Projeto da Camada de Domínio

- Diagramas de Colaboração/Comunicação
- Diagrama de Classes de Projeto (DCP)

Projeto da Camada de Domínio

- Diagramas de Colaboração (Comunicação na UML 2) permitem realizar a modelagem dinâmica do sistema, ou seja, como os objetos que fazem parte da arquitetura trocam mensagens para realizar suas responsabilidades.

Em relação ao Modelo Conceitual, o DCP apresenta:

- *Adição dos métodos*
- *Adição da direção das associações*
- *Possível detalhamento dos atributos e associações*
- *Possível alteração na estrutura das classes e associações*
- *Possível criação de atributos privados ou protegidos*

Pseudocódigo concentrador

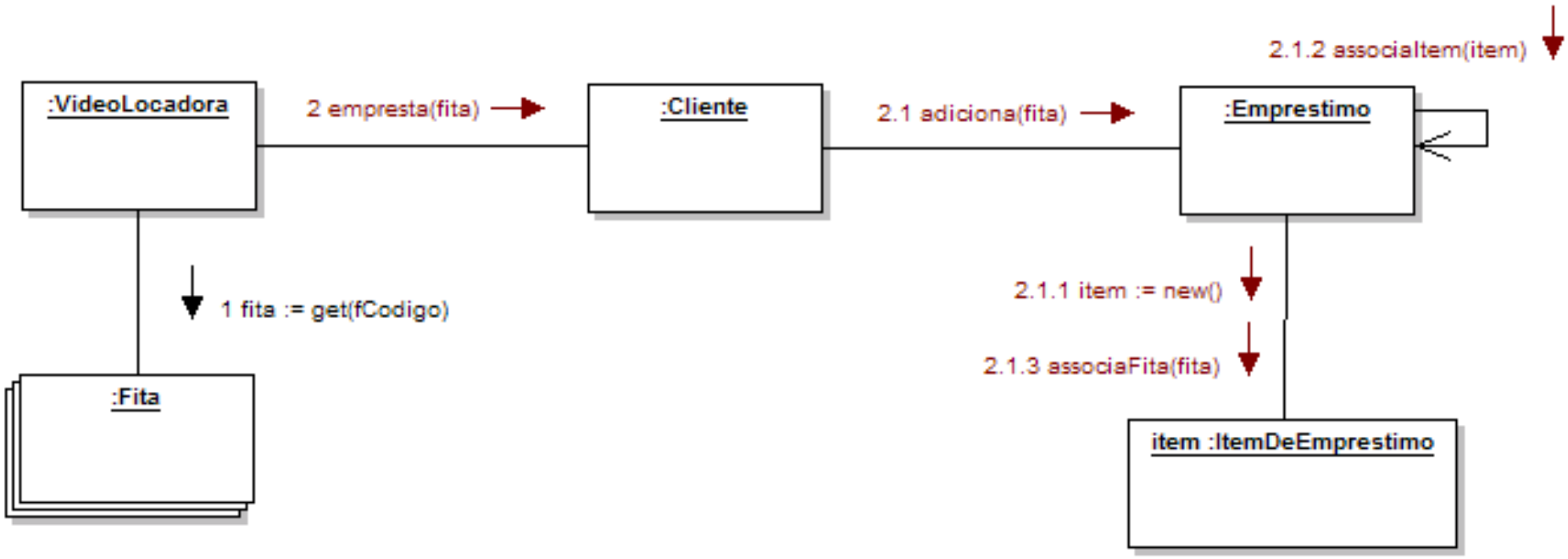
Classe VideoLocadora

fitas : Conjunto;
clienteCorrente : Cliente;

Método emprestaFita(fCodigo: String)

```
    fita : Fita;
    emprestimoCorrente : Emprestimo;
    item : ItemDeEmprestimo;

    fita := fitas.get(fCodigo);
    emprestimoCorrente := clienteCorrente.getEmprestimoCorrente();
    item := ItemDeEmprestimo.new();
    item.associaFita(fita);
    emprestimoCorrente.associaItem(item);
Fim Método;
Fim Classe.
```

Diagrama de Colaboração



Código com Responsabilidades Distribuídas

Classe VideoLocadora

fitas : Conjunto ;
clienteCorrente : Cliente;

Metodo emprestaFita(fCodigo : String);
    fita : Fita;

    fita := fitas.get(fCodigo);
    clienteCorrente.empresta(fita)
Fim Metodo;
Fim Classe.

Classe Cliente

emprestimoCorrente : Emprestimo;

Metodo empresta(fita : Fita);
    emprestimoCorrente.adiciona(fita);
Fim Metodo;
Fim Classe.

Classe Emprestimo

itens : Conjunto;

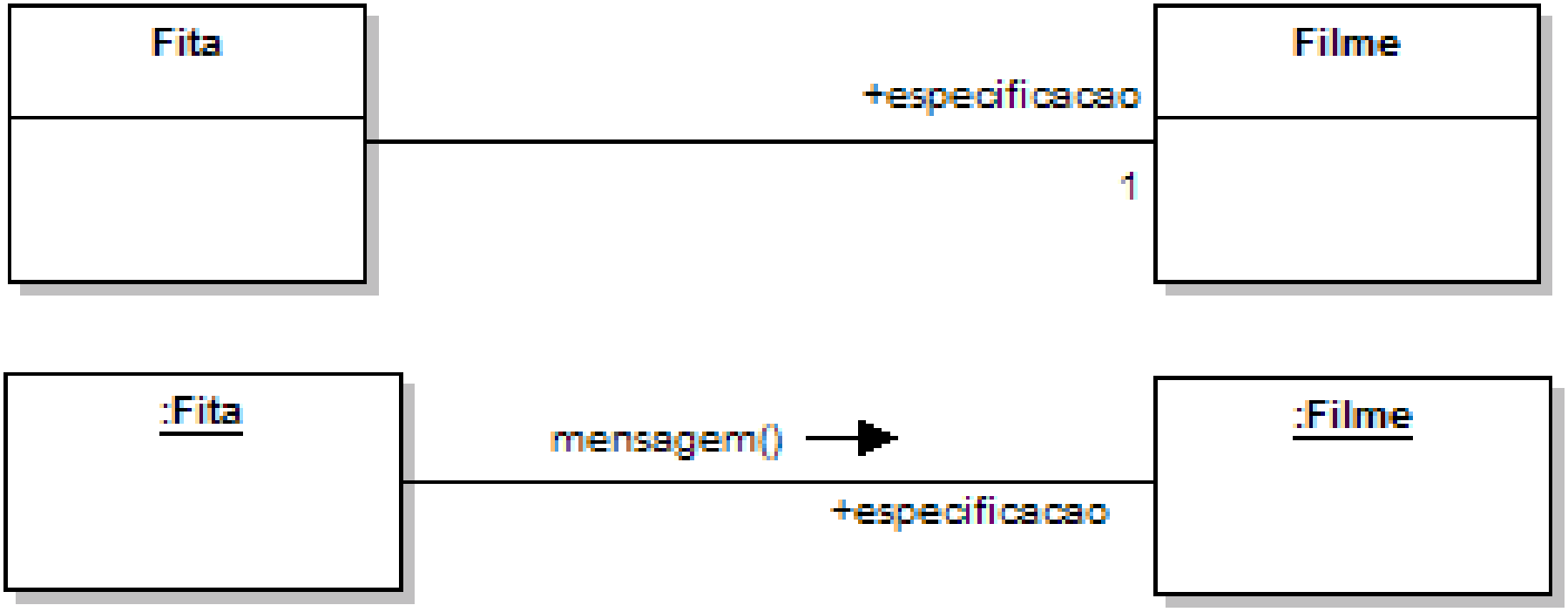
Metodo adiciona(fita : Fita);
    item : ItemDeEmprestimo;

    item := ItemDeEmprestimo.new();
    self.associaItem(item);
    item.associaFita(fita);
Fim Metodo;
Fim Classe.

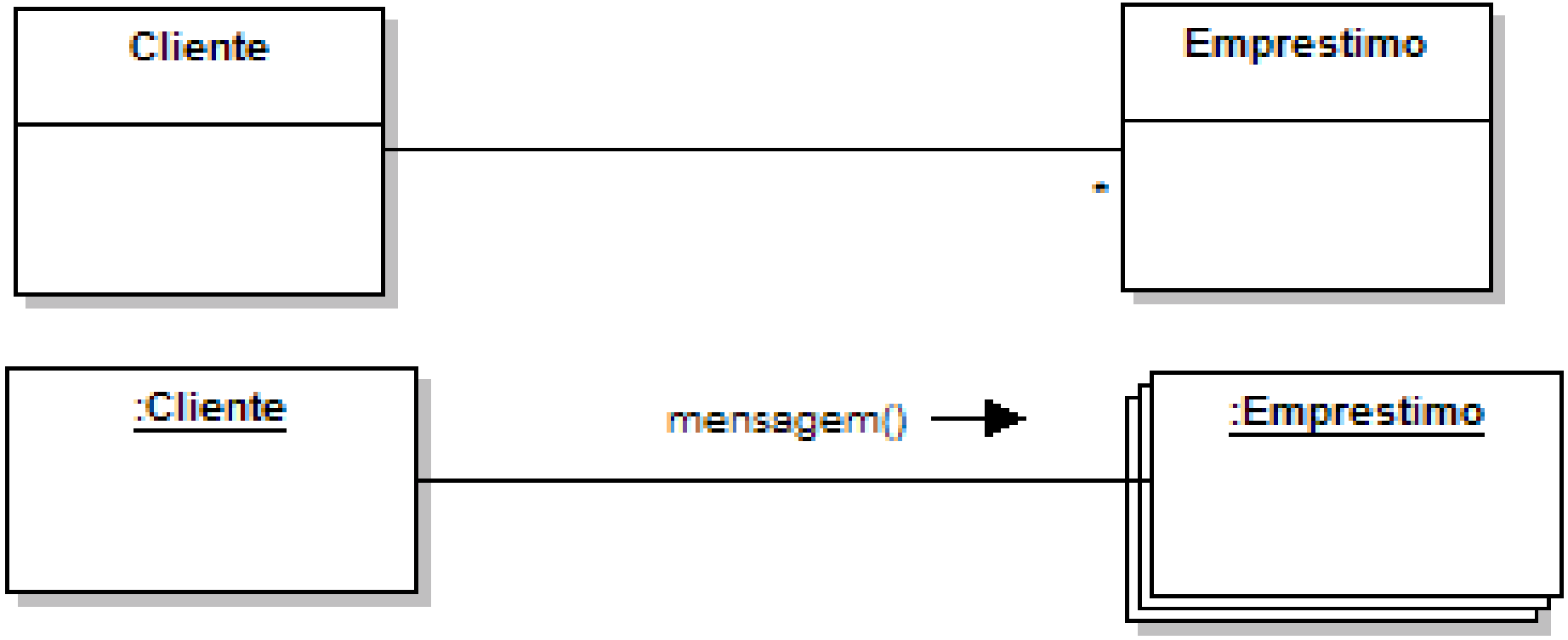
Visibilidade

- *Por associação*. Quando as classes de dois objetos estão associadas no DCP
- *Por parâmetro*. Quando um objeto recebe outro como parâmetro em um método
- *Localmente declarada*. Quando um objeto recebe outro como retorno de um método
- *Global*. Quando um objeto é declarado globalmente

