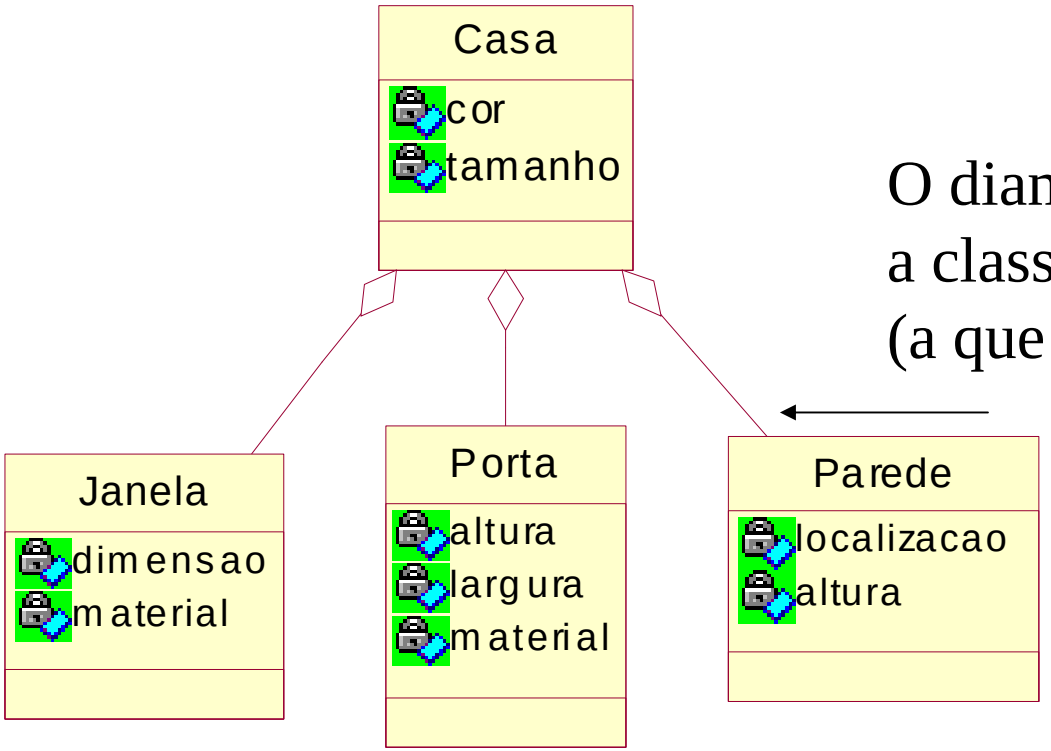
Agregação (simples)

- A associação entre classes sem agregação reflete que ambas as classes se encontram no mesmo nível conceitual.
- Por outro lado, uma relação de associação com agregação traduz que existe uma relação do tipo “is-part-of” ou “has-a”, o que corresponde ao fato de uma instância de determinada classe possuir ou ser composta por várias instâncias de outra classe.
- A informação de agregação é representada por um losango colocado junto à classe que representa o elemento agregador ou “o todo”.

Relacionamentos - Agregação

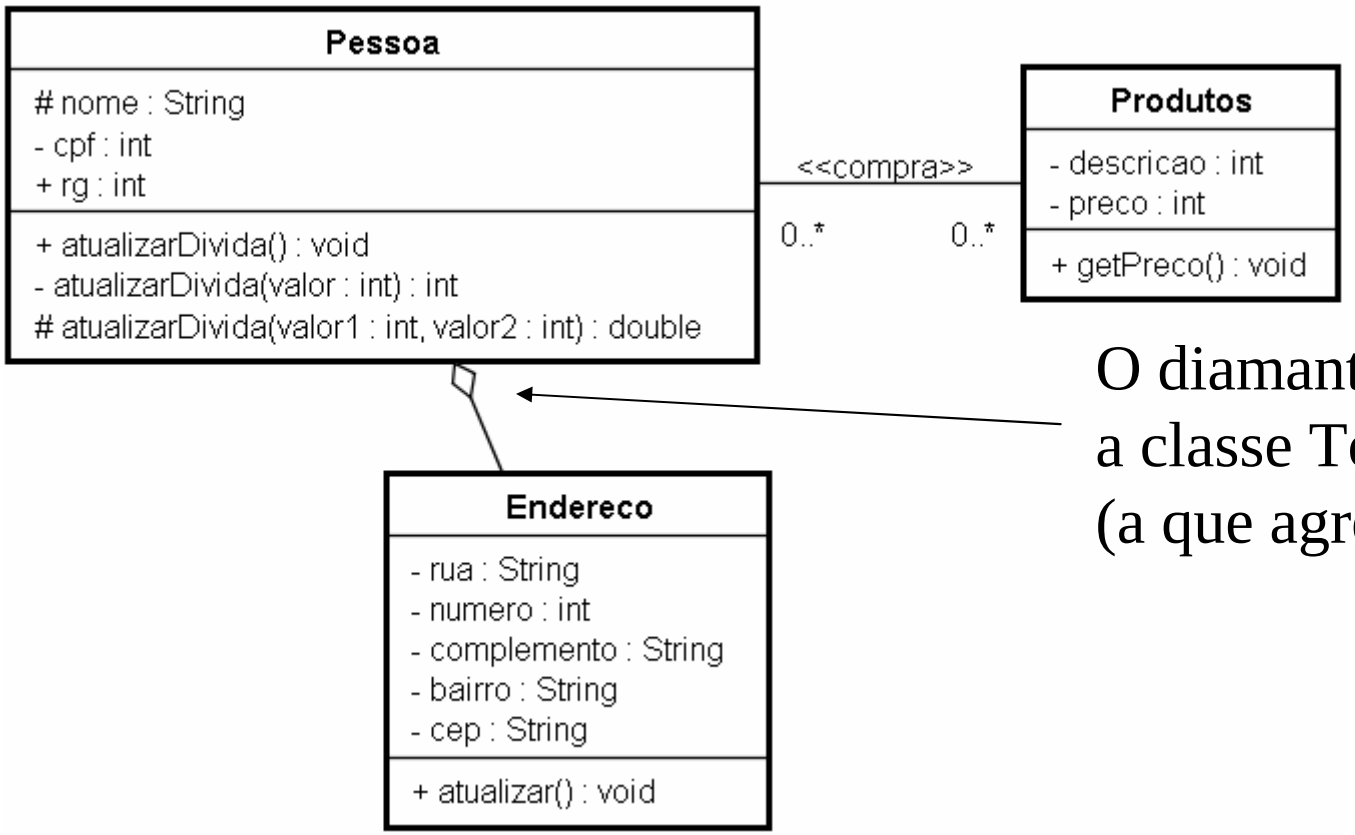


Relacionamentos - Agregação



O diamante indica
a classe Todo
(a que agrega)

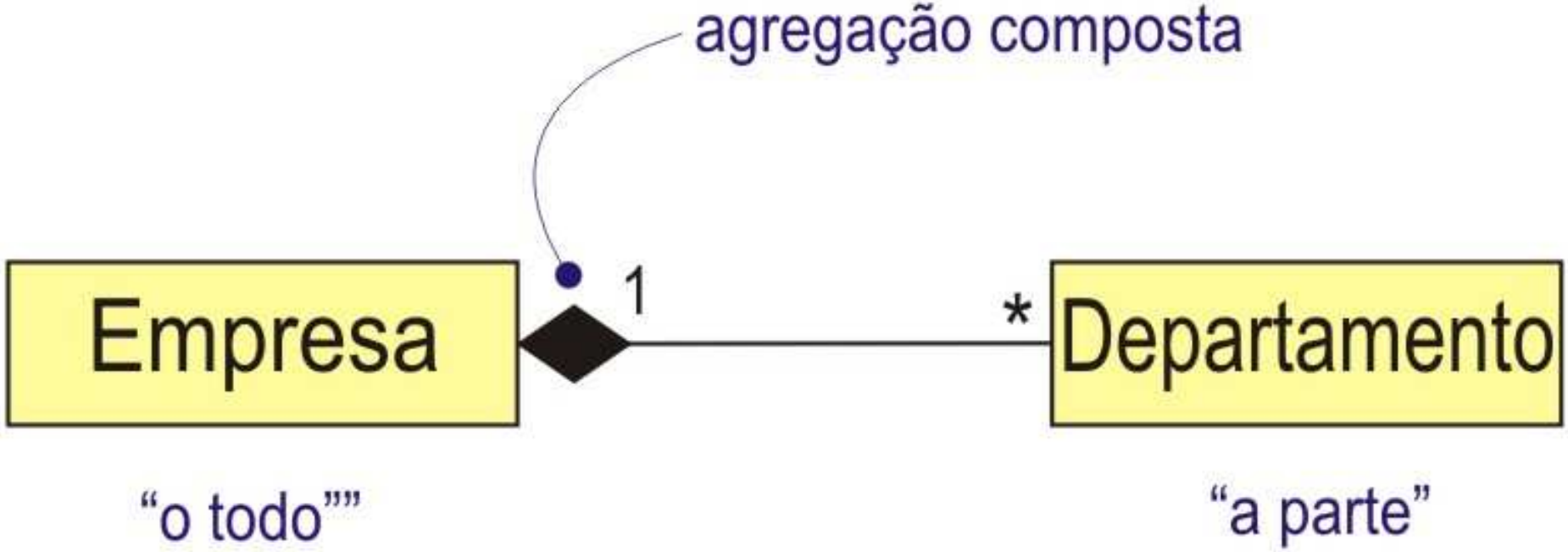
Relacionamentos - Agregação



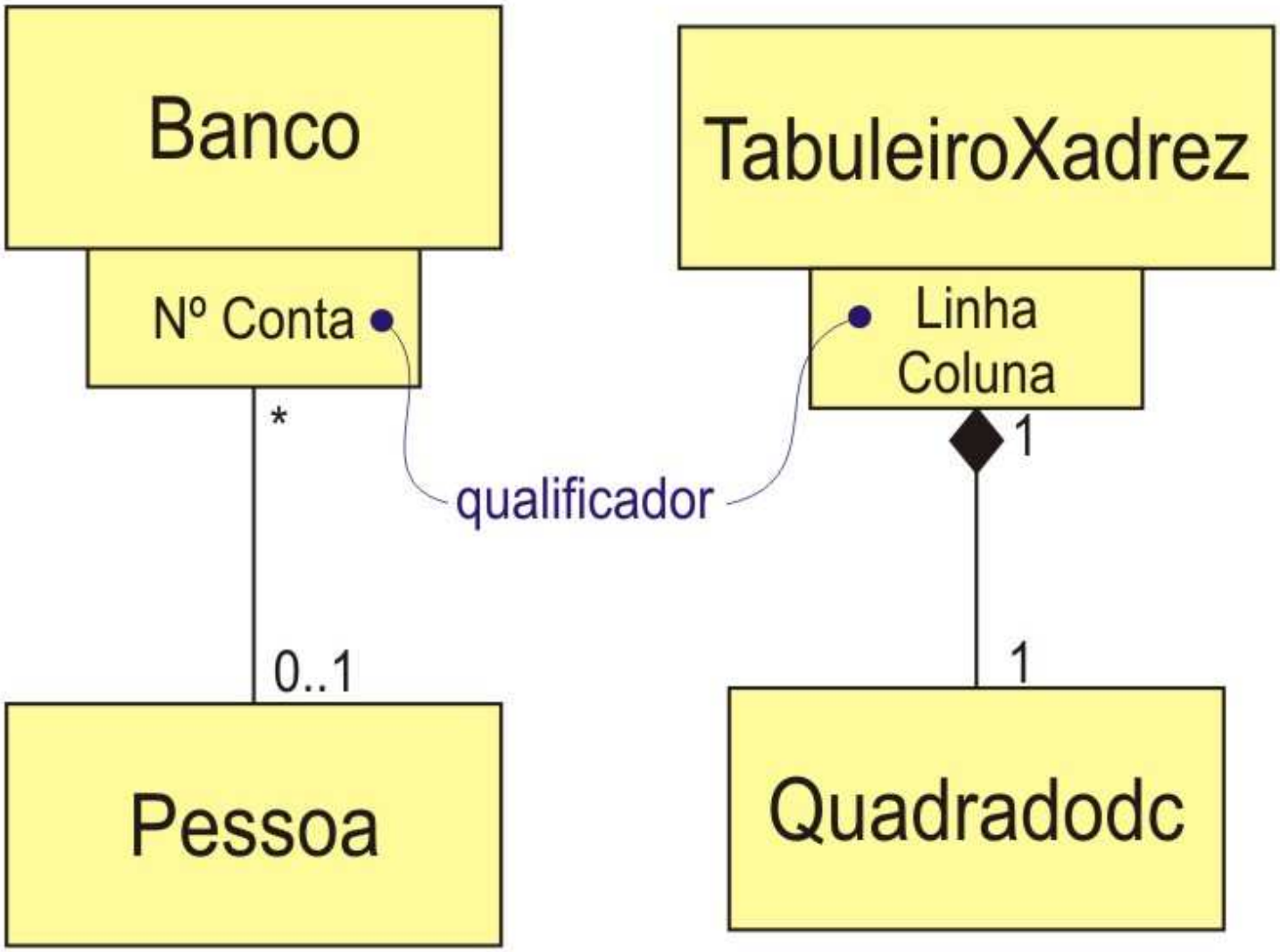
Agregação (composta)

- A composição, ou agregação composta é uma variante à agregação simples, em que é adicionada a seguinte semântica:
  - (1) forte pertença do “todo” em relação à “parte”, e
  - (2) tempo de vida delimitado (as “partes” não podem existir sem o “todo”).
- Adicionalmente, o “todo” é responsável pela disposição das suas “partes”, ou seja, “o todo” é responsável pela criação e destruição das suas “partes”.
- A informação de agregação composta é representada por um losango cheio colocado junto à classe que representa o elemento agregador ou “o todo”.