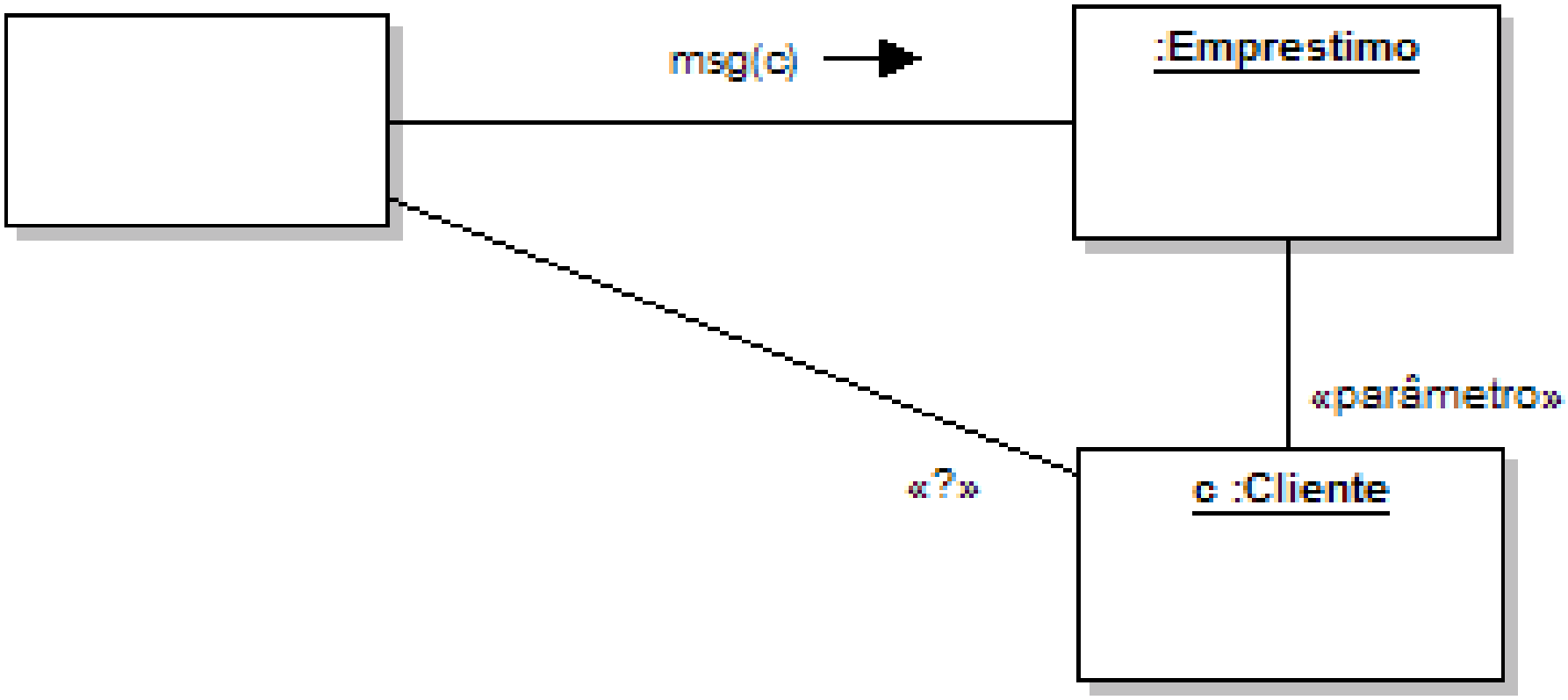
Visibilidade por Associação (para 1)



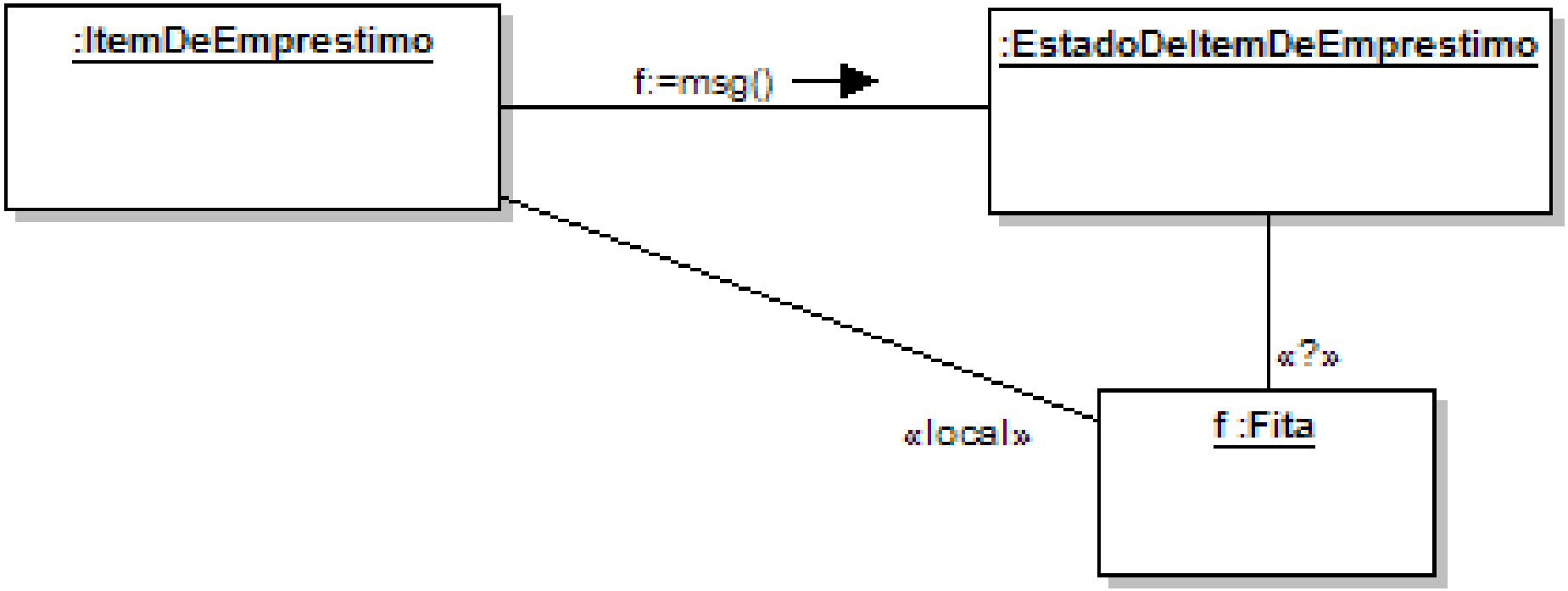
Visibilidade por Associação (para muitos)



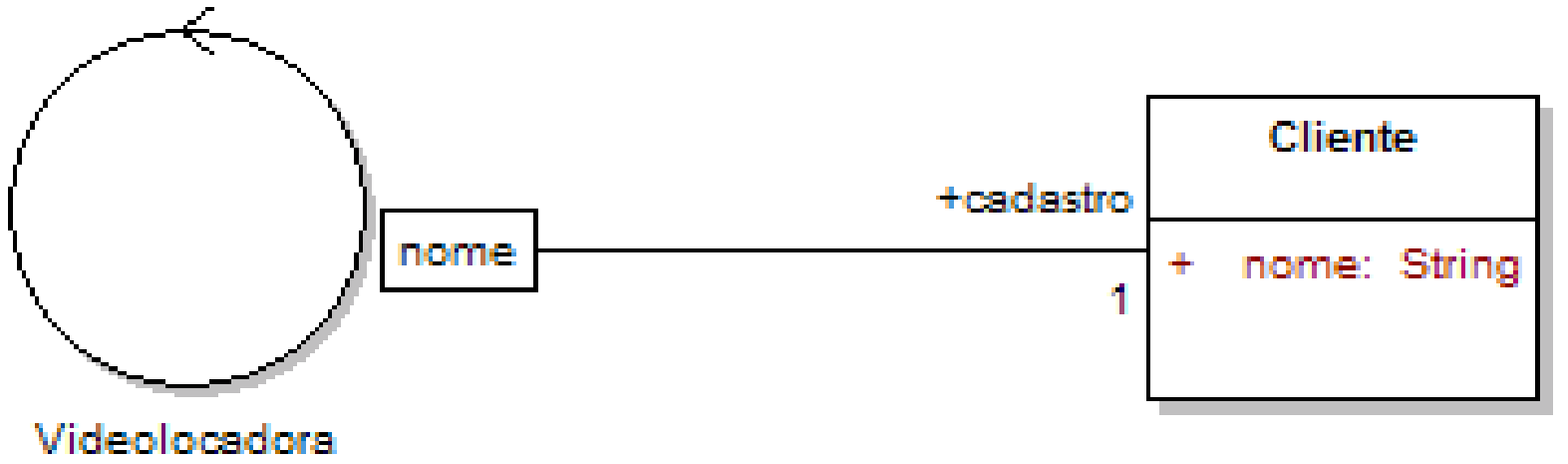
Visibilidade por Parâmetro

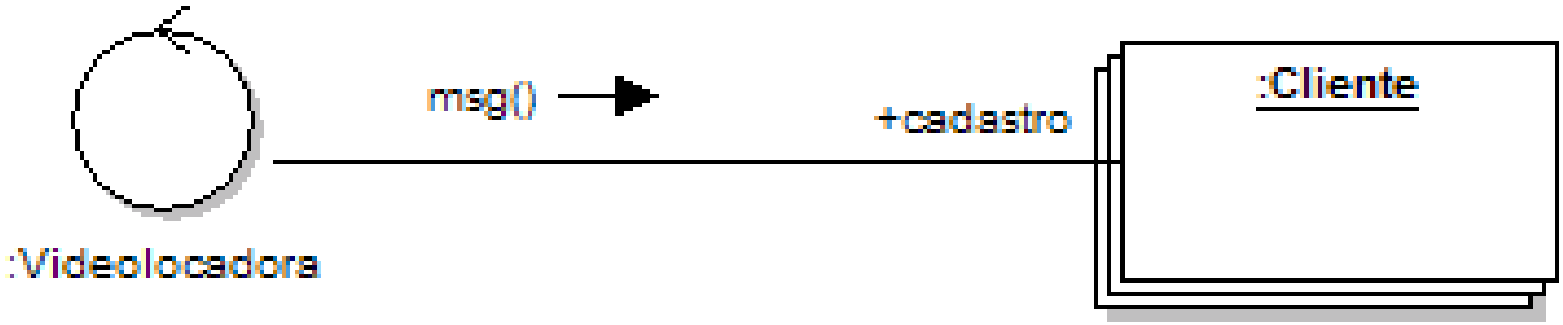


Visibilidade Localmente Declarada

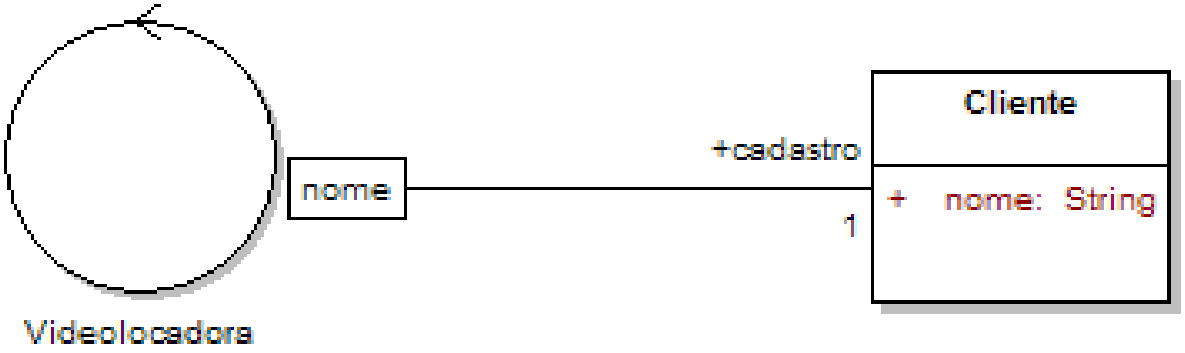


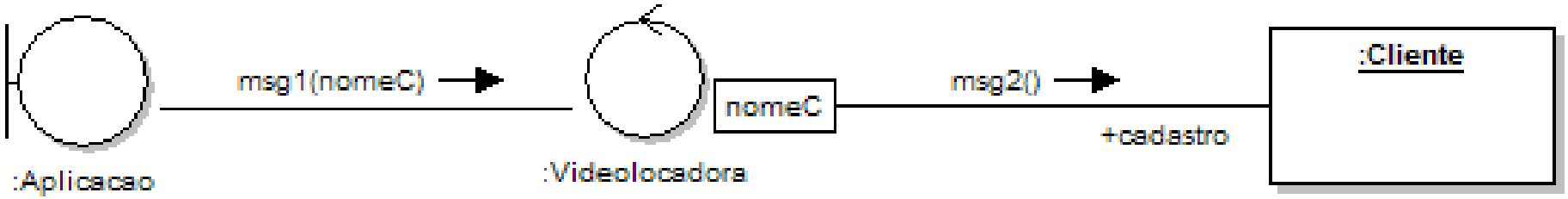
Visibilidade por Associação (qualificada – sem qualificador)



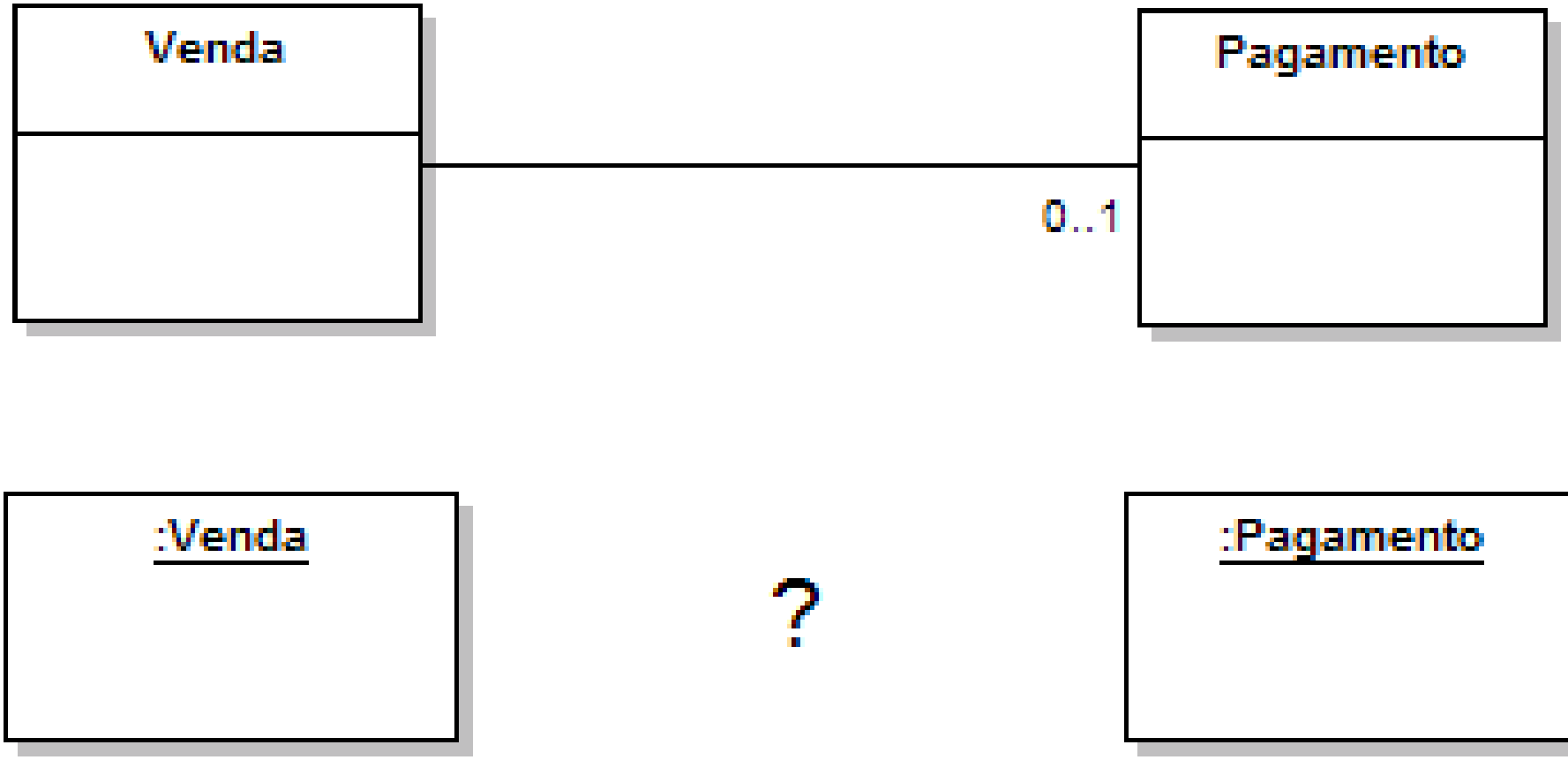


Visibilidade por Associação (qualificada – com qualificador)



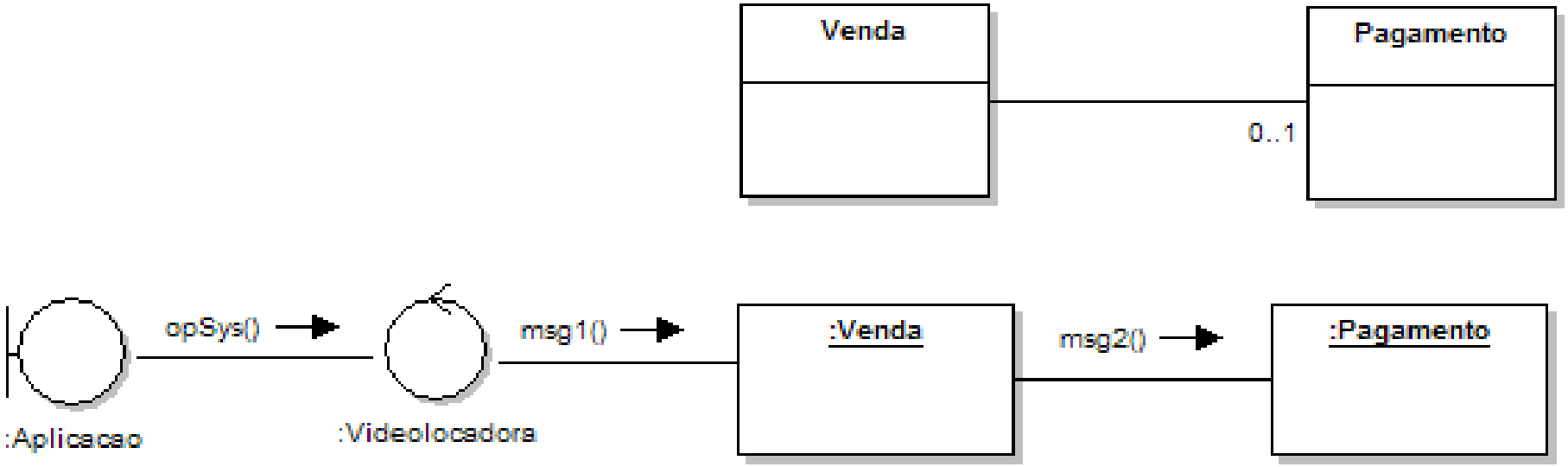


Visibilidade por Associação (0..1)

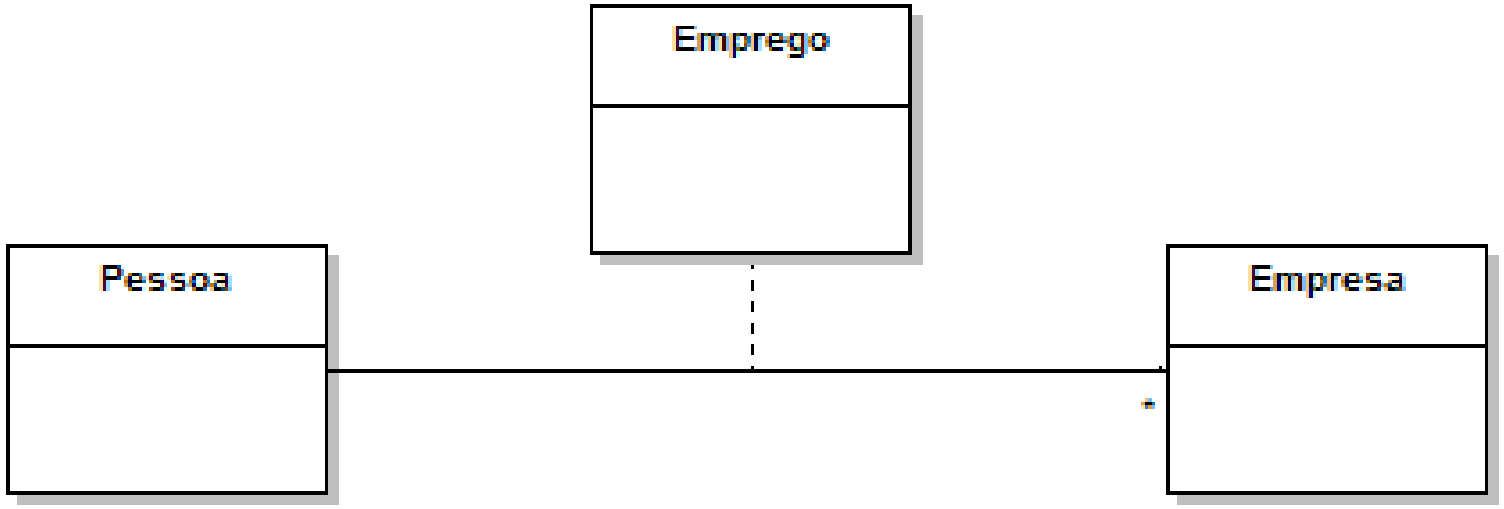


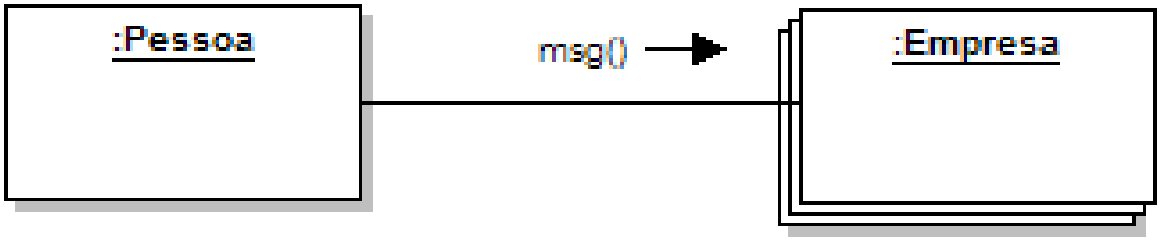
Estabelecida por Pré-Condição de Contrato