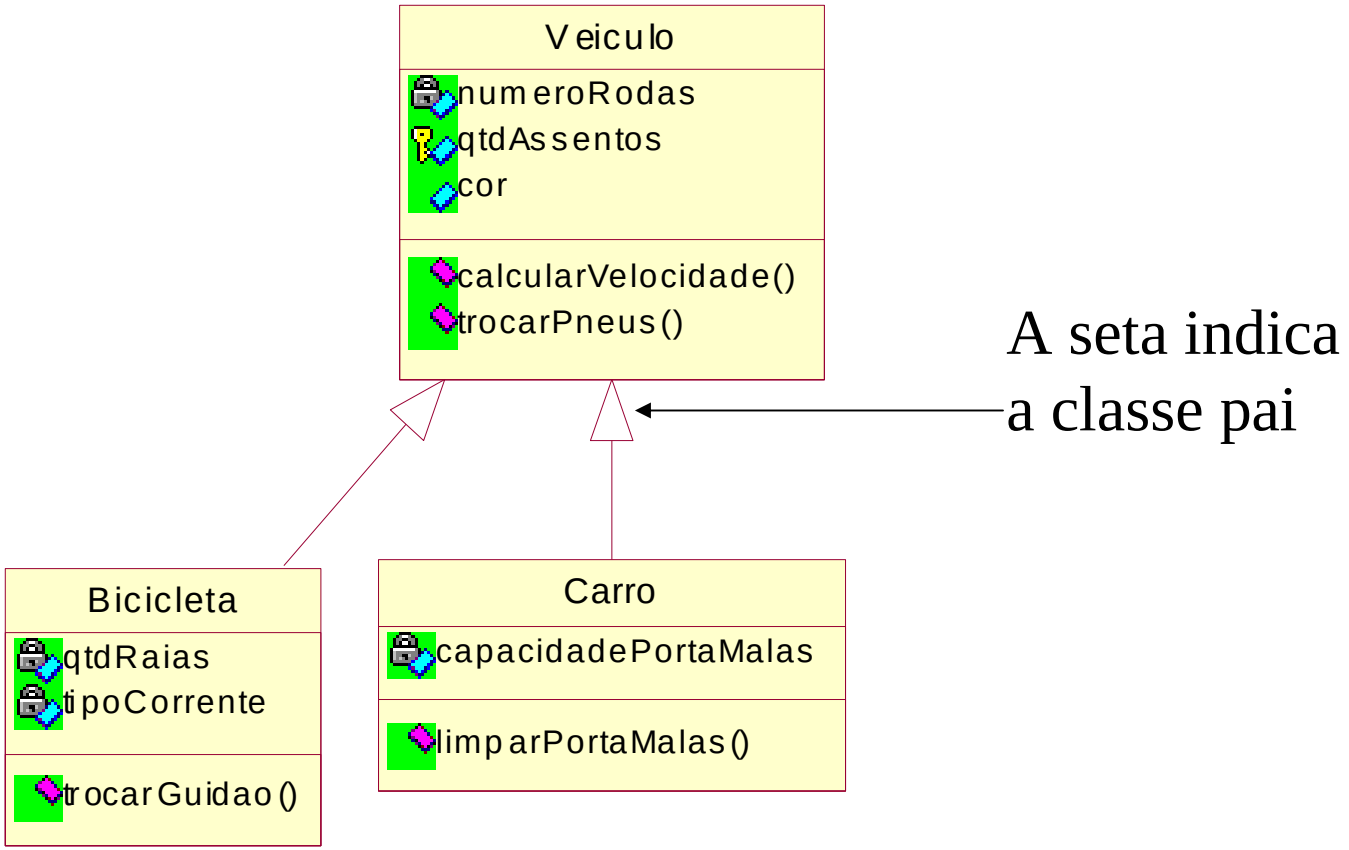
Agregação composta



Agregação composta



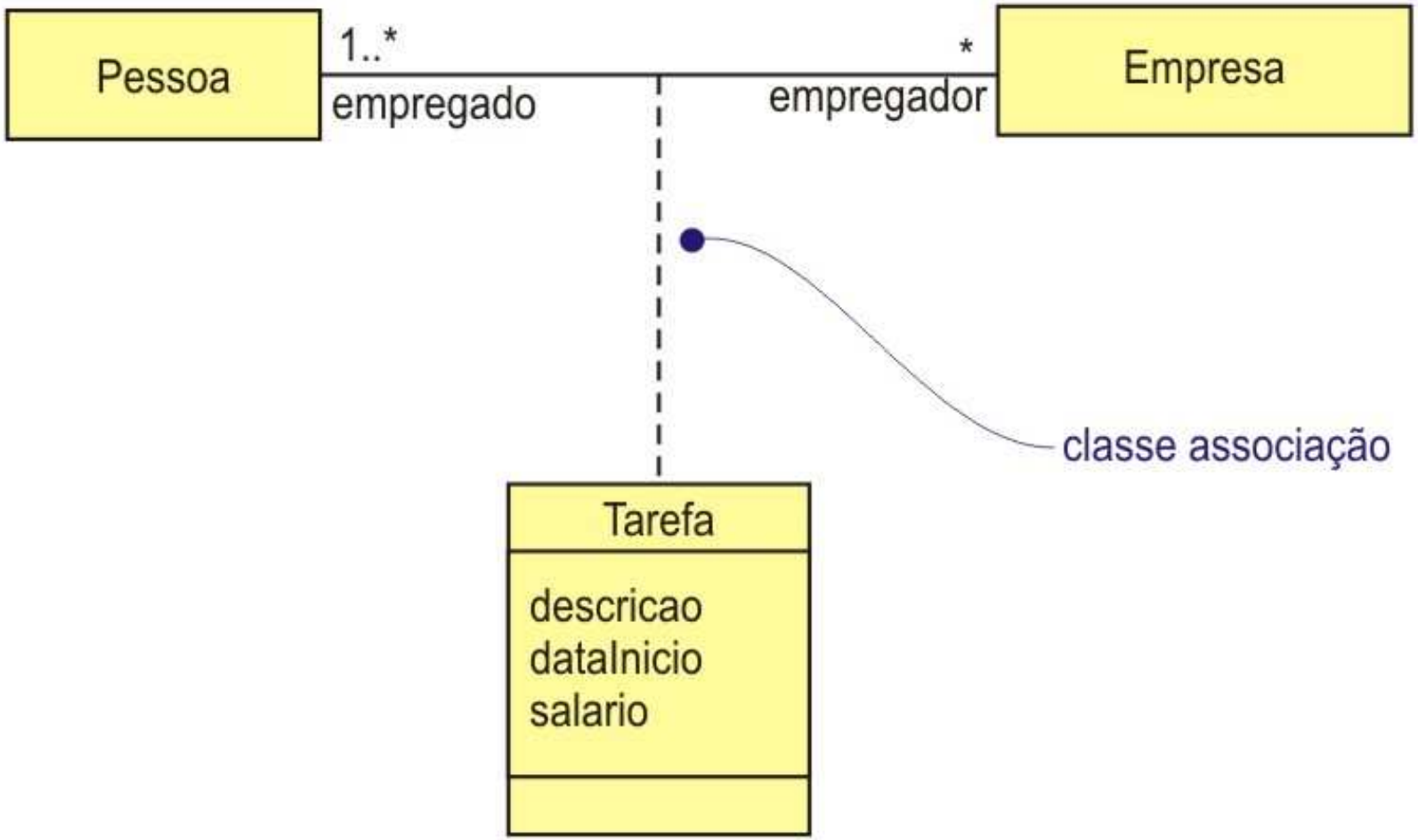
Relacionamentos - herança



Classe de Associação

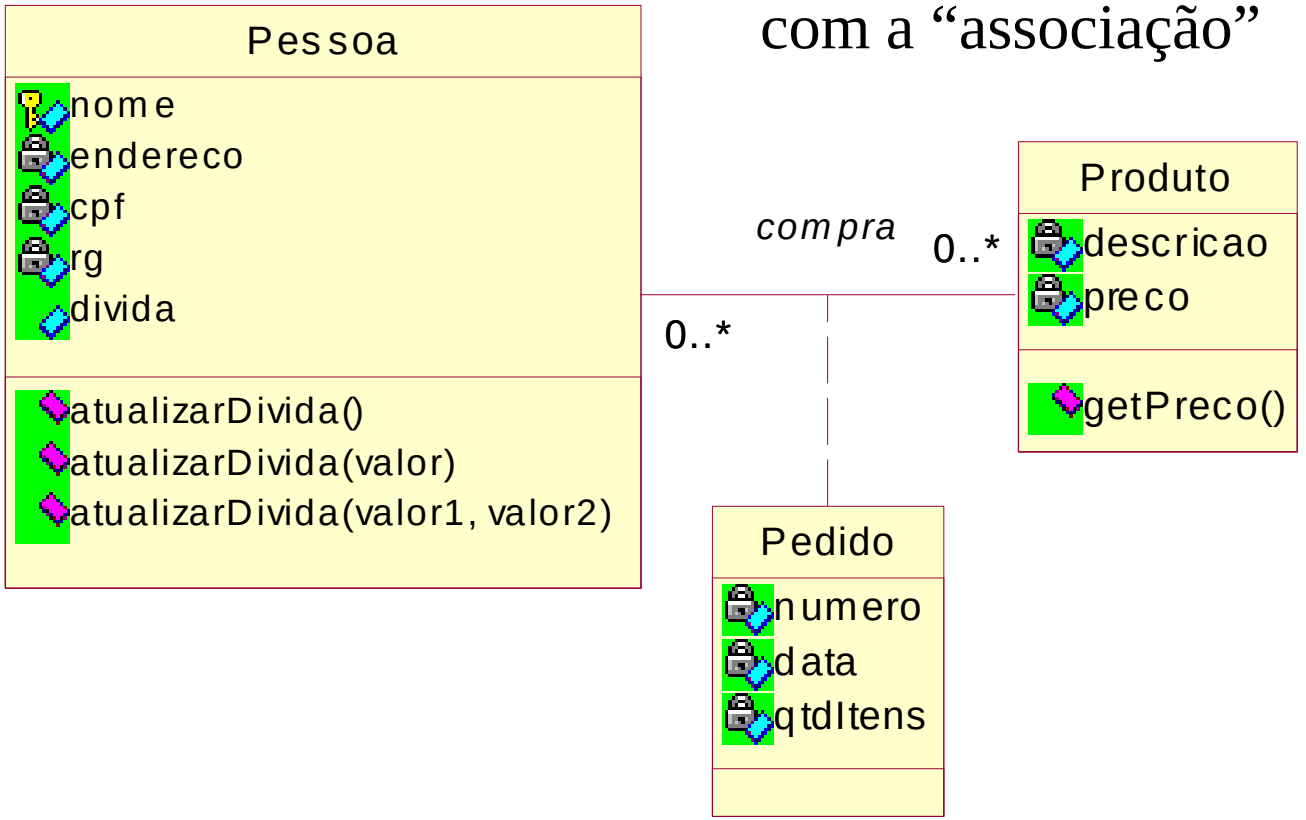
- Numa relação de associação entre classes, a associação pode também ter os seus próprios atributos (e eventualmente operações), devendo ser, por conseguinte, modelada também como uma classe.
- Este tipo de classes designa-se por classe-associação