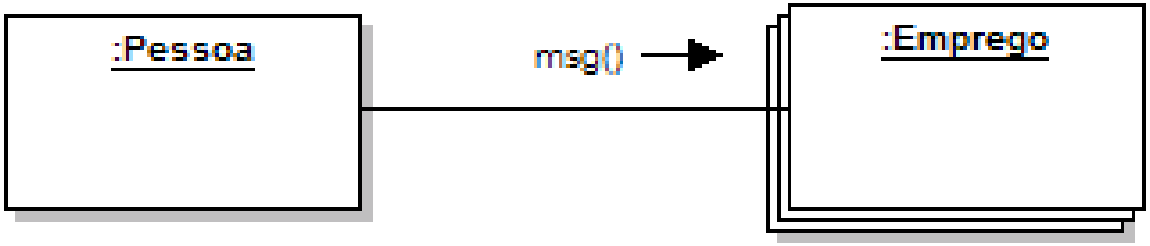
- existe um Pagamento associado à



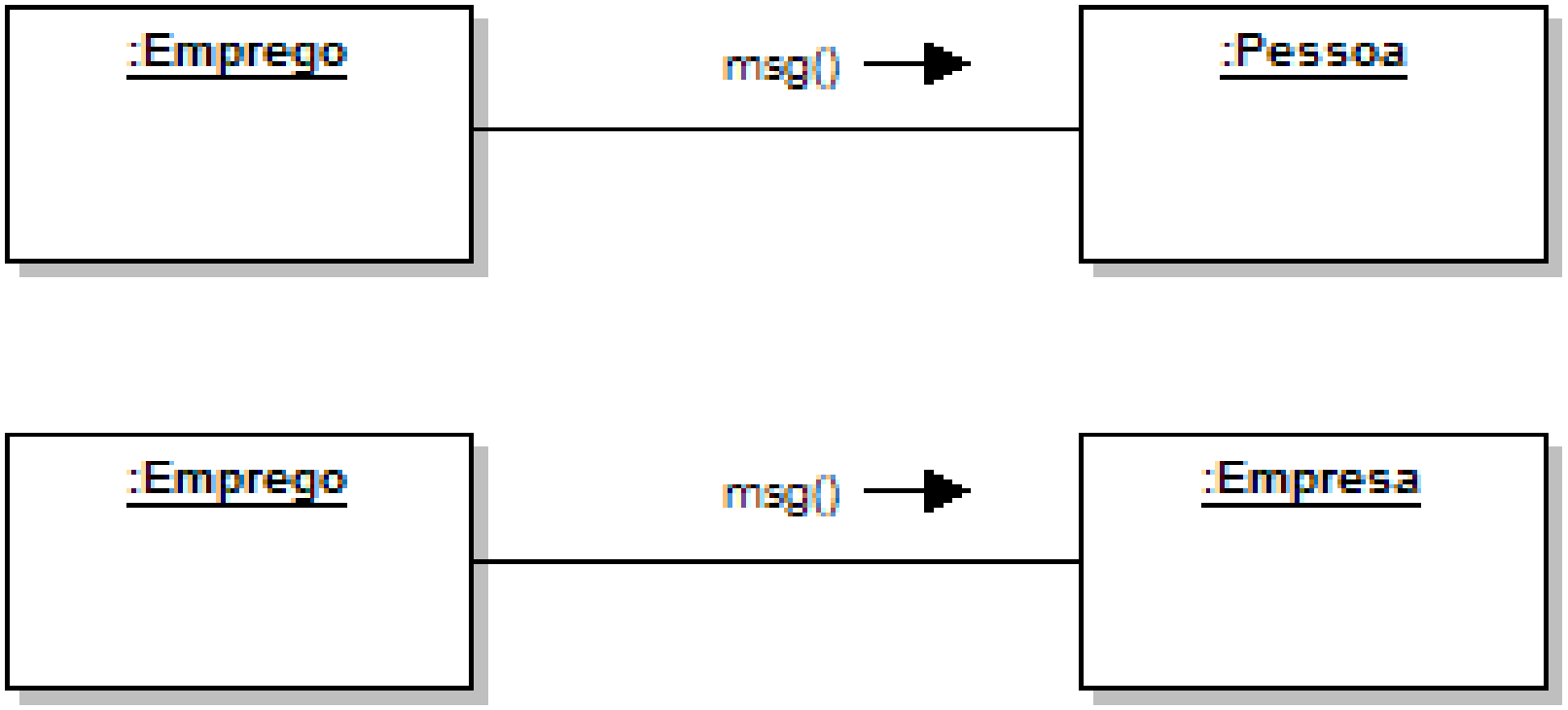
Visibilidade de Classes de Associação



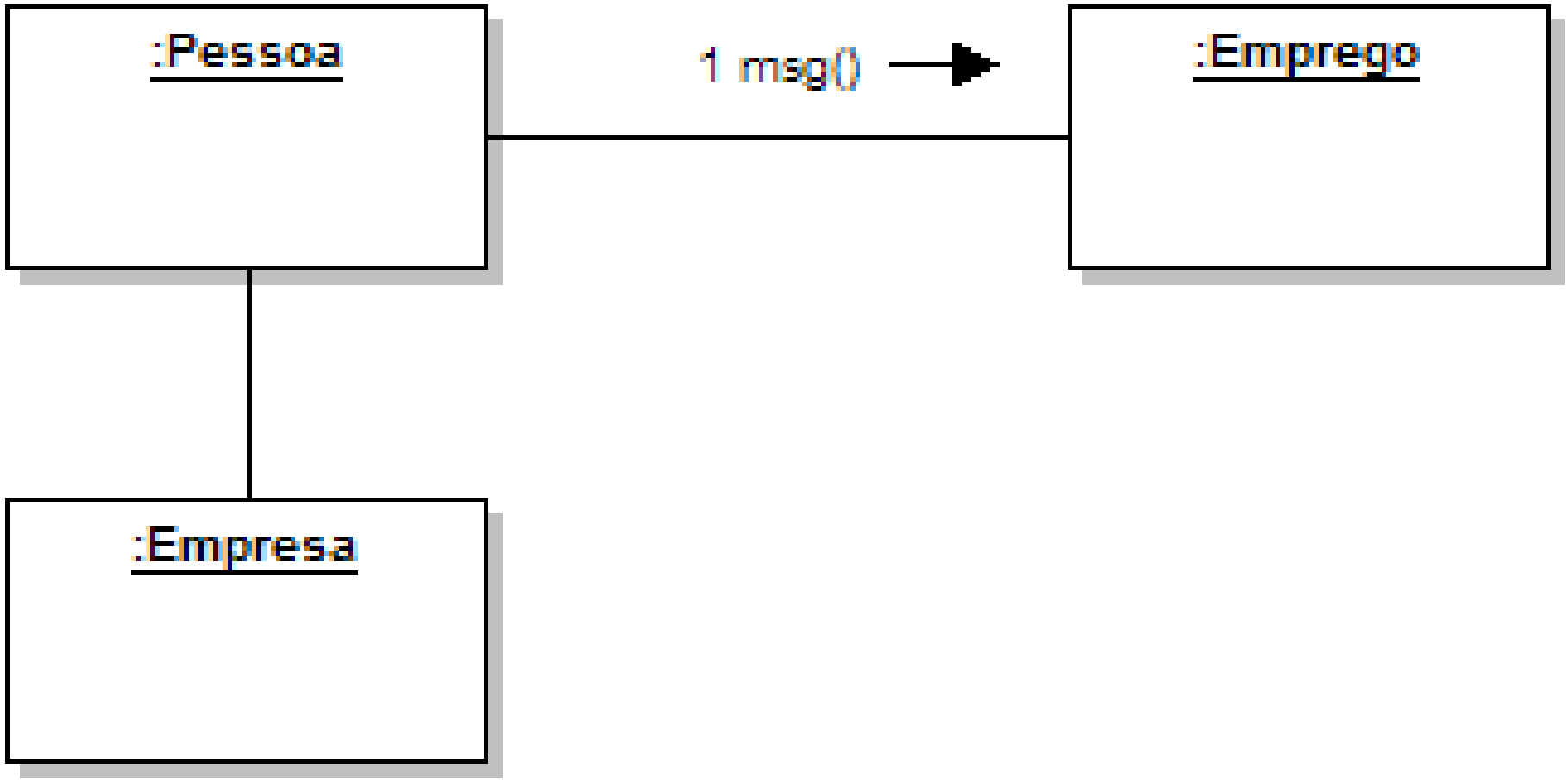




Do ponto de vista da classe de associação:

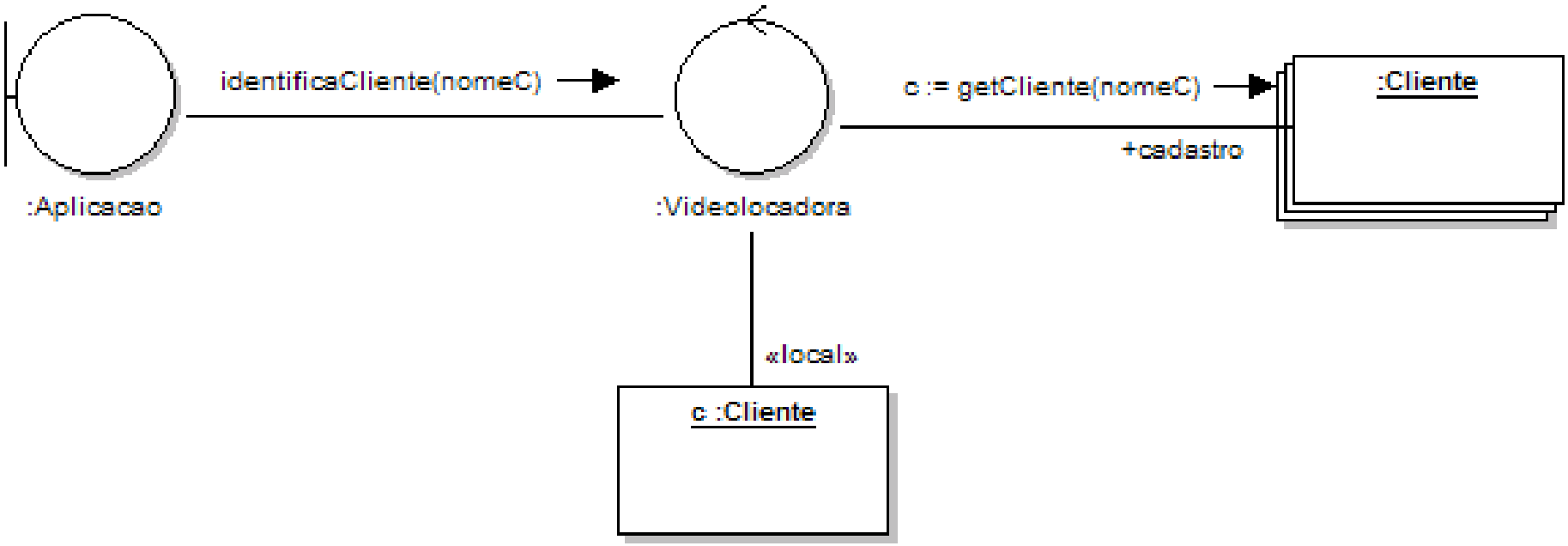


Com visibilidade para uma instância associada



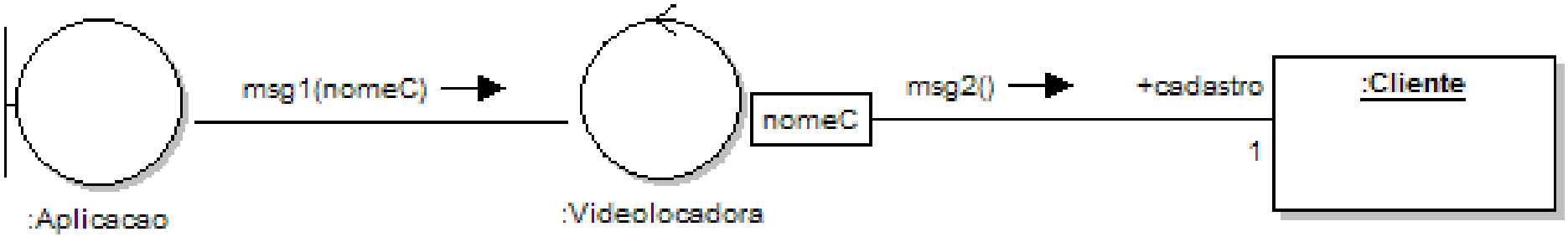
Influência das Pré-Condições de Contrato nos Diagramas

- Garantia de Parâmetros (associação não qualificada)



Influência das Pré-Condições de Contrato nos Diagramas

- Garantia de Parâmetros (associação qualificada)



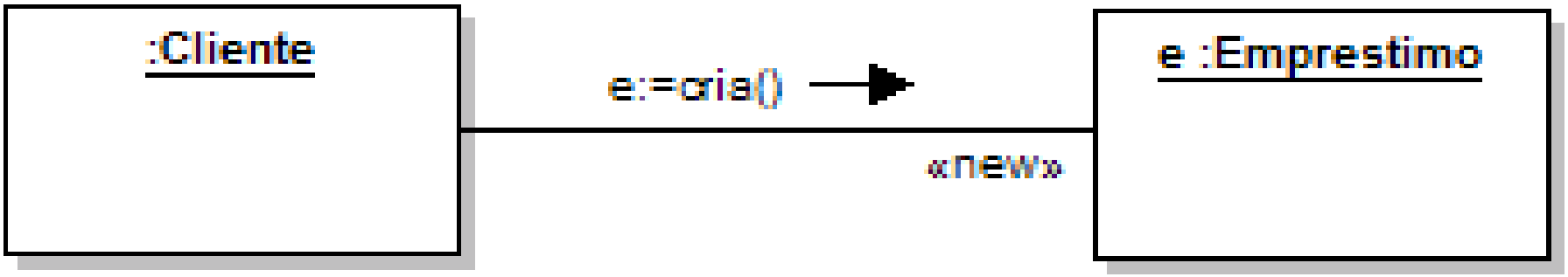
Realização das Pós-Condições dos Contratos nos Diagramas

- *Mensagens básicas*. São aquelas que efetivamente realizam aquilo que a pós-condição requer
- *Mensagens delegadas*. Passam adiante a responsabilidade de realizar uma operação básica quando o objeto que detém o controle da execução não possui visibilidade direta para o objeto que deve executar a operação

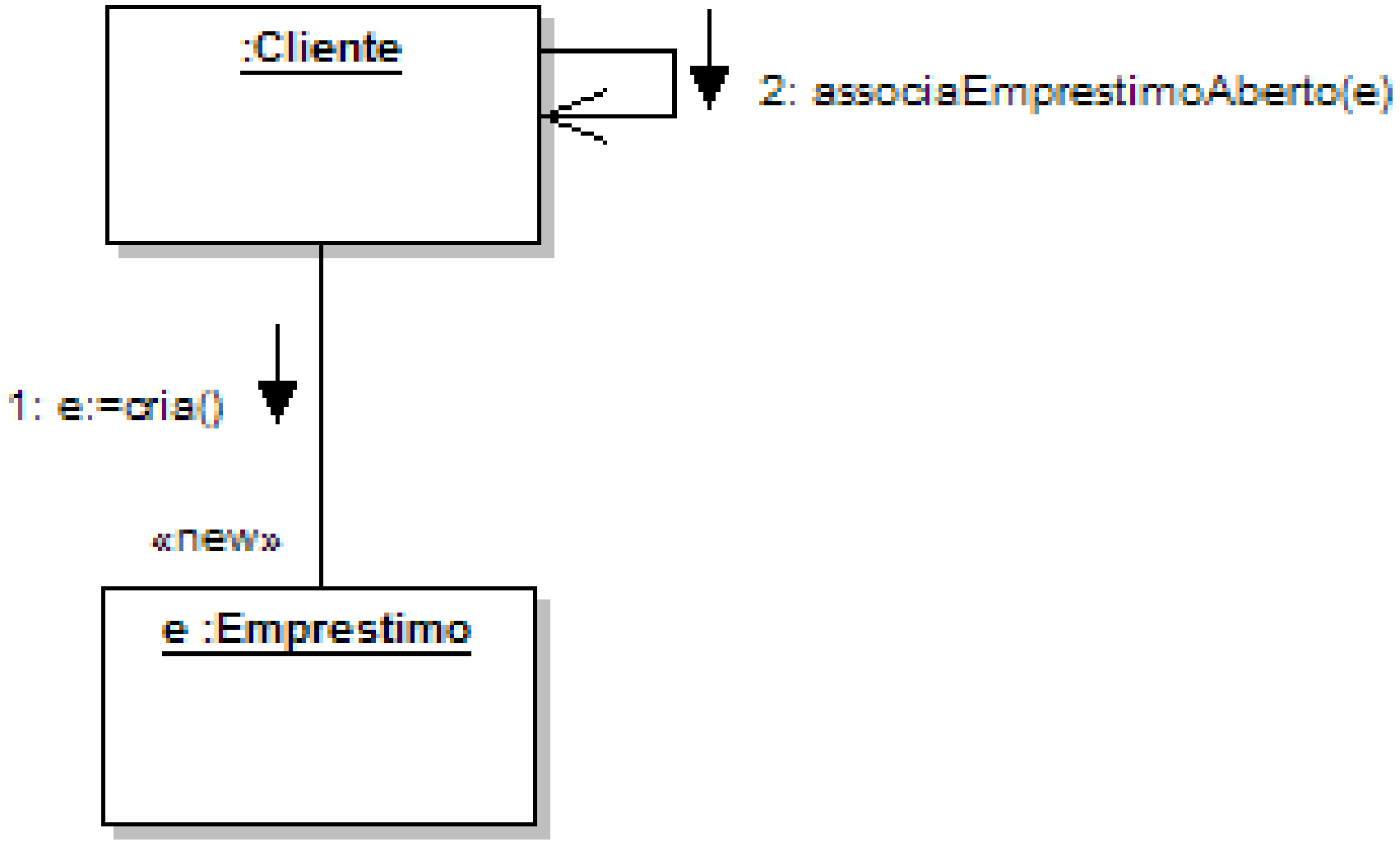
Mensagens Básicas

- Criação de instância
- Destruição de instância
- Criação de associação
- Destruição de associação
- Consulta de associação
- Alteração de valor de atributo
- Consulta de valor de atributo