Classe de Associação

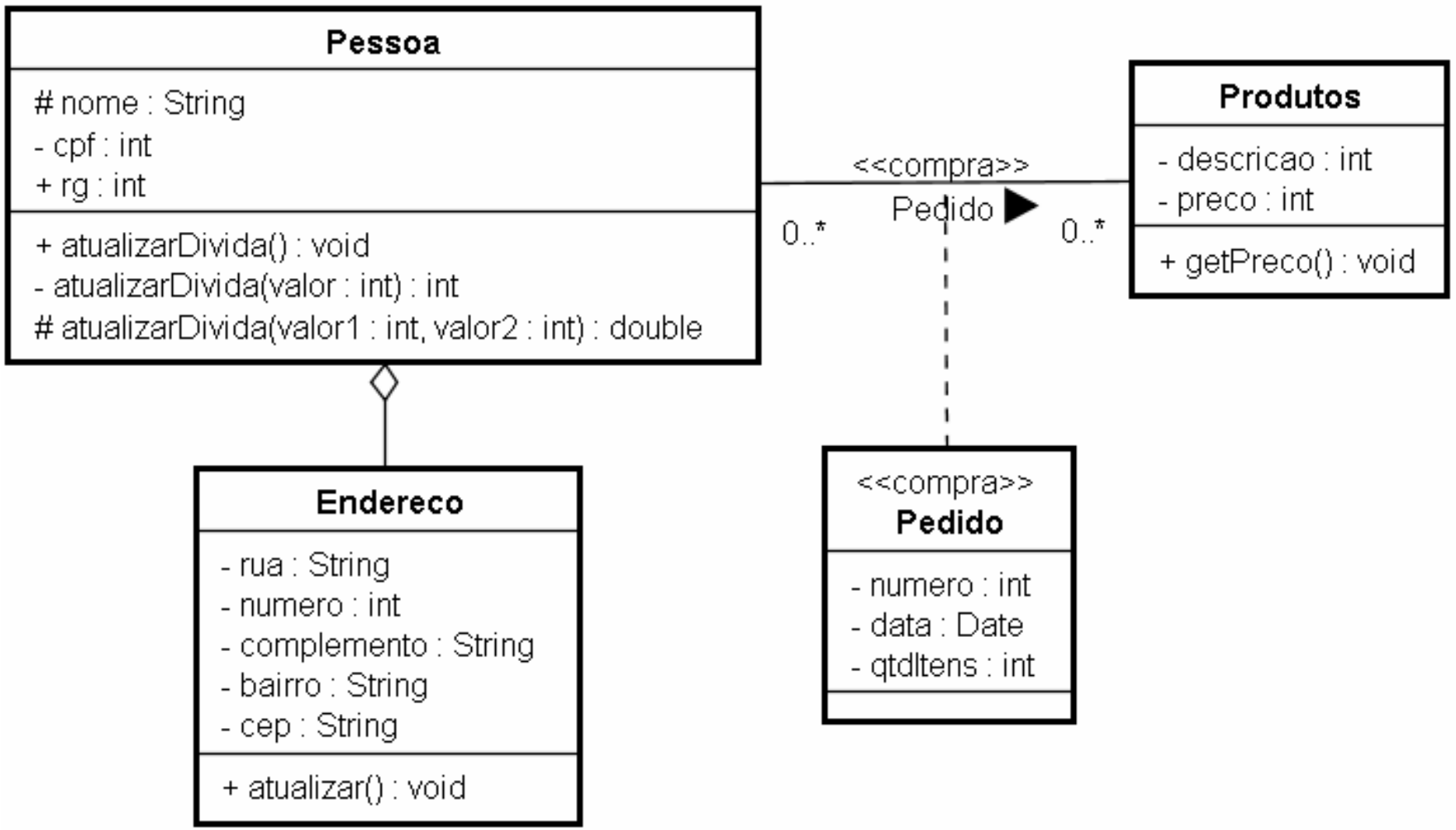


Classe de Associação

Relacionamento de uma classe com a “associação”



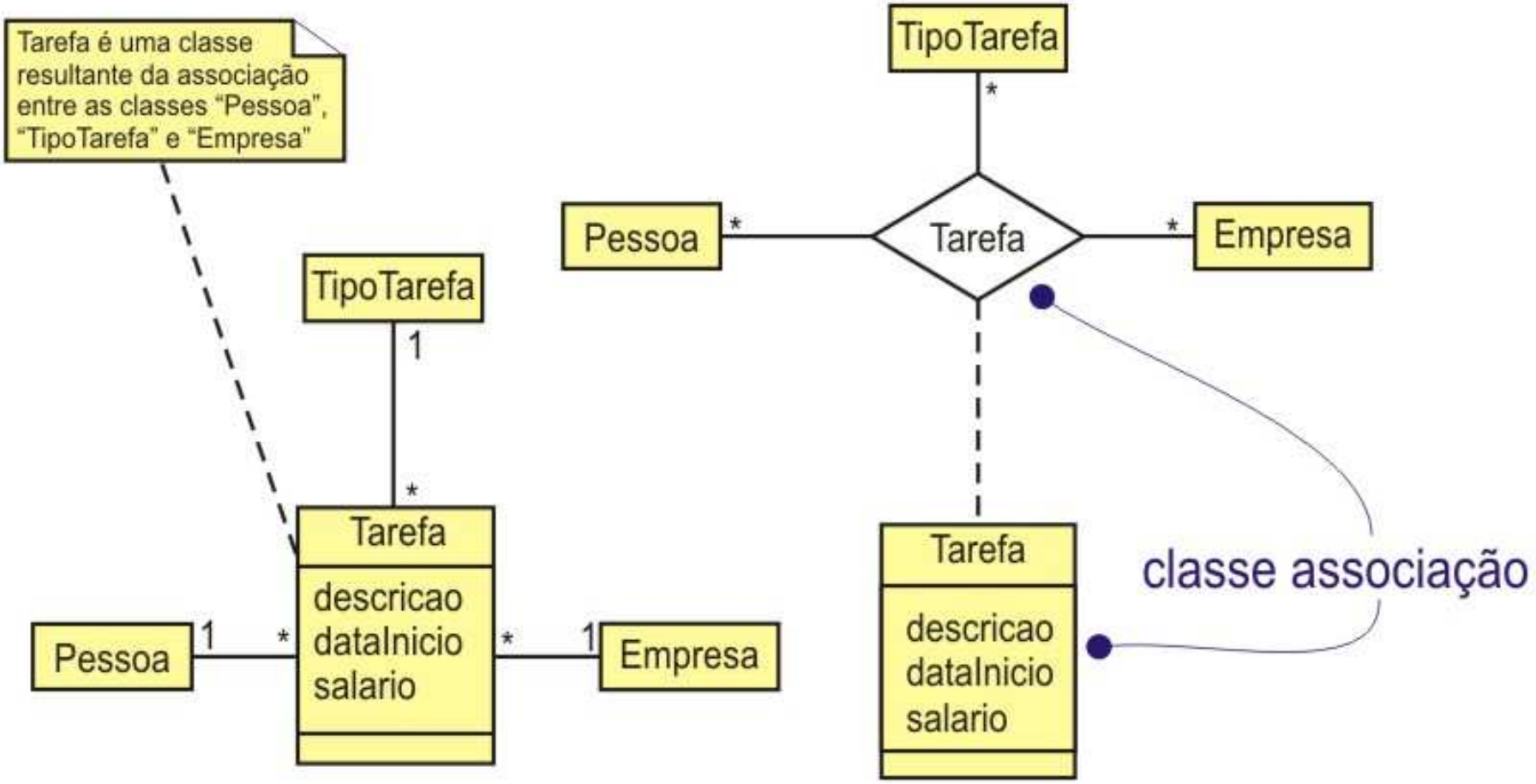
Classe de Associação



Associações N - Árias (N 3)

- Associações N-ária, com “n” maior ou igual a 3, são pouco comuns na modelagem de classes. Contudo, há situações em que a aplicação deste tipo de associações é vantajosa em termos da clareza do modelo.
- Nestas circunstâncias, a associação é representada por um losango com linhas para todas as suas classes participantes

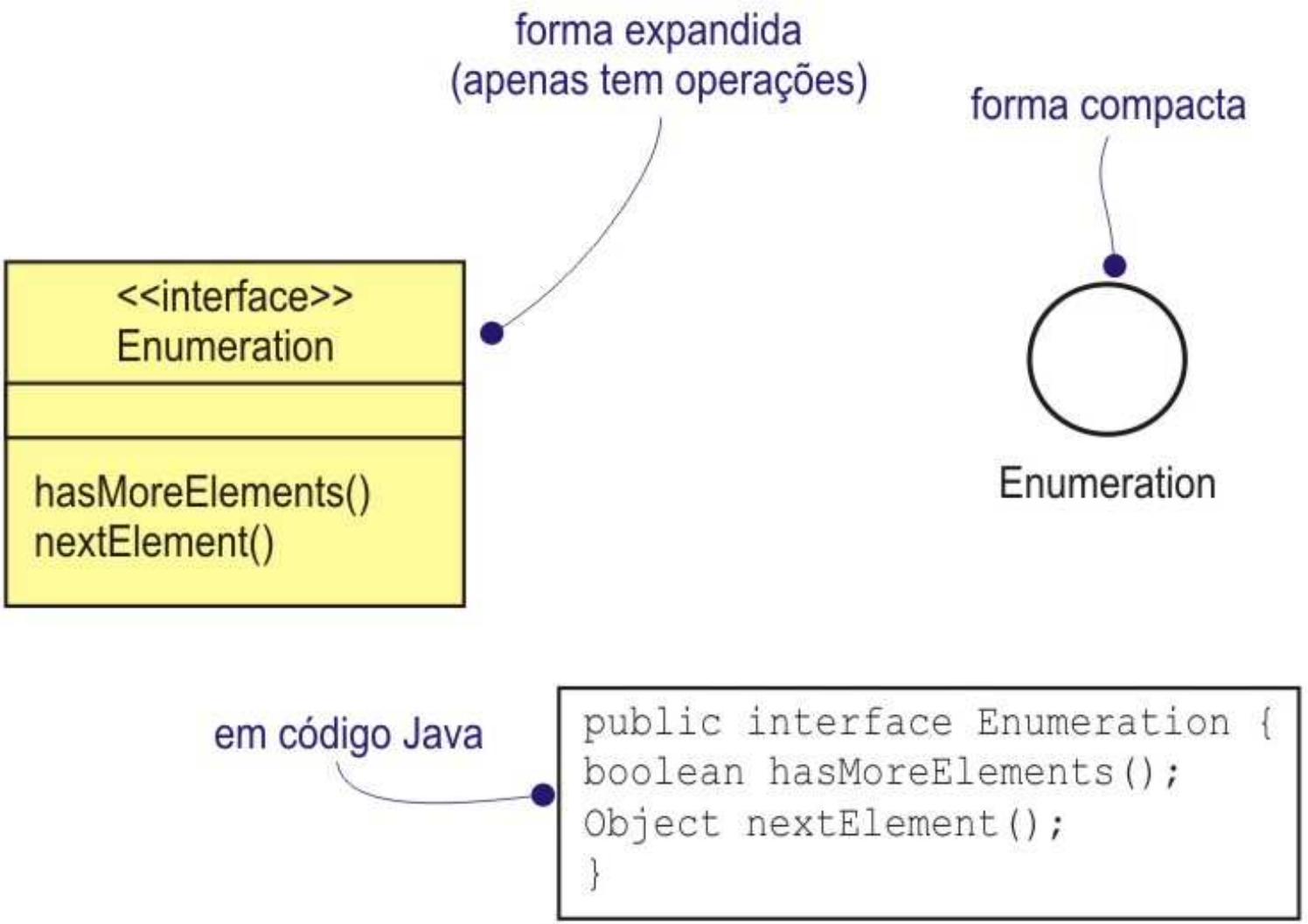
Associações N - Árias (N 3)



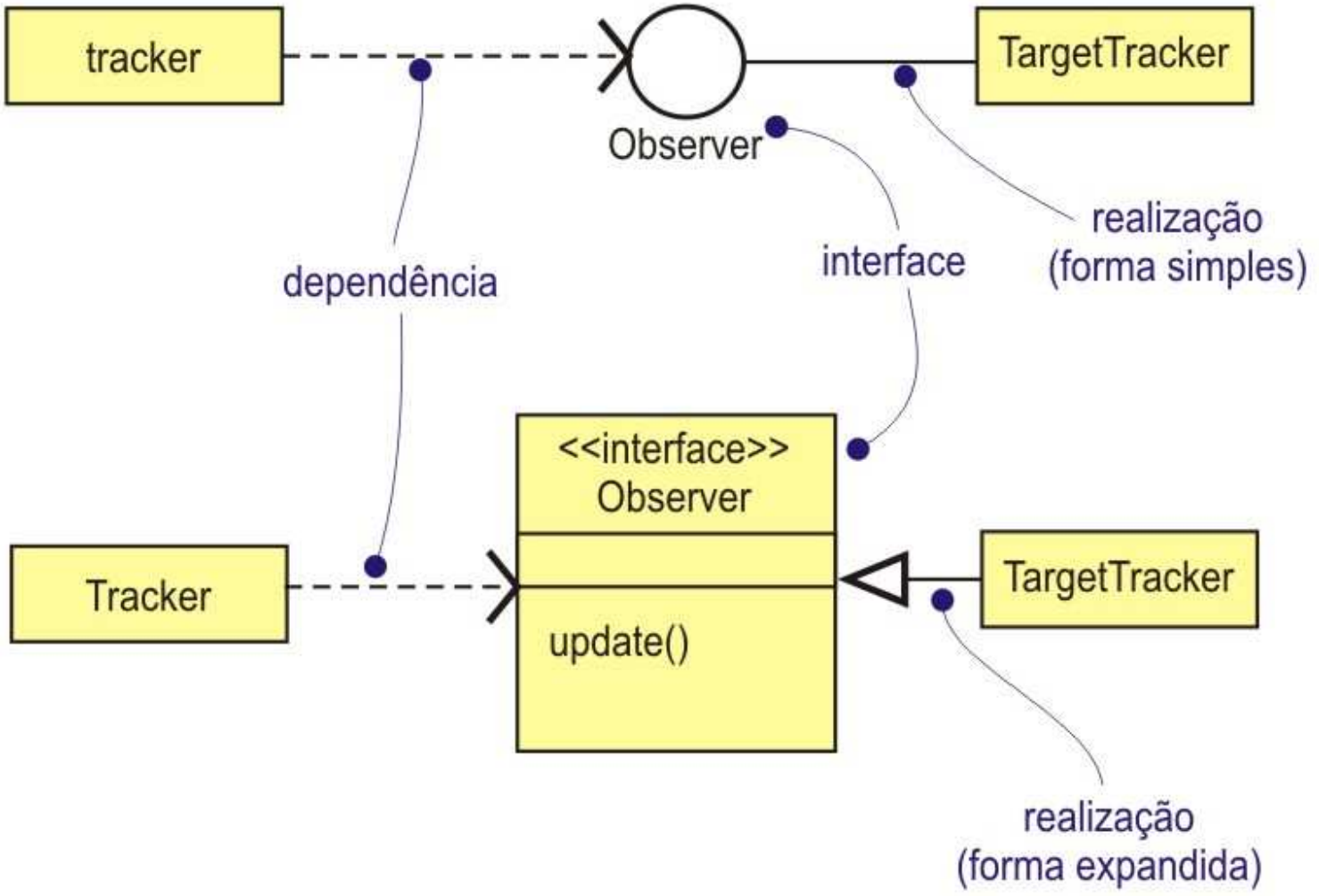
Interfaces

- Uma interface define um comportamento obrigatório que alguma classe deve possuir.
- As interfaces permitem conhecer um determinado elemento, escondendo os seus detalhes internos, por exemplo: os detalhes de implementação.
- Uma interface é realizada (ou implementada) por uma ou mais classes, as quais prometem implementar todos os métodos nela especificados.

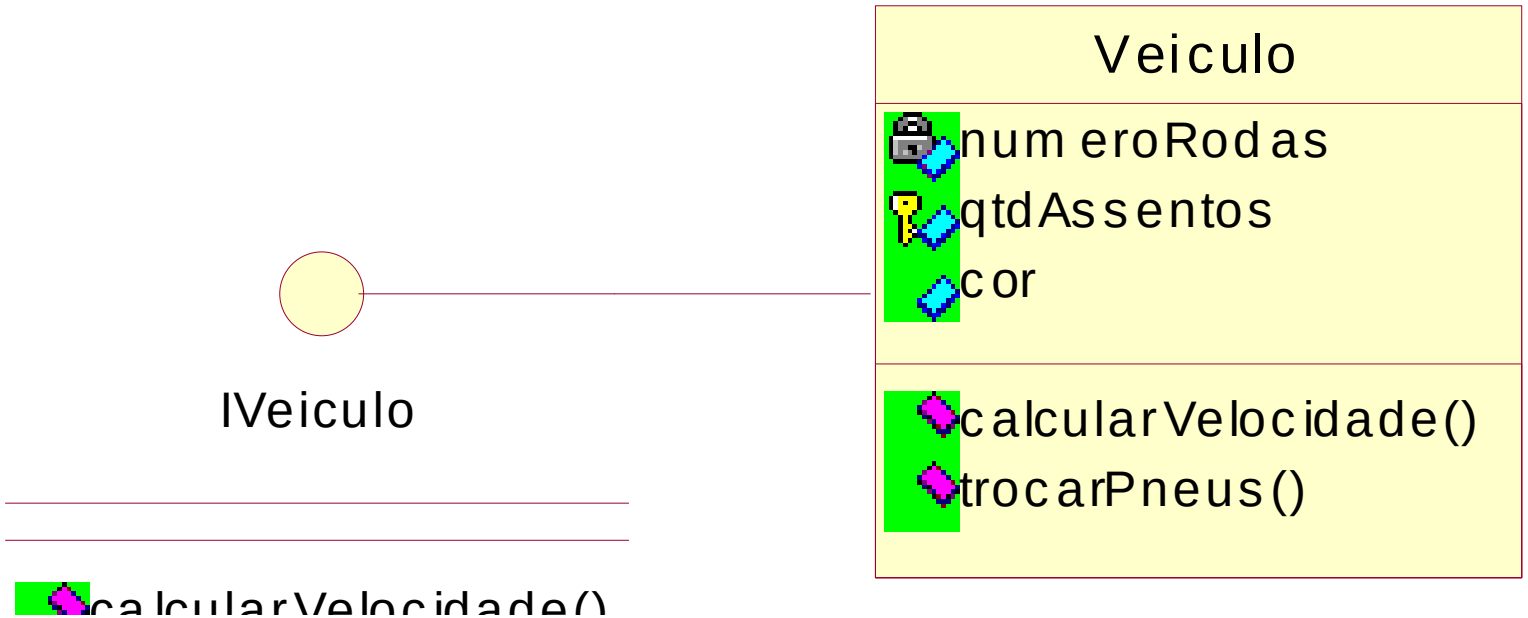
Como representar a Interface



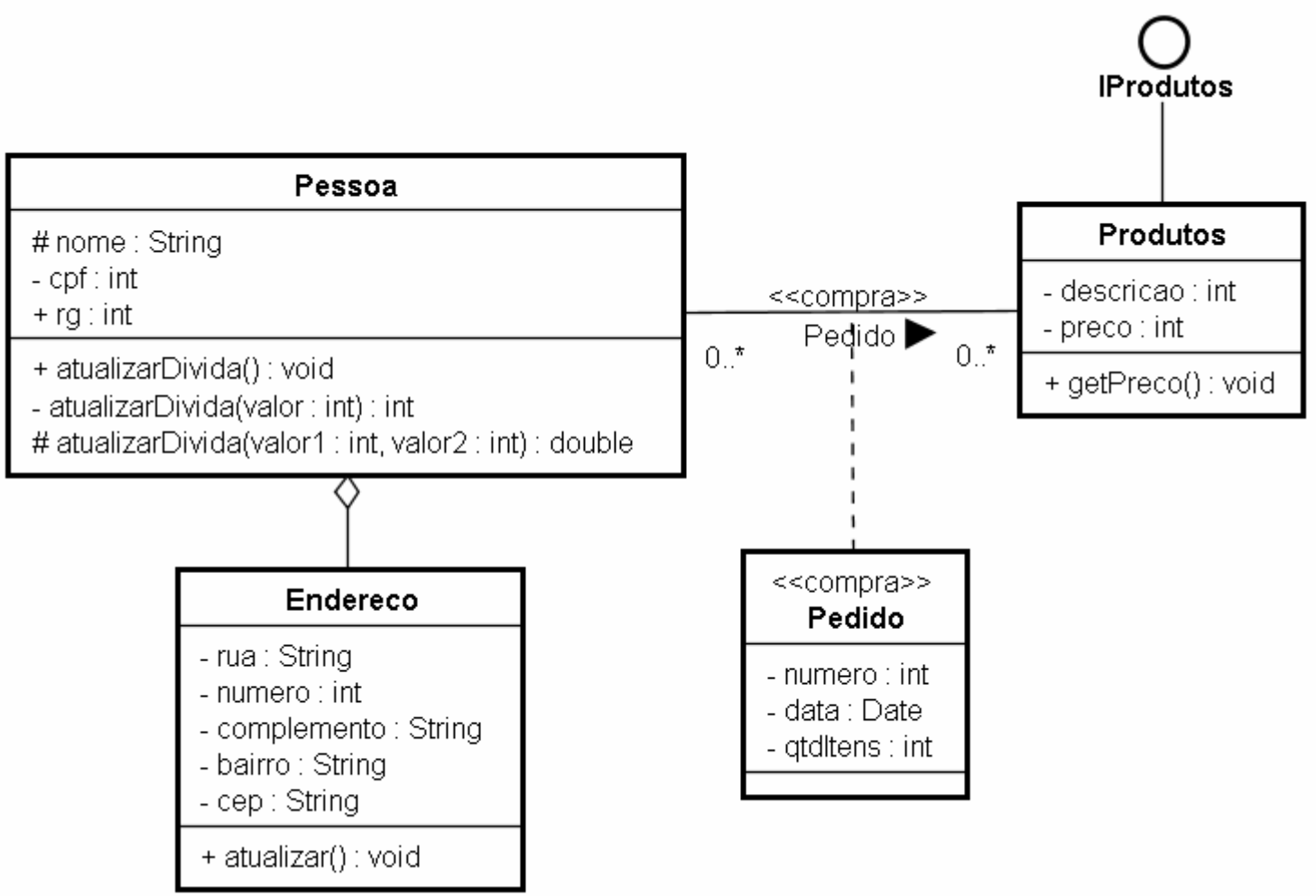
Interfaces
Formas Expandidas e Compactas



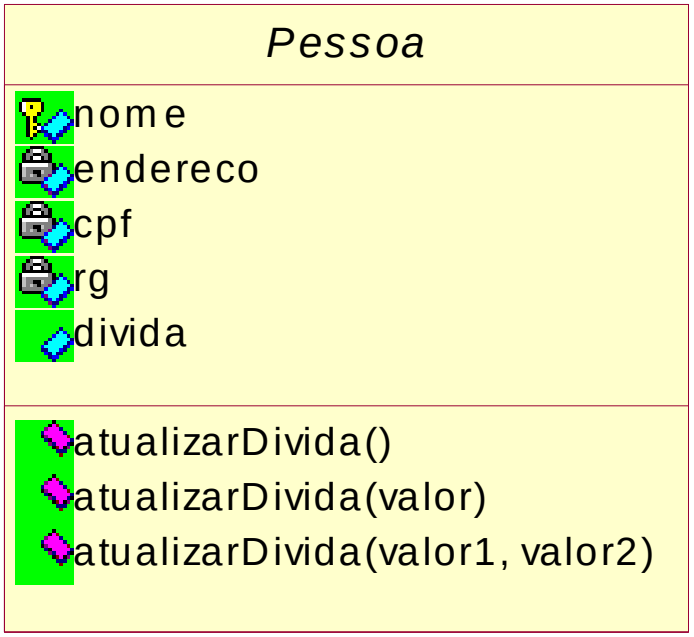
Interfaces
Exemplo (Rational Rose)



Interfaces
Exemplo (astah)