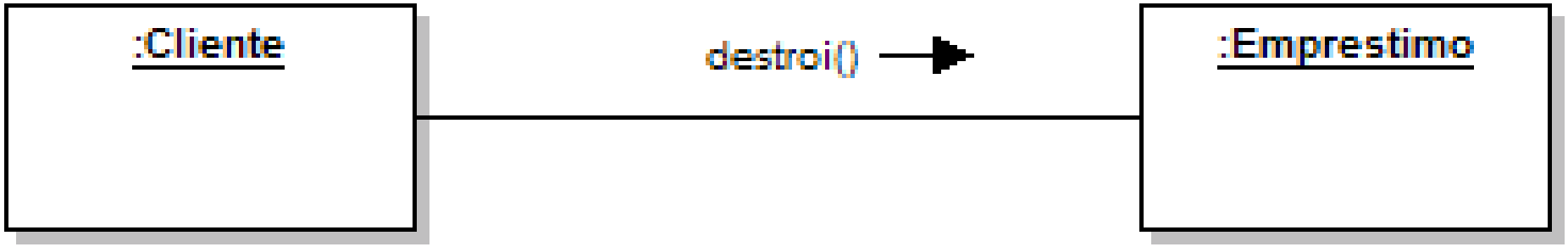
Criação de Instância



Cria e imediatamente associa

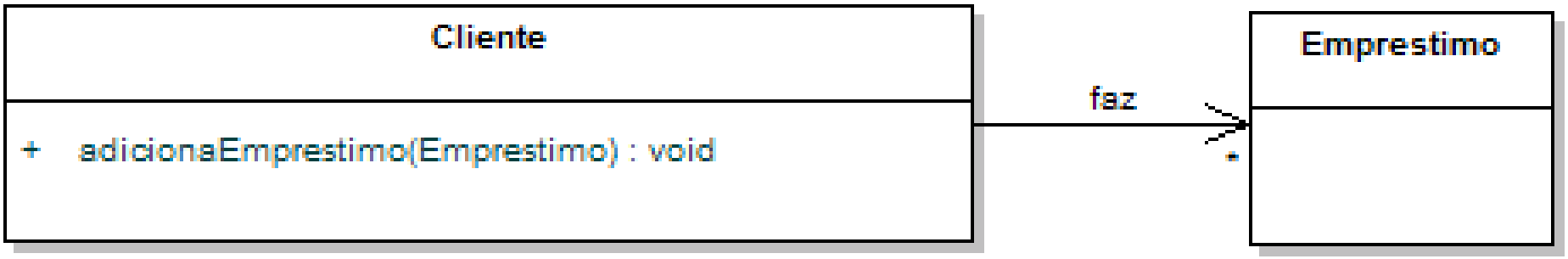


Destruição de Instância



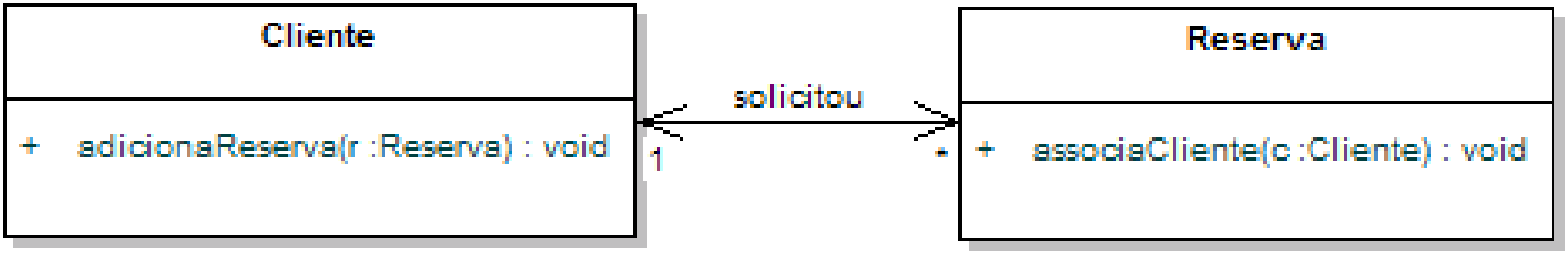
Criação de Associação

- Método implementado na origem da associação



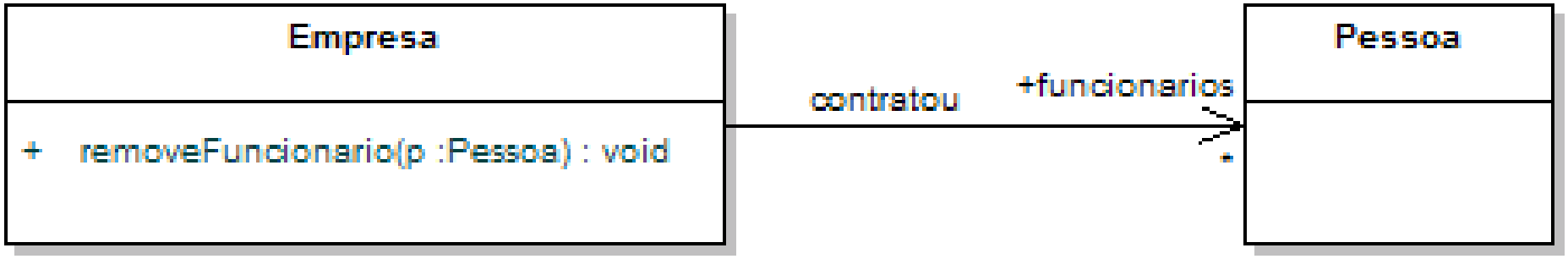
Criação de Associação

- Associação bidirecional

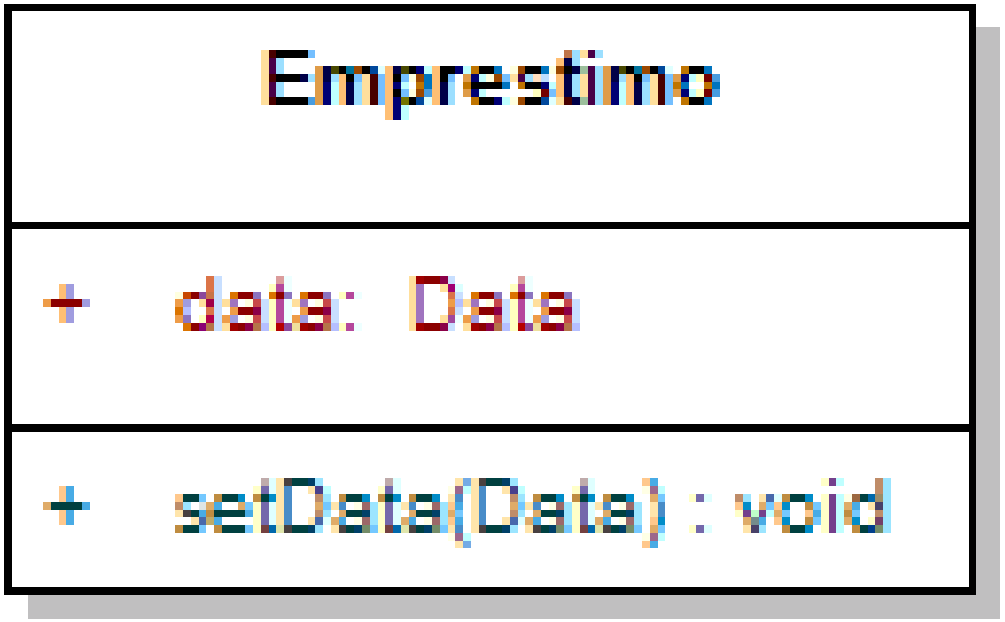


Destruição de Associação

- Implementação na origem



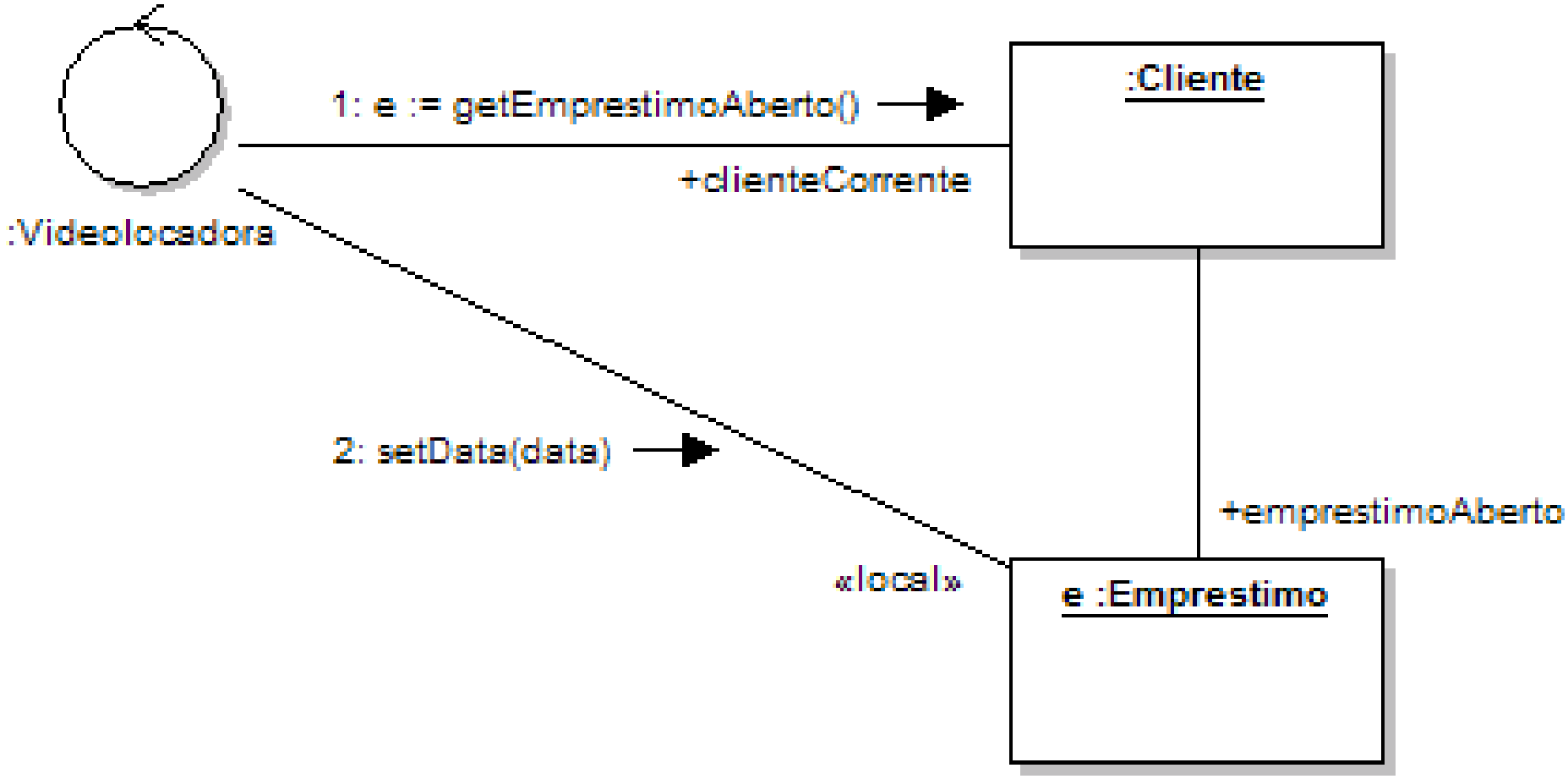
Modificação de Valor de Atributo



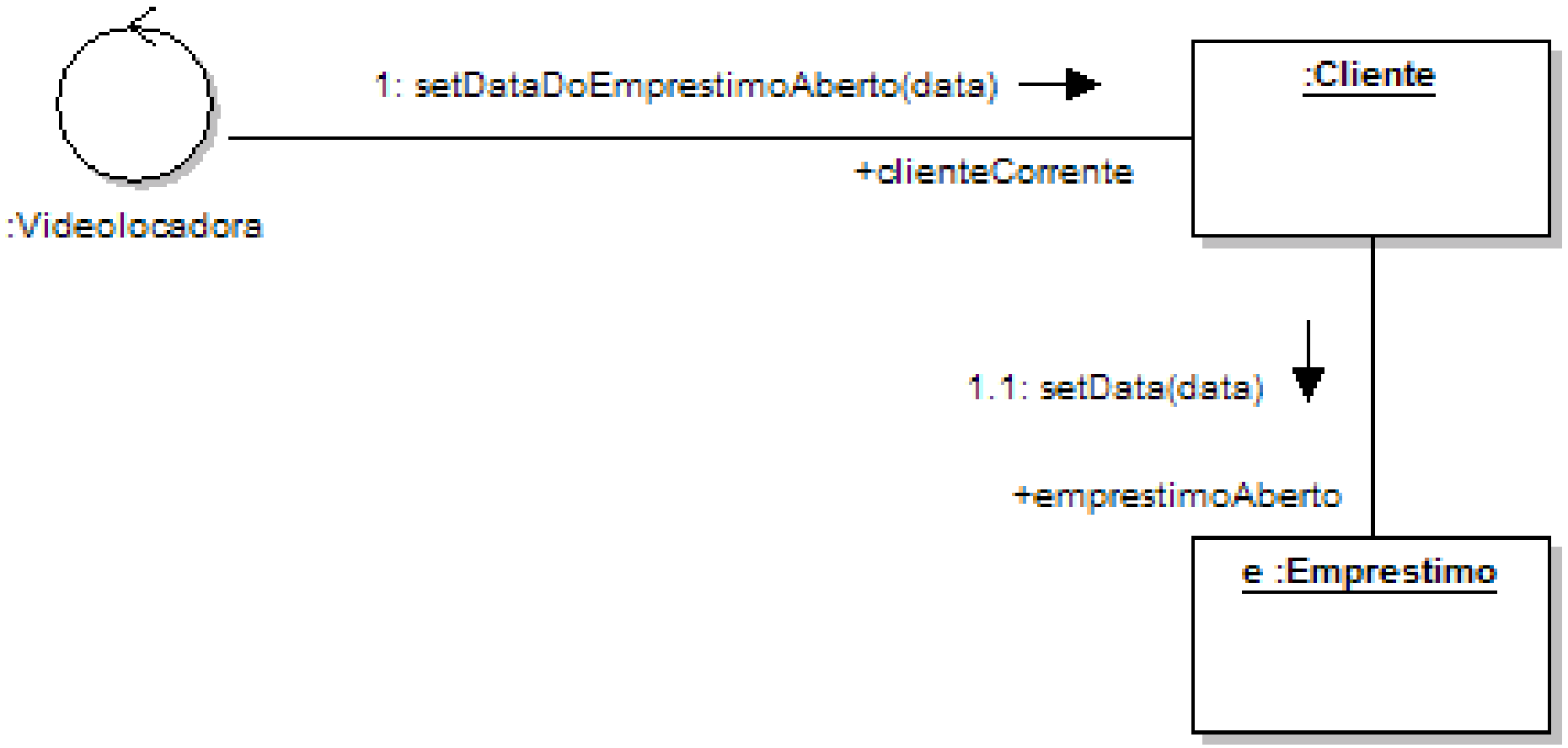
Delegação

- Faculta o acoplamento fraco

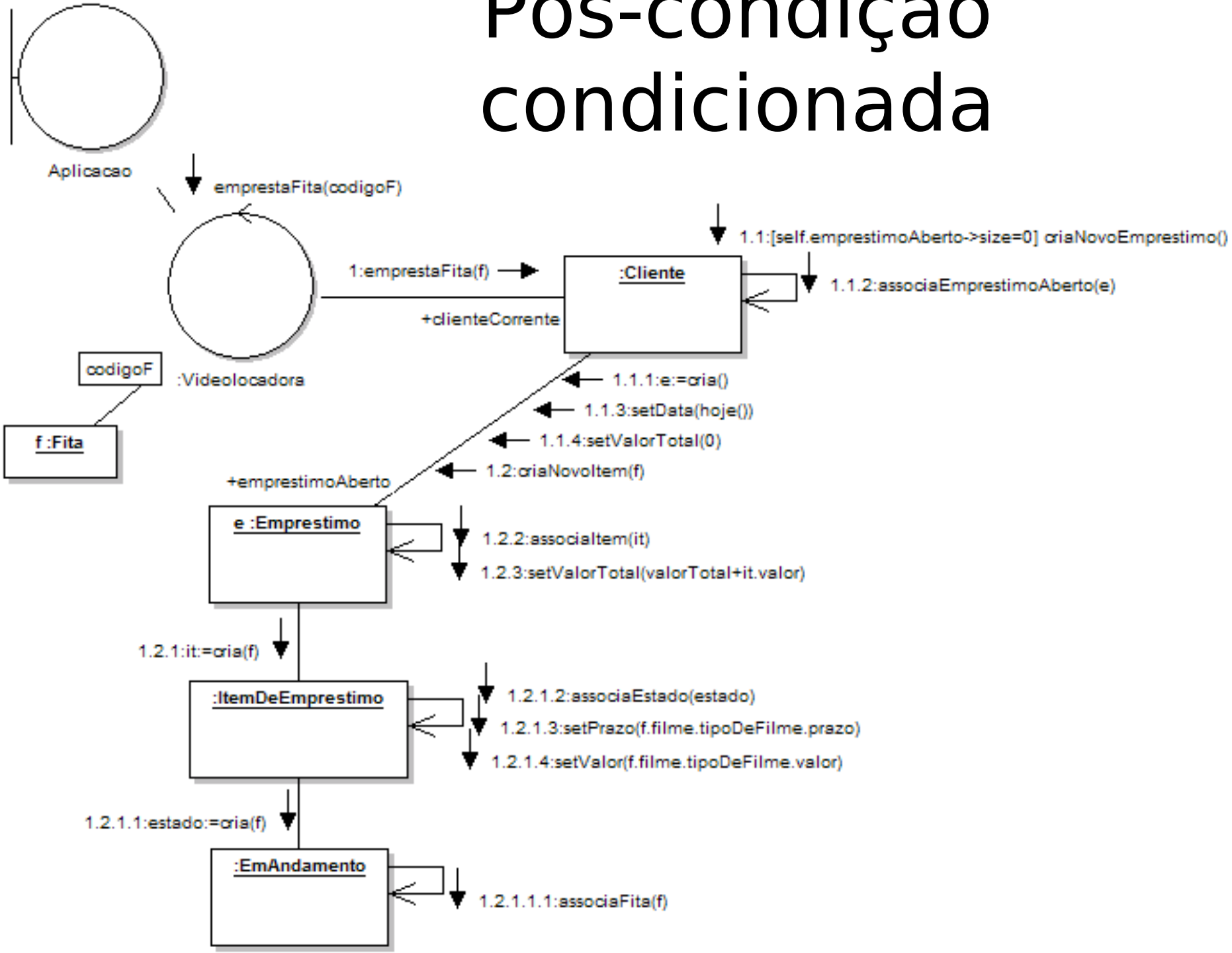
Estilo de projeto sem delegação



Estilo de projeto com delegação

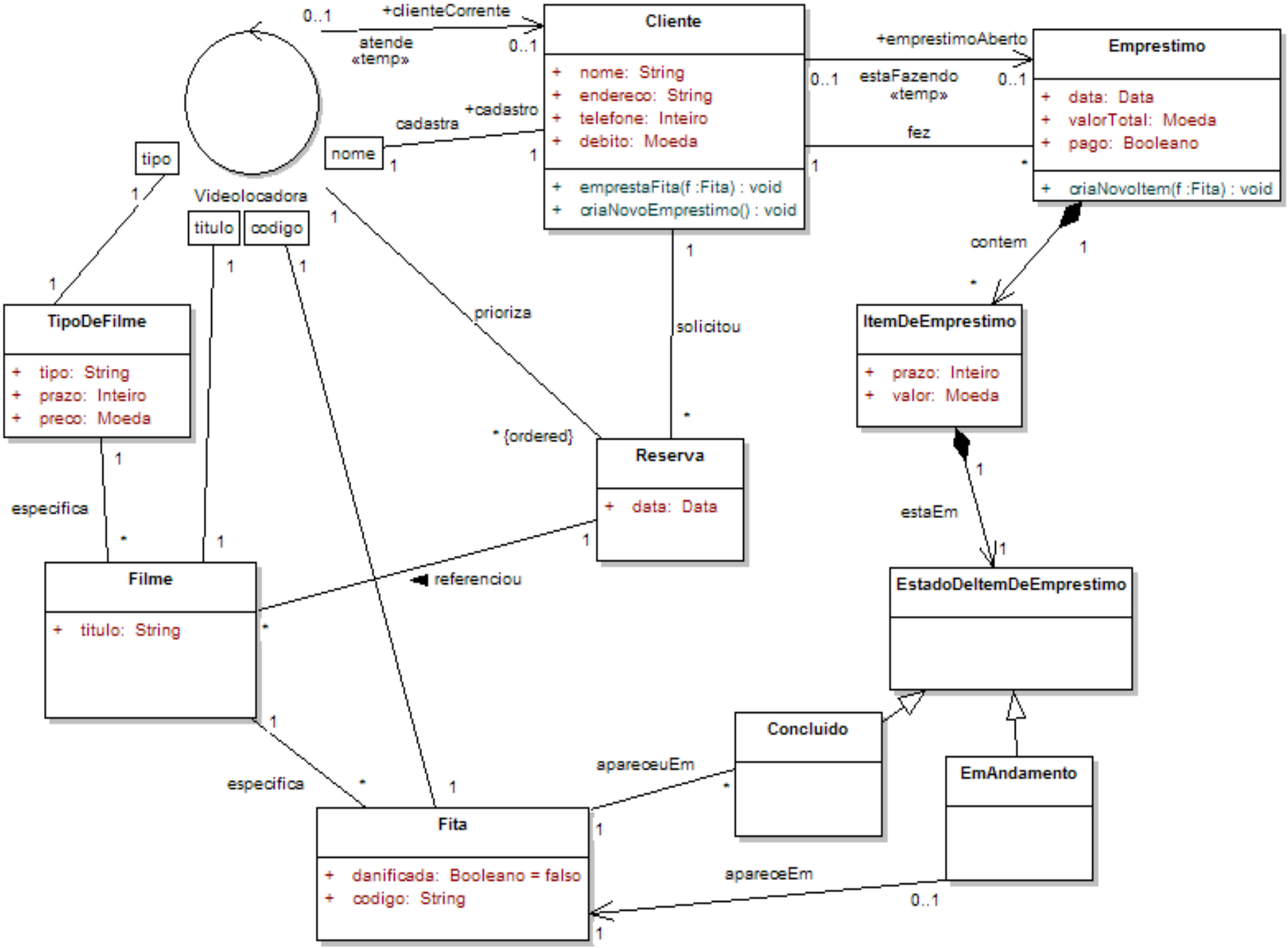


Pós-condição condicionada



Contribuições dos Diagramas de Colaboração ao DCP

- *Métodos delegados*. Sempre que um objeto receber uma mensagem delegada, a classe correspondente ao objeto deve registrar a implementação deste método
- *Sentido das associações*. O sentido das associações no DCP corresponderá ao sentido do envio das mensagens sobre as ligações de visibilidade baseadas em associações

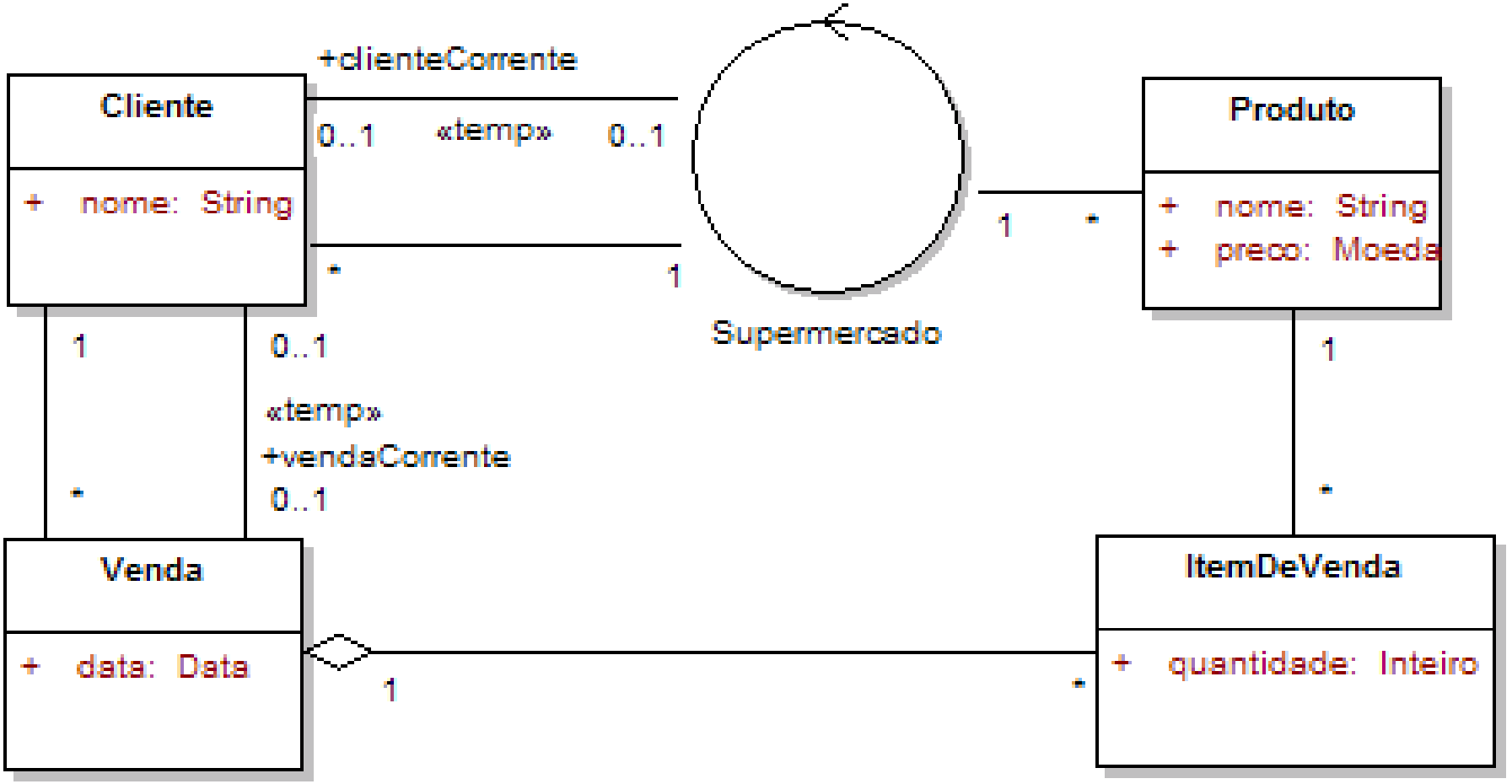

Cliente
+ nome: String
+ endereco: String
+ telefone: Inteiro
+ debito: Moeda
+ emprestaFita(f :Fita) : void
+ criaNovoEmprestimo() : void
Emprestimo
+ data: Data
+ valorTotal: Moeda
+ pago: Booleano
+ criaNovoItem(f :Fita) : void
Videolocadora
TipoDeFilme
+ tipo: String
+ prazo: Inteiro
+ preco: Moeda
ItemDeEmprestimo
+ prazo: Inteiro
+ valor: Moeda
Reserva
+ data: Data
Filme
+ titulo: String
EstadoDeItemDeEmprestimo
Concluido
EmAndamento
Fita
+ danificada: Boolean = falso
+ codigo: String
0..1
+clienteCorrente
atende
«temp»
+cadastro