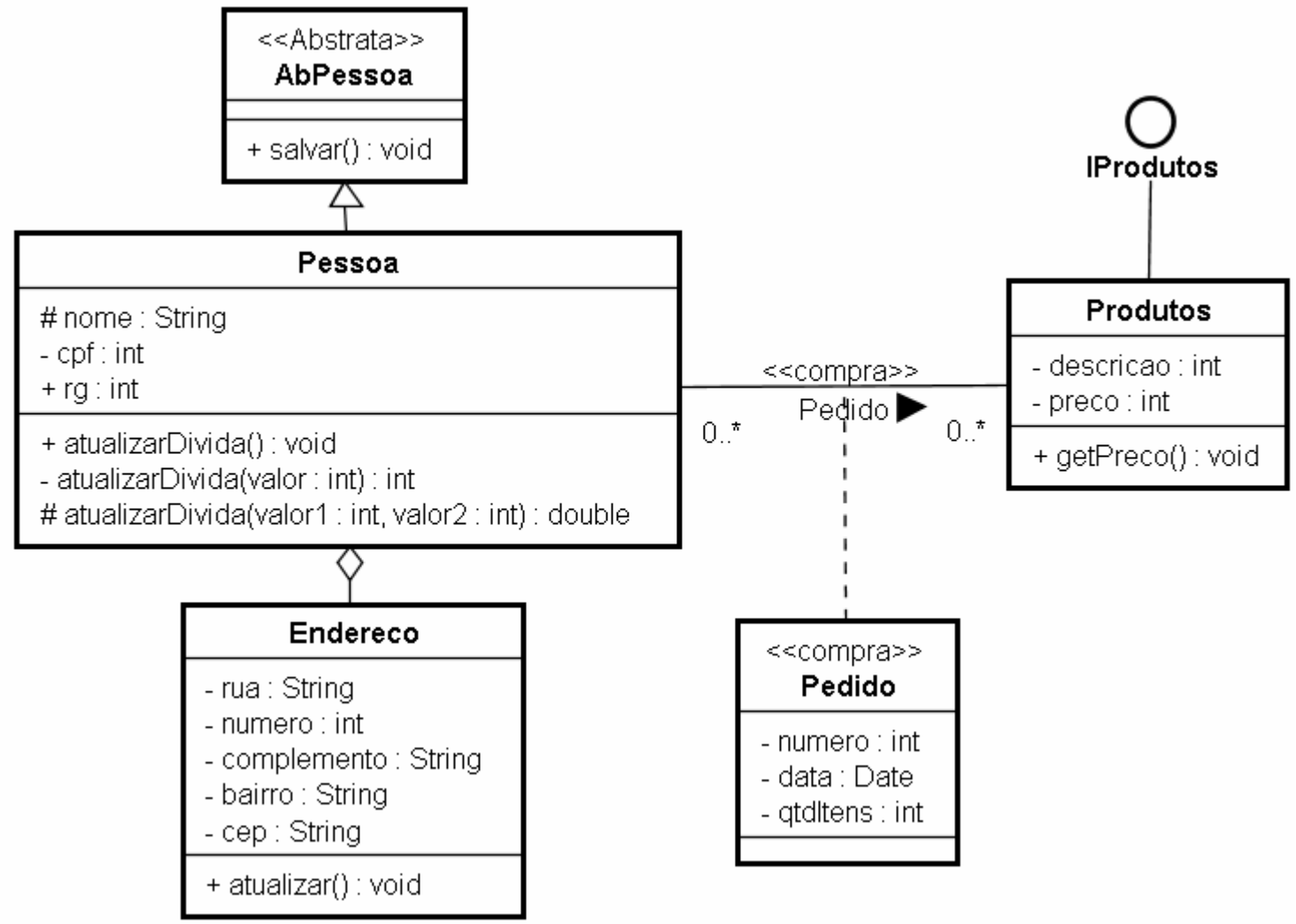


Classes Abstratas

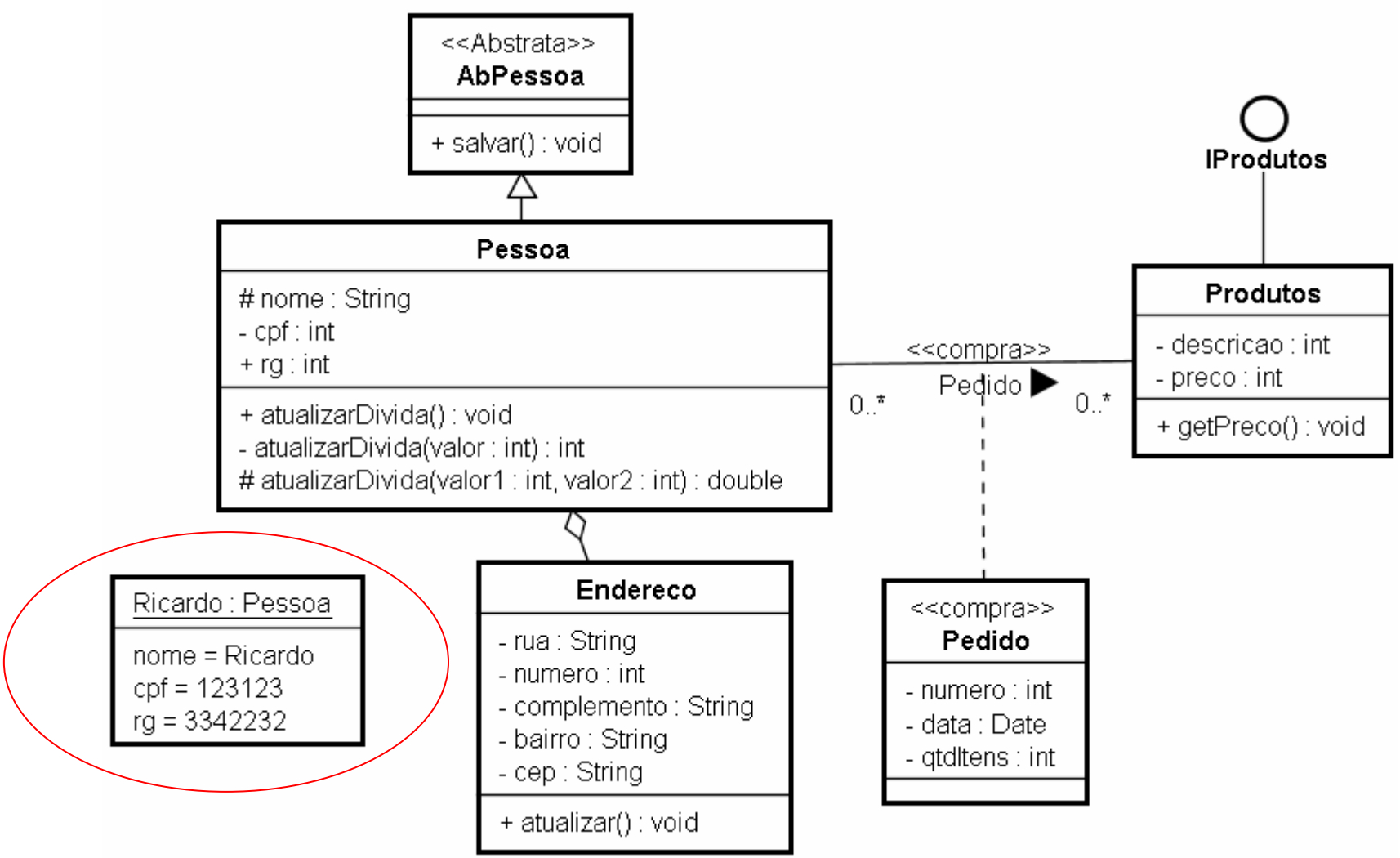


O nome em itálico indica que uma classe é abstrata
Uma classe abstrata não pode ser instanciada, sendo assim, sua finalidade fica restrita a definir comportamentos para que outras classes possam herdar.

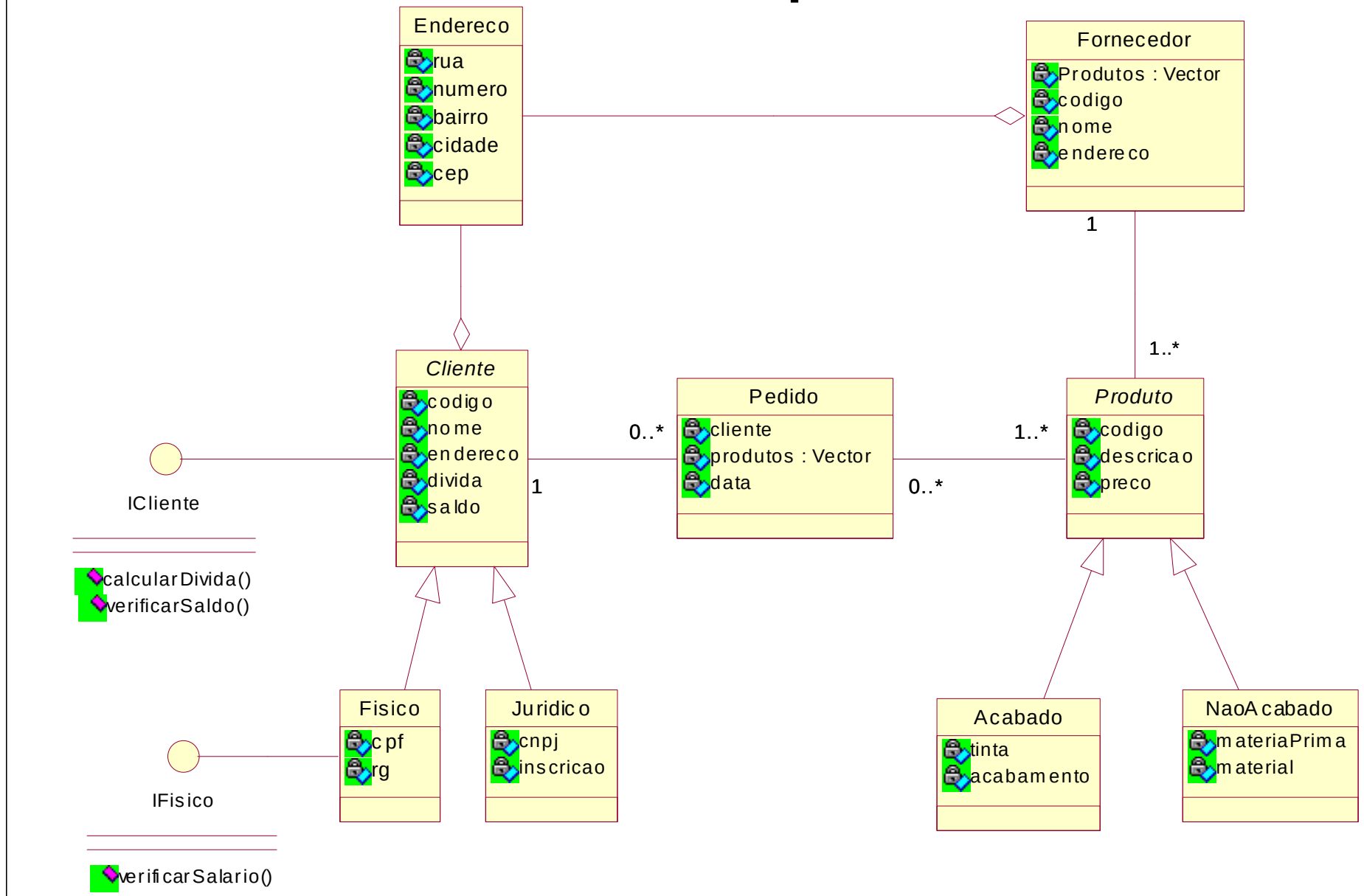
Um Estereotipo pode ser inserido para indicar uma Classe Abstrata