cadastra
nome
tipo
titulo
codigo
+emprestimoAberto
estaFazendo
«temp»
fez
contem
prioriza
solicitou
* {ordered}
especifica
referenciou
estaEm
apareceuEm
apareceEm

Design Patterns Básicos

- Especialista
- Criador
- Acoplamento Fraco
- Coesão Alta

Especialista

- Quem implementa a consulta “valorTotalDaVenda”?



Pseudocódigo que não atende ao padrão Especialista

Classe Supermercado

clienteCorrente : Cliente;

consulta valorTotalDaVendaCorrente();

venda : Venda;
item : ItemDeVenda;
total : Moeda = 0,00;

venda := clienteCorrente.getVendaCorrente();
repita para cada item em venda.getItensDeVenda():
total := total + (item.getQuantidade() * item.getProduto().getPreco());
fim repita
retorna total

fim consulta

Fim Classe.

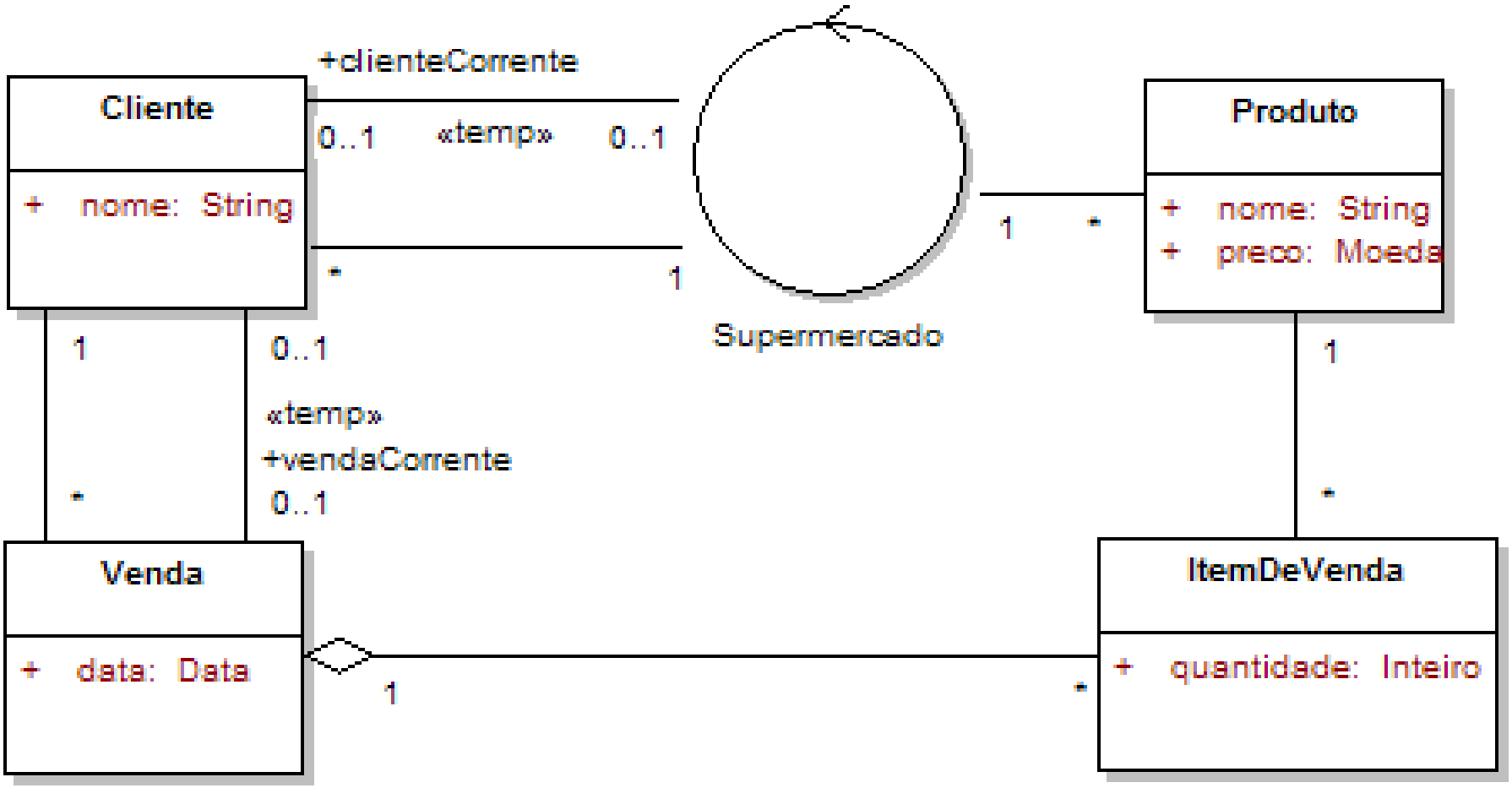
Pseudocódigo que atende ao padrão especialista

```
Classe Supermercado
        clienteCorrente : Cliente;

        consulta valorTotalDaVendaCorrente();
                retorna clienteCorrente.getValorTotalDaVendaCorrente();
        fim consulta
Fim Classe.

Classe Cliente
        vendaCorrente : Venda;

        consulta getValorTotalDaVendaCorrente();
                retorna vendaCorrente.getValorTotal();
        fim consulta
Fim Classe

Classe Venda
        itens : Conjunto de ItemDeVenda;
        total : Moeda = 0,00;

        consulta getValorTotal()
                repita para cada item em itens:
                        total := total + item.getSubtotal();
                fim repita
                retorna total;
        fim consulta
Fim Classe

Classe ItemDeVenda
        produto : Produto;

        consulta getSubtotal()
                retorna getQuantidade() * produto.getPreco();
        fim consulta
Fim Classe
```

Criador – quem deve criar uma instância?