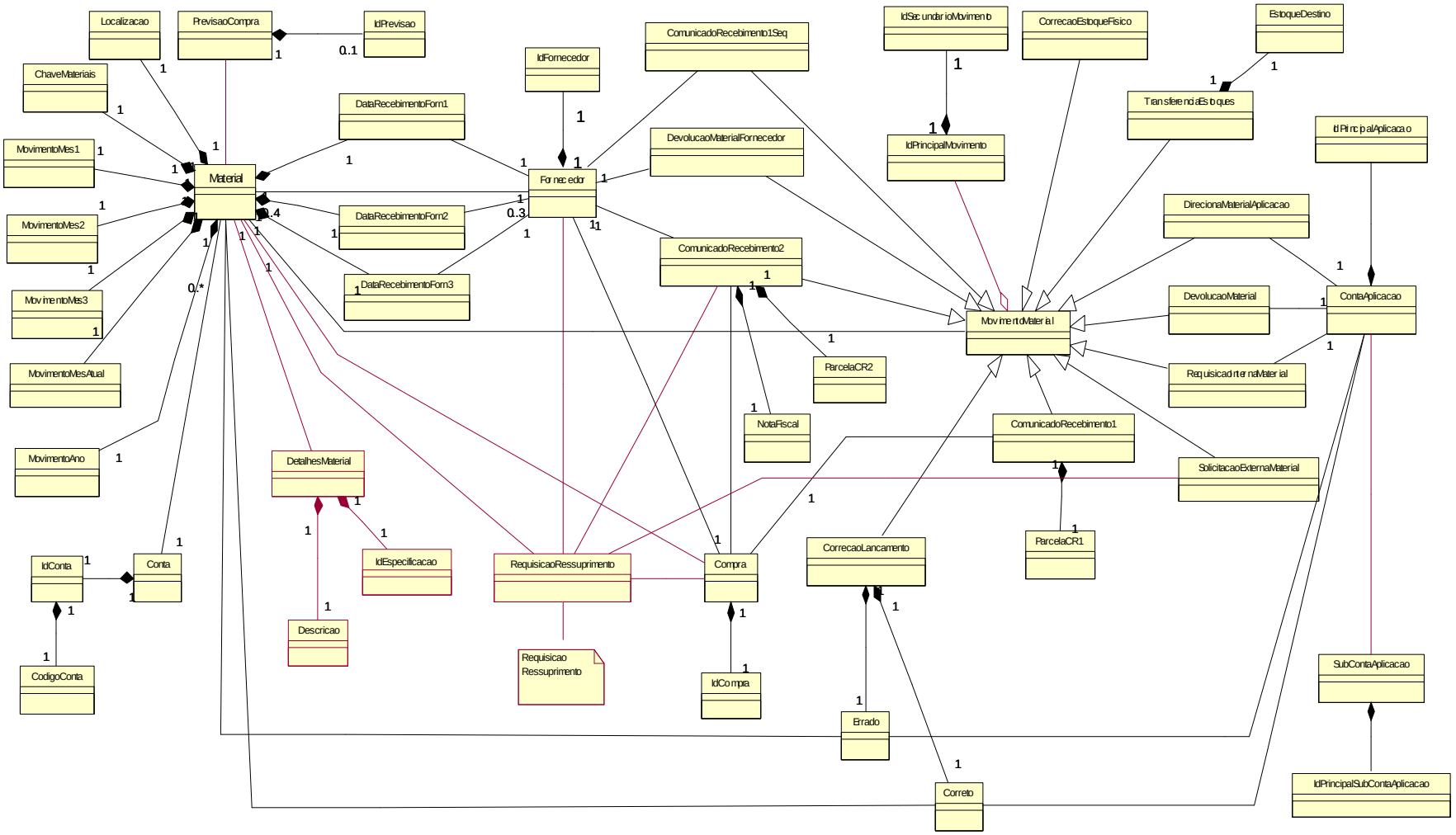
Objetos instanciados



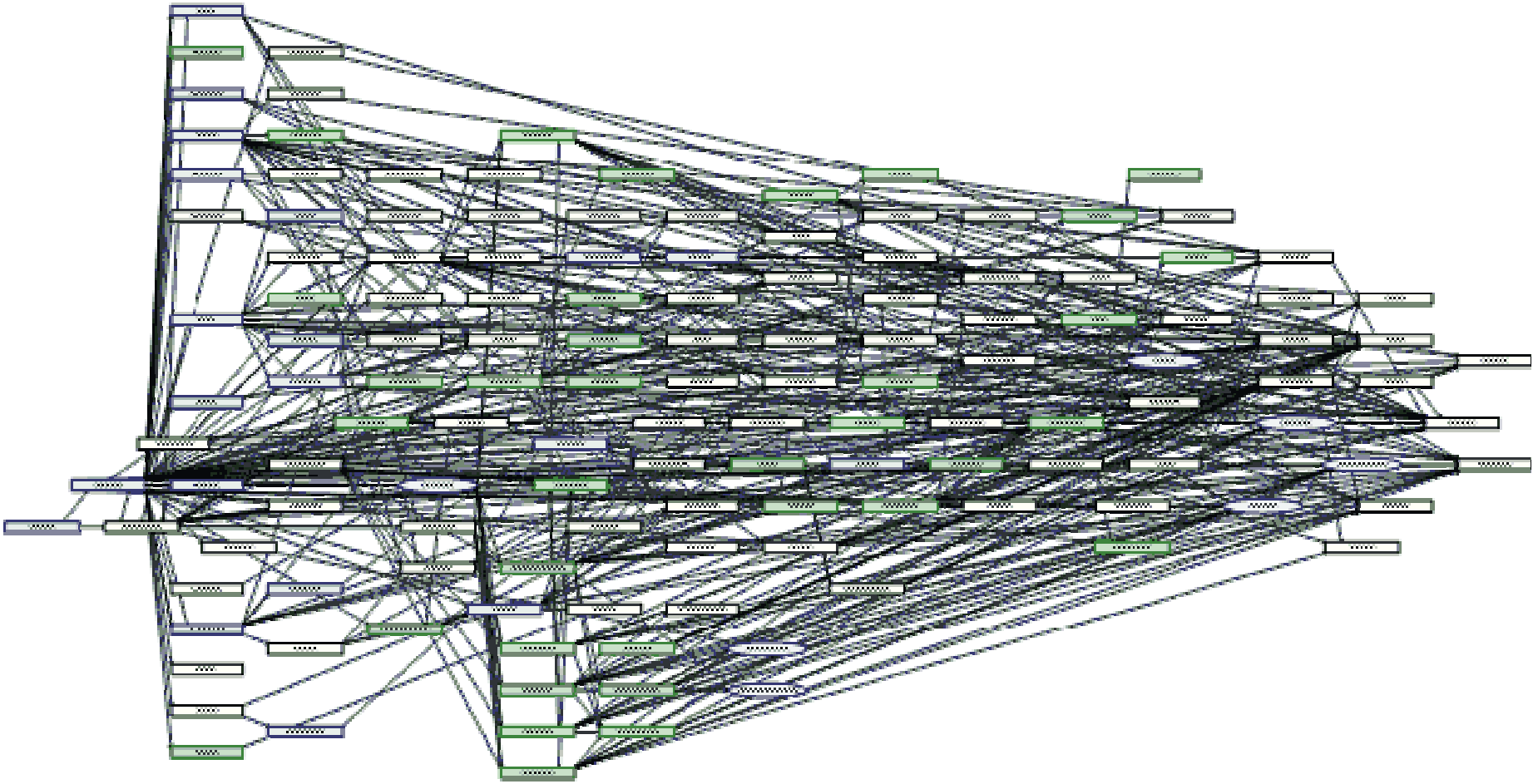
Exemplo


Endereco
rua
numero
bairro
cidade
cep
Fornecedor
Produtos : Vector
codigo
nome
endereco
1
1..*
Cliente
codigo
nome
endereco
divida
saldo
1
0..*
Pedido
cliente
produtos : Vector
data
0..*
1..*
Produto
codigo
descricao
preco
ICliente
calcularDivida()
verificarSaldo()
Fisico
cpf
rg
Juridico
cnpj
inscricao
IFisico
verificarSalario()
Acabado
tinta
acabamento
NaoAcabado
materiaPrima
material

Exemplo


Localizacao
PrevisaoCompra
IdPrevisao
ChaveMateriais
IdFornecedor
ComunicadoRecebimento1Seq
IdSecundarioMovimento
CorrecaoEstoqueFisico
EstoqueDestino
TransferenciaEstoques
IdPrincipalAplicacao
DataRecebimentoForn1
DevolucaoMaterialFornecedor
IdPrincipalMovimento
MovimentoMes1
Material
Fornecedor
DirecionaMaterialAplicacao
MovimentoMes2
DataRecebimentoForn2
ComunicadoRecebimento2
DevolucaoMaterial
ContaAplicacao
DataRecebimentoForn3
MovimentoMes3
MovimentoMaterial
ParcelaCR2
RequisicaoInternaMaterial
MovimentoMesAtual
NotaFiscal
ComunicadoRecebimento1
MovimentoAno
DetalhesMaterial
SolicitacaoExternaMaterial
IdConta
Conta
IdEspecificacao
RequisicaoRessuprimento
Compra
CorrecaoLancamento
ParcelaCR1
Descricao
SubContaAplicacao
CodigoConta
Requisicao Ressuprimento
IdCompra
Errado
IdPrincipalSubContaAplicacao
Correto

O Caos na UML com os diagramas de Classes



Conceitos da OO Usando UML

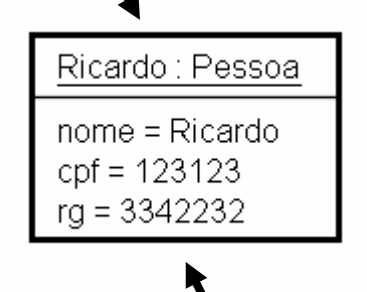
Classes /Objetos

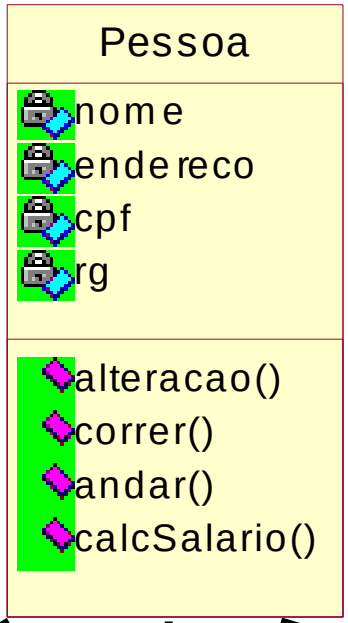
- Uma classe pode ser vista como uma fábrica de objetos similares
- Define os dados e comportamento que todos os seus objetos terão
- Cada objeto de uma classe diferencia-se do outro por meio do valor de seus atributos

Classes e Objetos

Classe Pessoa

Em UML





Cada objeto possui seu próprio nome, endereco, cpf e rg

Ricardo

João

Pedro

Objetos - instâncias

Encapsulamento

- Com o advento da Internet e a exposição de sistemas nessa grande rede, a segurança tornou-se algo fundamental.
- Esse conceito está relacionado à proteger os dados da classe

Encapsulamento

• Cada componente do programa deve conter uma única funcionalidade

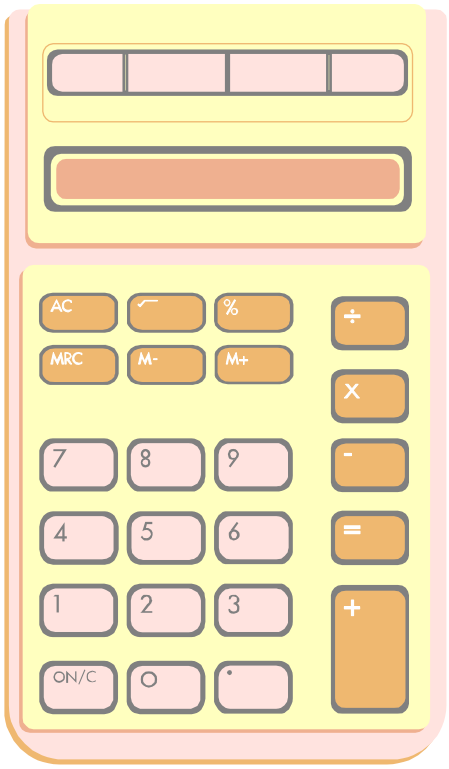
• A interface do componente deve expor o mínimo possível sobre o funcionamento interno do componente

• Usuários de uma biblioteca necessitam saber apenas sua interface (assinatura dos métodos) para utilizá-la. Alterações no algoritmo não afetam os usuários que a utilizam.

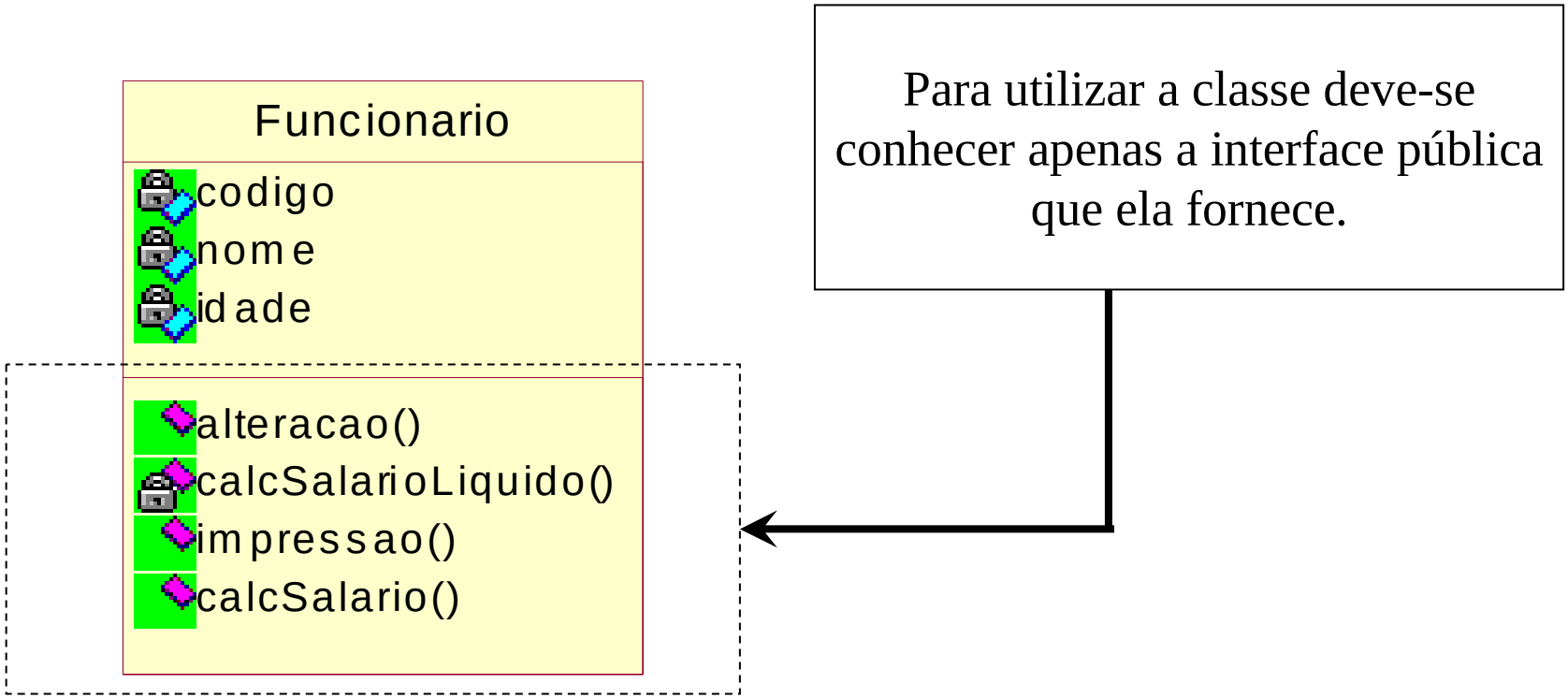
• Combina-se atributos e serviços que agem sobre esses atributos

Encapsulamento – ex.

- Agrupa seus registradores internos e disponibiliza ao usuário apenas as funções necessárias.
- Não se tem acesso ao seu Interior.
- é uma caixa preta



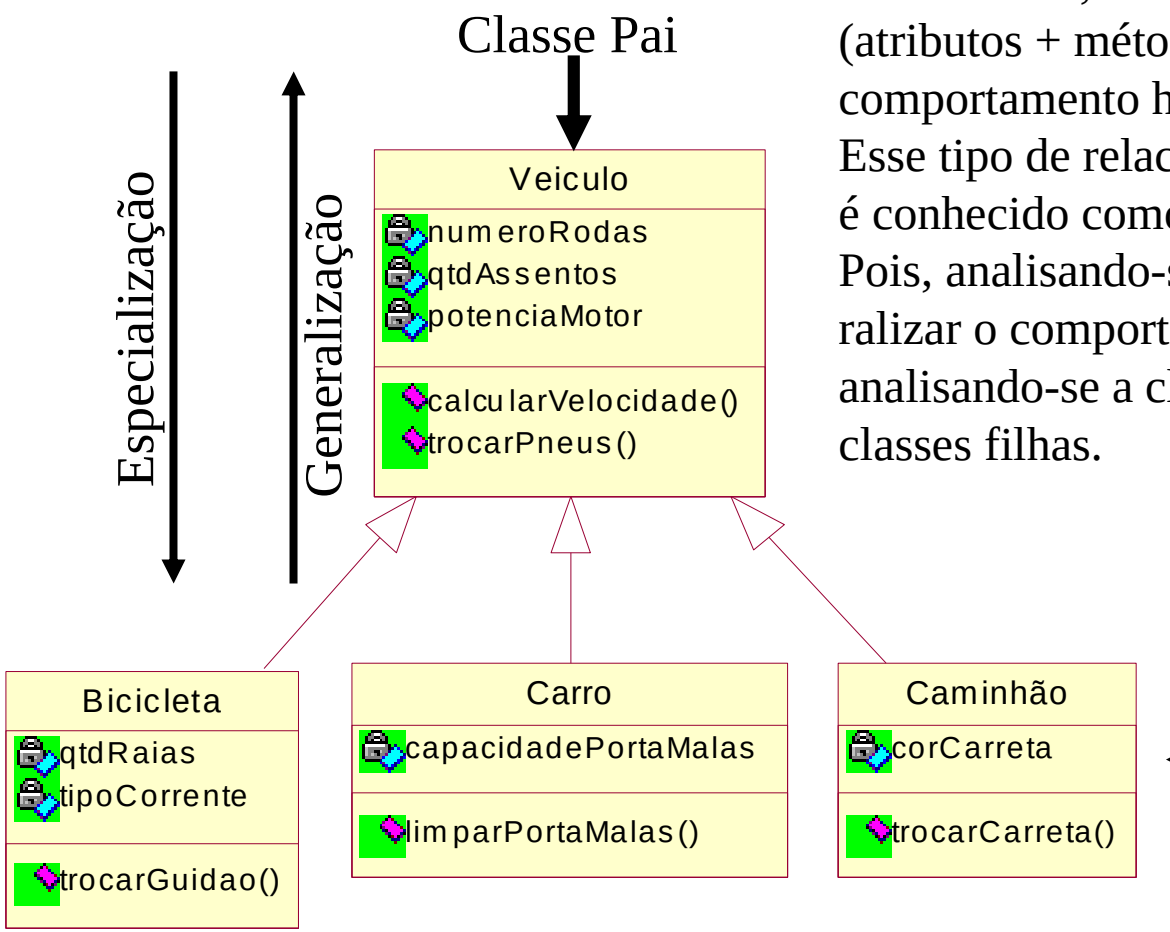
Encapsulamento – ex.



Herança

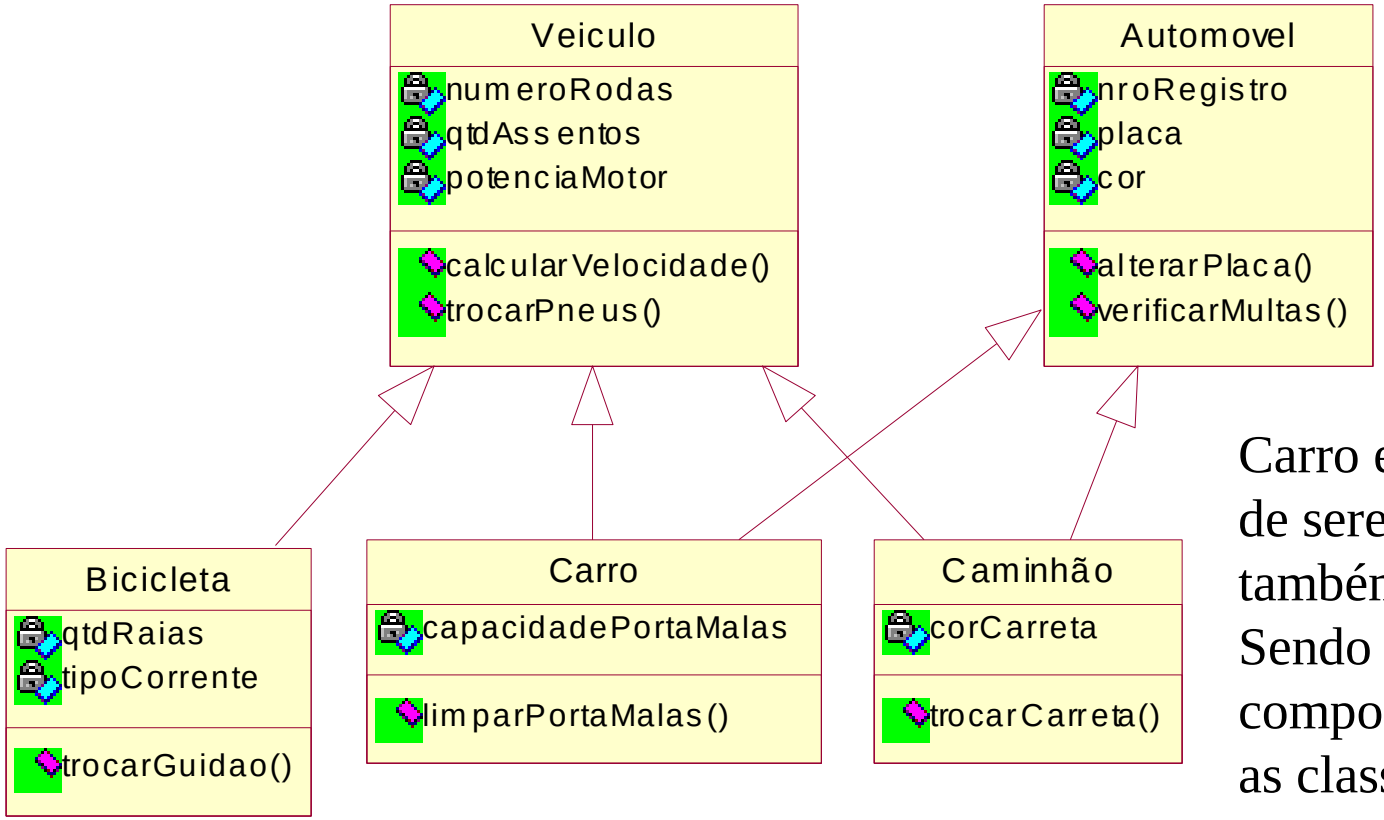
- Um módulo (classe) pode ser quase o que queremos...
- Simplifica a definição de classes que são quase iguais às que já foram definidas
- Permite a reutilização de definições comuns
- Geralmente identifica-se uma herança quando diz-se a palavra “é um”
- Por exemplo:
  - Bicicleta é um veículo
  - Carro é um veículo
  - Caminhão é um veículo

Herança



Bicicleta, Carro e Caminhão são veículos. Sendo assim, eles possuem seu comportamento (atributos + métodos) específico mais o comportamento herdado.
Esse tipo de relacionamento entre classes também é conhecido como **generalização/especialização**. Pois, analisando-se as classes filhas pode-se generalizar o comportamento na Pai. Do mesmo modo analisando-se a classe Pai pode-se especializar classes filhas.

Herança múltipla

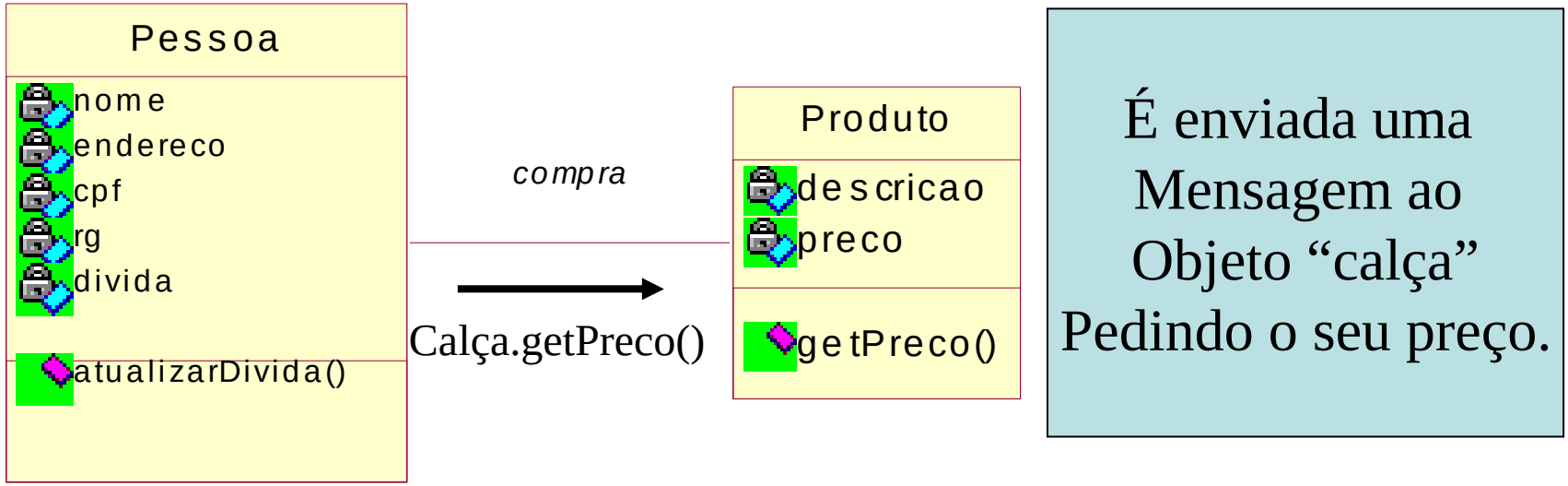


Carro e Caminhão, além de serem Veículos também são automóveis. Sendo assim, herdam o comportamento de ambas as classes

Mensagens

- Objetos se comunicam por meio de mensagens
- Um mensagem é um sinal enviado à um objeto requisitando a execução de um serviço através da execução de uma operação
- Essa operação é executada dentro do objeto que recebe a mensagem com base nos dados de seus alcance na hierarquia de classes
- Sender e receiver
- As mais conhecidas são: Agregação e Associação

Exemplo→ Mensagem

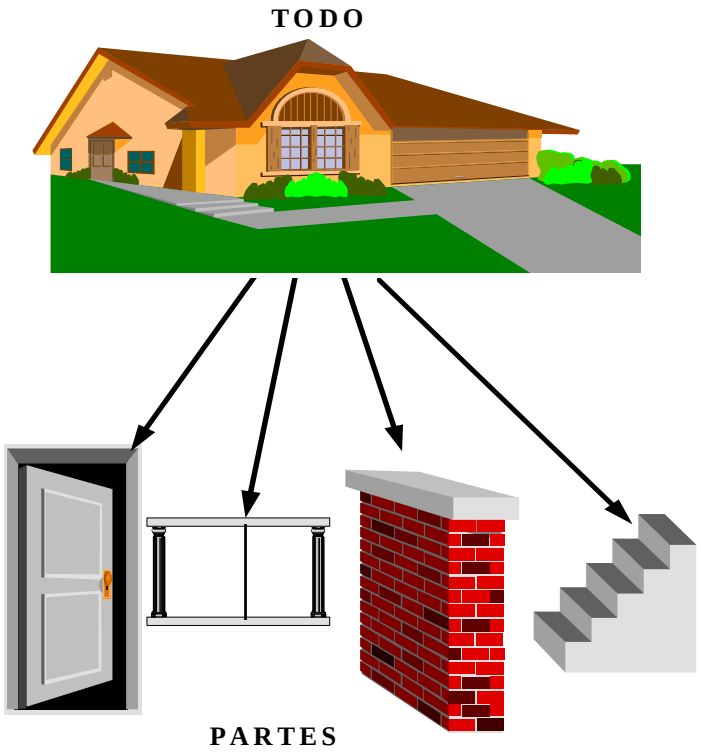


Ex. Um objeto “João” necessita atualizar seu atributo “divida”. Para isso há necessidade de saber o preço do produto que o “João” comprou. Sendo assim, o método getPreco() da classe Produto deve ser invocado (mensagem) para obter o preço do produto.

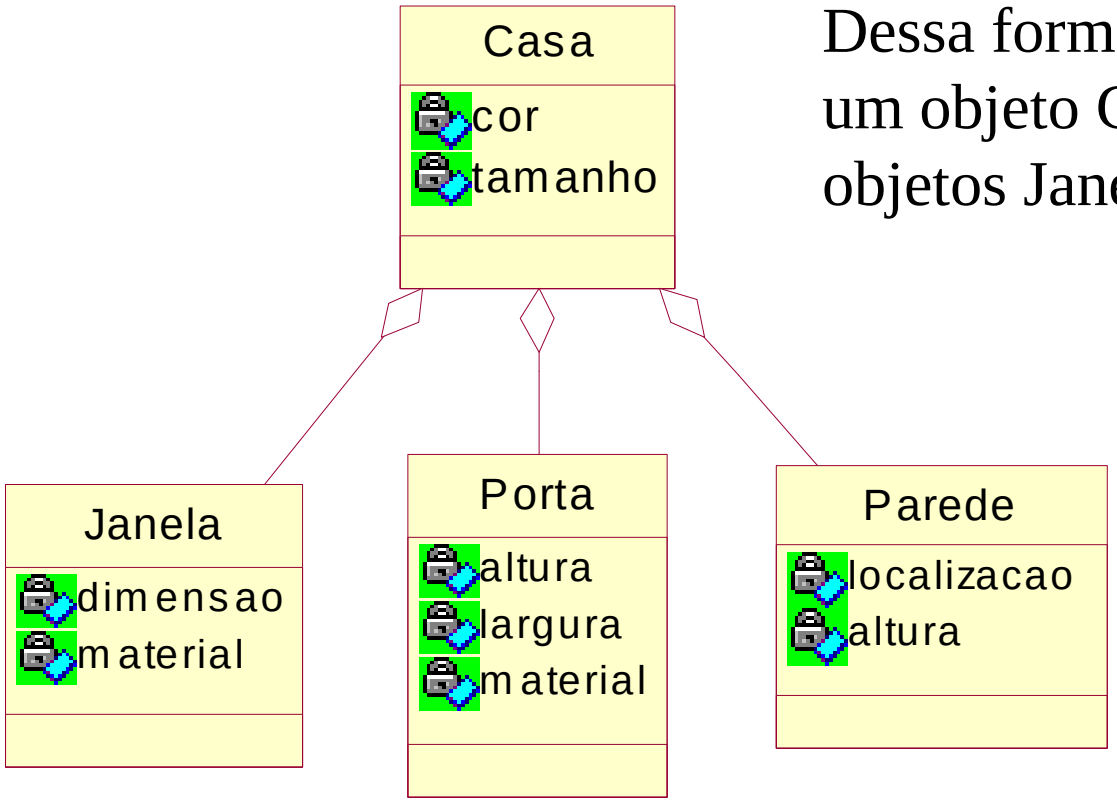
Todo-Parte (Agregação)

- Esse conceitos permite a construção de uma classe agregada a partir de outras classes componentes.
- Usa-se dizer que um objeto da classe Agregada (Todo) tem objetos da classe componente (Parte)
- Por exemplo: Pode-se imaginar esse tipo de relacionamento como uma casa, que é composta por portas, janelas, paredes, etc.
- A pergunta a ser feita para identificar um relacionamento de agregação é: “é parte de ?”