- Em primeiro lugar, verifique se o objeto é parte de uma agregação ou composição. Se for, o criador será o objeto agregador
- Caso contrário, verifique se alguma classe tem associação de 1 para * ou de 1 para 0..1 para a classe do objeto a ser criado. Se existir, uma classe nessa situação e ela estiver em algum caminho possível na direção do controlador, então ela poderá ser a criadora
- Se houver empate, decida pela classe que parecer mais fortemente associada à classe a ser criada. Esse critério é subjetivo, mas é exatamente onde entra o julgamento o projetista sobre qual a opção mais adequada para fazer um caminho de delegação até a operação básica de criação de um objeto

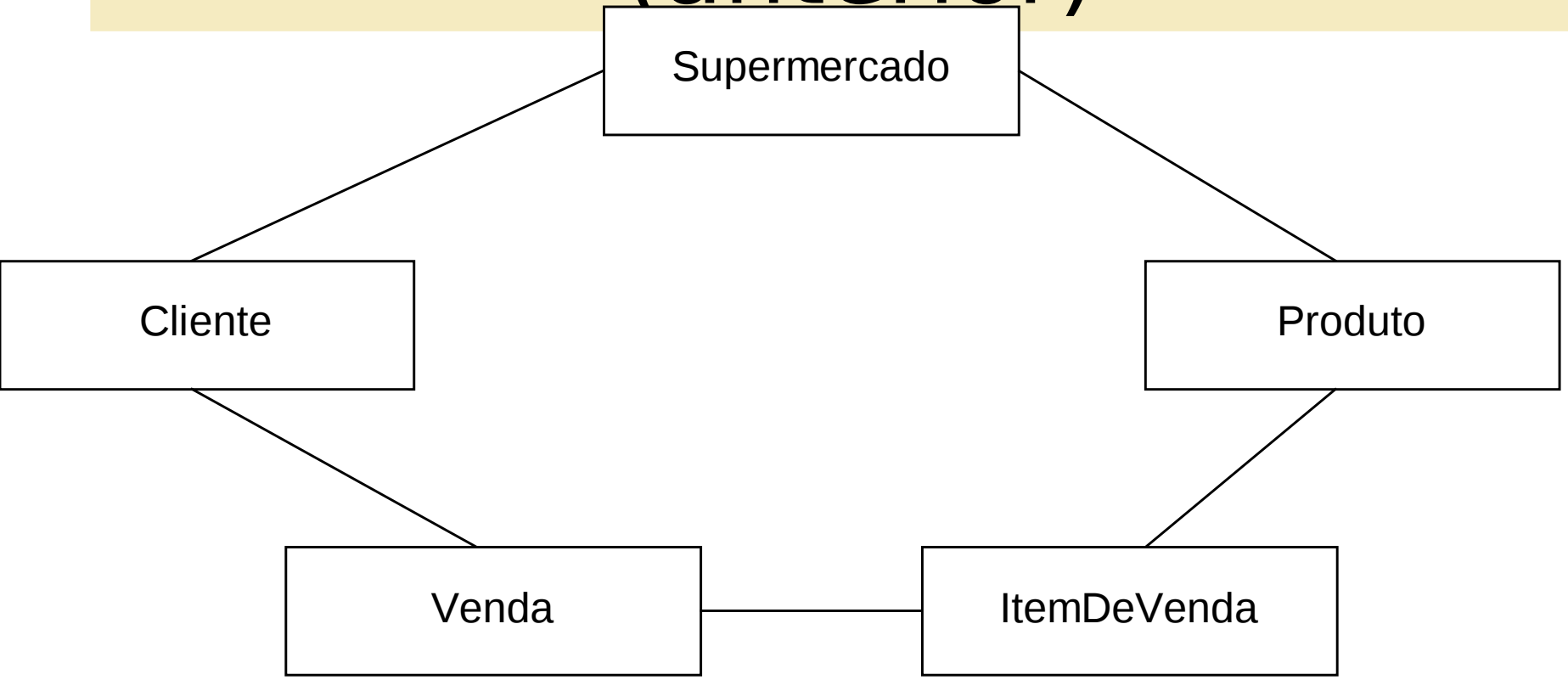
Quem cria quem?



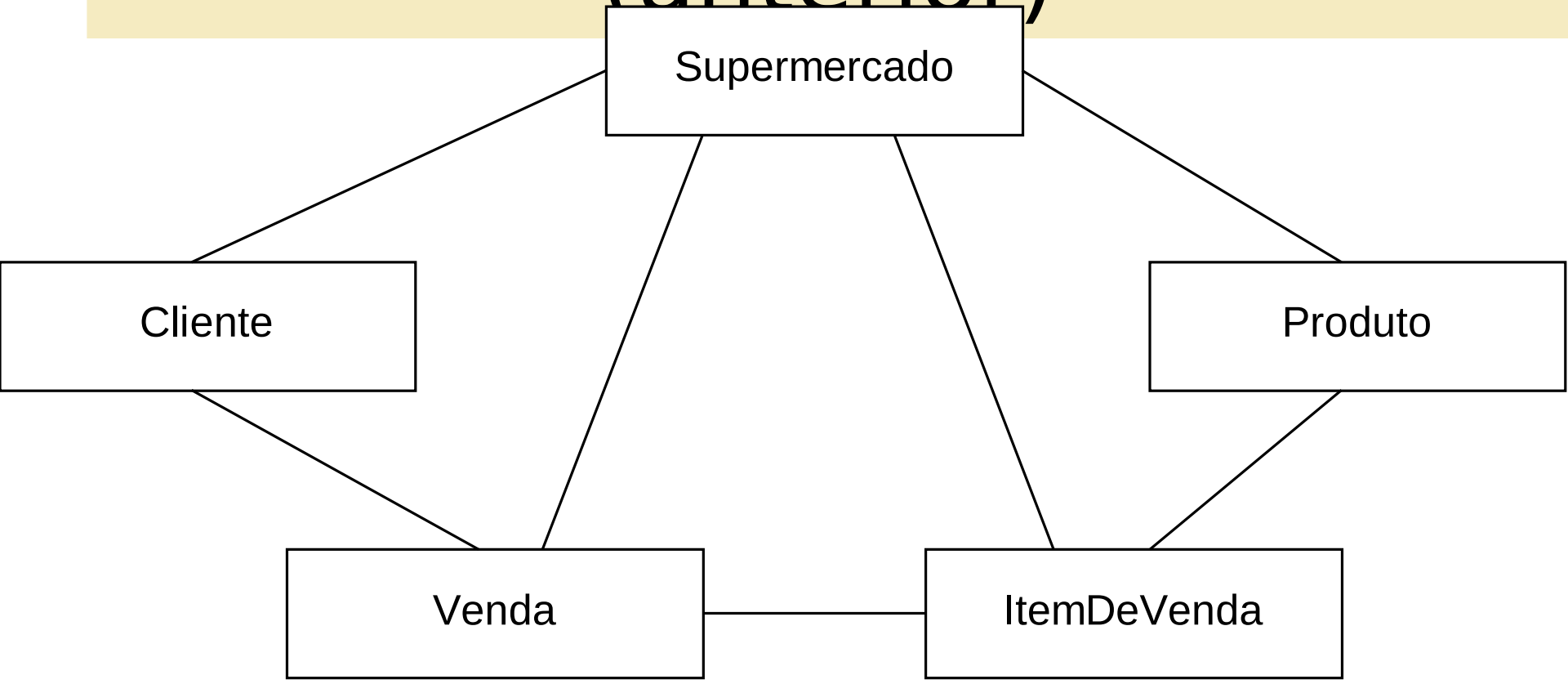
Acoplamento fraco

- Evite a criação de novos acoplamentos

Acoplamento definido pelo modelo conceitual e mantido pelo pseudocódigo que atende ao padrão especialista (anterior)



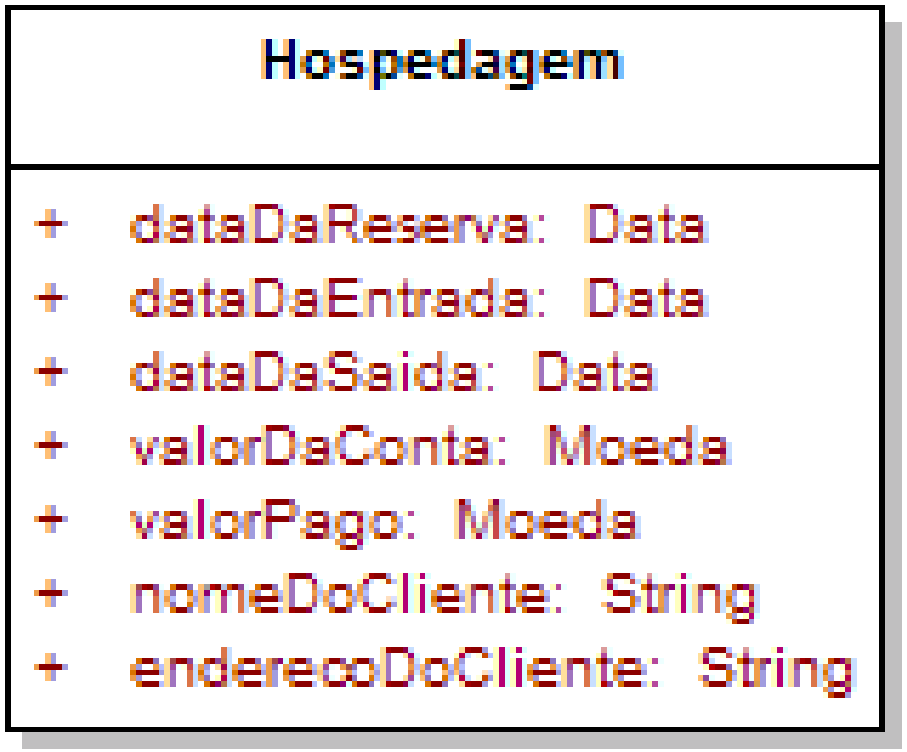
Acoplamento definido pelo pseudocódigo que não atende ao padrão especialista (anterior)



Coesão alta

- Verifique se o valor de algum atributo determina a possibilidade de outro atributo ser nulo ou não
- Verifique se existem subgrupos de atributos que estejam fortemente correlacionados
- Verifique se existem grupos de atributos que repetirão sempre os mesmos valores quando ocorrerem em diferentes instâncias

Exemplo de classe com baixa coesão



Solução com alta coesão