Todo-Parte (Agregação)



Exemplo Agregação

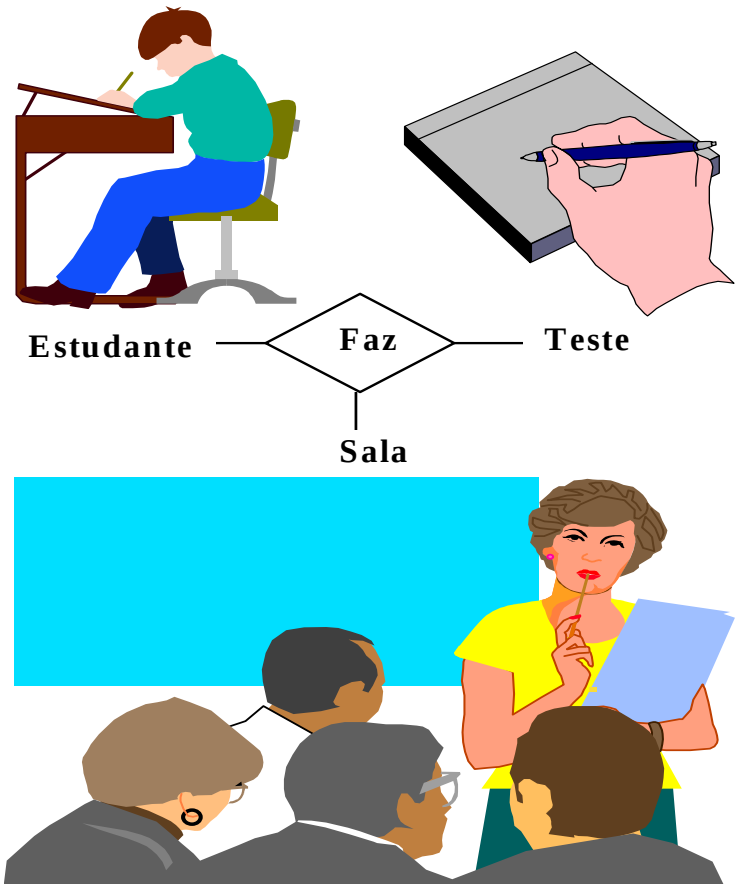


Dessa forma representa-se que um objeto Casa é composto pelos objetos Janela, Porta e Parede

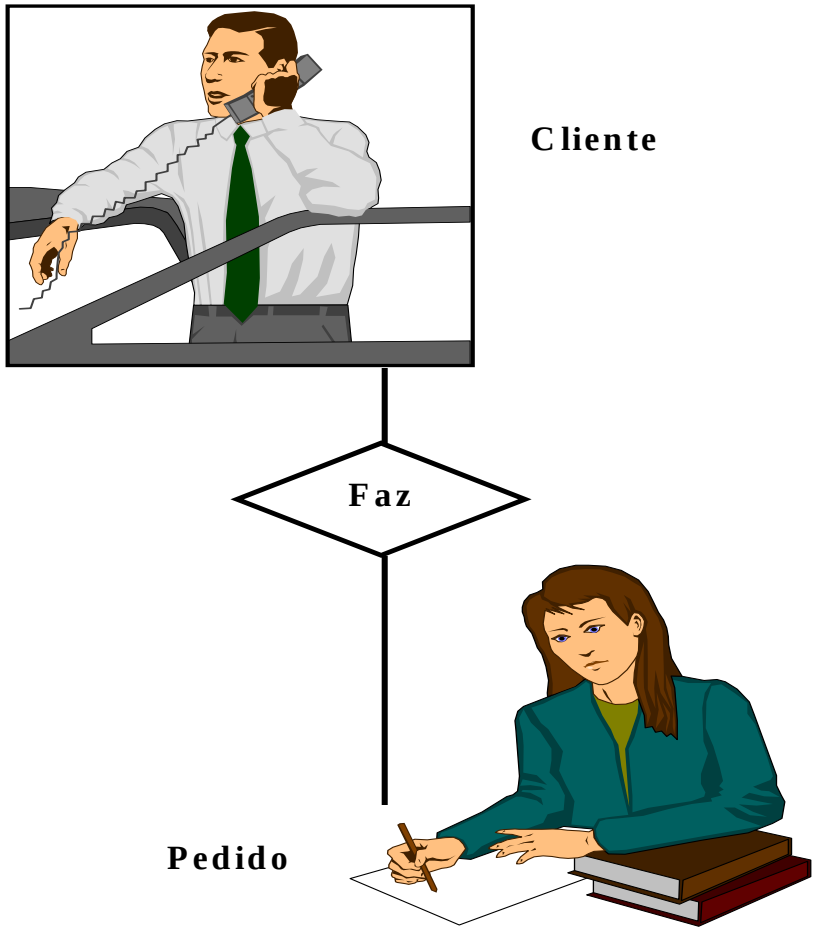
Associação

- Usada para agrupar objetos que ocorrem sob algumas circunstâncias similares ou um ponto específico no tempo
- Associação é um relacionamento estrutural que ocorre entre classes;
- Esse relacionamento existe porque um objeto necessita de outros para cumprir certas responsabilidades;

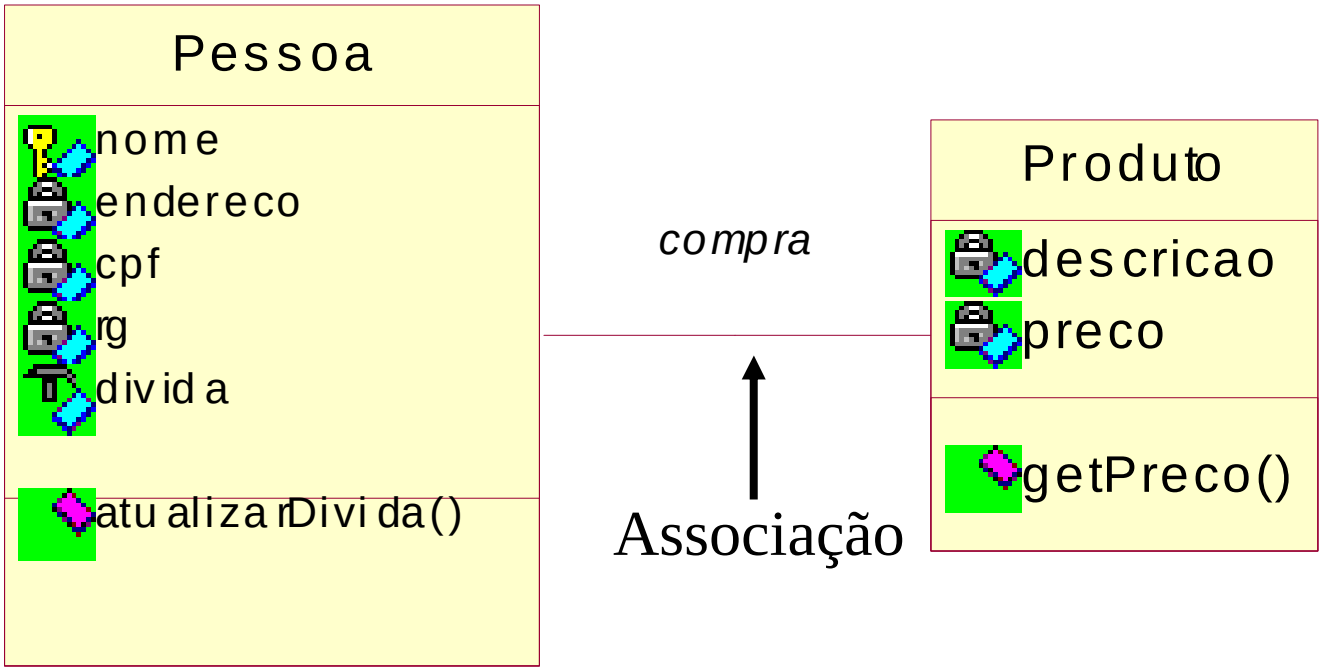
Associação



Associação



Associação - exemplo

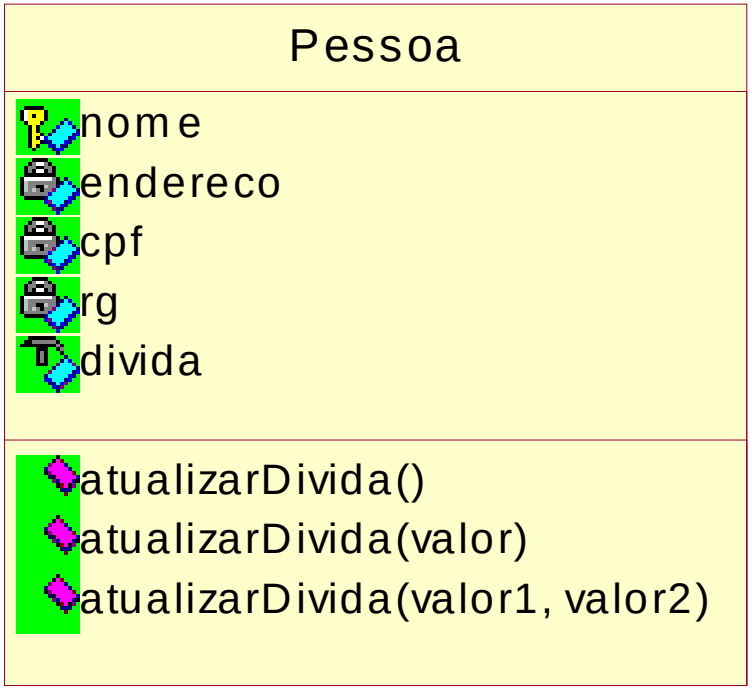


Uma compra é um evento que relaciona uma pessoa e um produto e que ocorre em algum ponto do tempo

Polimorfismo

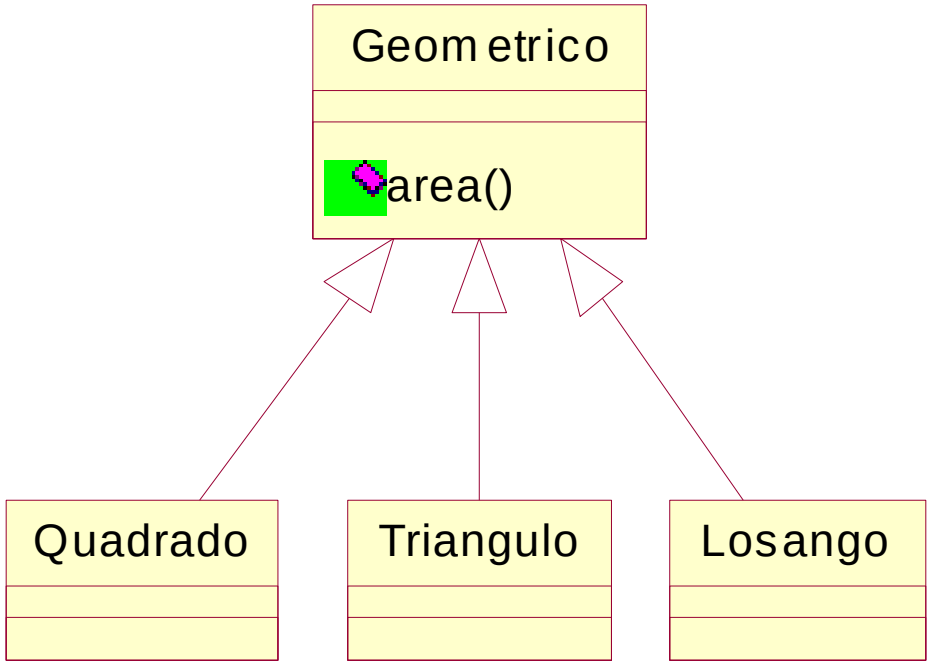
- Refere-se à diferentes formas de um objeto
- Polimorfismo refere-se a capacidade de uma mesma operação realizar funções diferentes dependendo do objeto que a recebe e dos parâmetros que lhes são passados.
- Por exemplo, pode-se ter em uma classe uma operação denominada “calcularDivida()”. Caso essa operação seja invocada sem parâmetros ela realizará algo, caso seja invocada passando um determinado parâmetro realizará algo diferente.

Exemplo – Polimorfismo



Três métodos com o mesmo nome, porém, são diferenciados devido a quantidade de parâmetros passados

Exemplo – Polimorfismo



Dependendo de quem invoca a msg area ela irá calcular a área do objeto correto.