FUNDAÇÃO EDSON QUEIROZ
UNIVERSIDADE DE FORTALEZA
ENSINANDO E APRENDENDO

GRASP

Prof.: Americo Sampaio

Contém slides do material de Helder da Rocha (argonavis)

Introdução

- *Sistema Orientado a Objetos*
  - *Composto de objetos*
  - *Objetos enviam mensagens a outros objetos para completar operações*
- *Projeto de software*
  - *Atribuição de responsabilidades e interação entre objetos podem ter várias alternativas*
  - *Deve-se fazer boas escolhas*

Introdução

- *Más decisões de projeto geram*
  - *Sistemas difíceis de entender*
  - *Dificuldade de extender e fazer mudanças*
- *Padrões GRASP*
  - *Estabelecem princípios para atribuição de responsabilidades*
  - *Ajudam na criação de diagramas de interação*
    - *Seqüência ou colaboração*
  - *Ajudam a alocar responsabilidades para classes*

Introdução

- *Diagramas de interação são importantes nas fases de análise e projeto*
- *Atribuição de responsabilidades deve ser feita com cuidado*
- *Padrões e princípios ajudam na melhoria da qualidade do projeto*
- *Padrões de atribuição de responsabilidades (GRASP)*
  - *Servem para ajudar a tomar decisões de atribuir responsabilidades*

Responsabilidades

- *Booch e Rumbaugh "Responsabilidade é um contrato ou obrigação de um tipo ou classe."*
- *Dois tipos de responsabilidades dos objetos:*
  - *De conhecimento (knowing): sobre dados privativos e encapsulados; sobre objetos relacionados; sobre coisas que pode calcular ou derivar.*
  - *De realização (doing): fazer alguma coisa em si mesmo; iniciar uma ação em outro objeto; controlar e coordenar atividades em outros objetos.*
- *Responsabilidades são atribuídas aos objetos durante o design (projeto OO)*

Responsabilidades

- *Exemplo: Sistema de venda*
  - *Classe venda tem as seguintes responsabilidades*
  - *Fazer (realizar)*
    - *Imprimir a si própria*
  - *Conhecer*
    - *Conhecer sua data*
- *Responsabilidades de conhecer (Knowing)*
  - *Freqüentemente dedutíveis do modelo conceitual*
    - *Atributos e associações*

Responsabilidades e Métodos

- *A tradução de responsabilidades em classes e métodos depende da granularidade da responsabilidade*
  - *Ex: Fornecer acesso ao banco de dados*
    - *Dezenas de classes e métodos*
  - *Ex: Imprimir uma venda*
    - *Poucas classes e métodos*
- *Métodos são implementados para cumprir responsabilidades*
  - *Uma responsabilidade pode ser cumprida por um único método ou uma coleção de métodos trabalhando em conjunto*

Responsabilidades e Diagramas de Interação

- *Diagramas de interação mostram escolhas ao atribuir responsabilidades a objetos*
  - *No diagrama de colaboração ao lado objetos Order têm a responsabilidade de se prepararem: método prepare()*
  - *O cumprimento dessa responsabilidade requer colaboração com objetos Order Line e Stock Item*

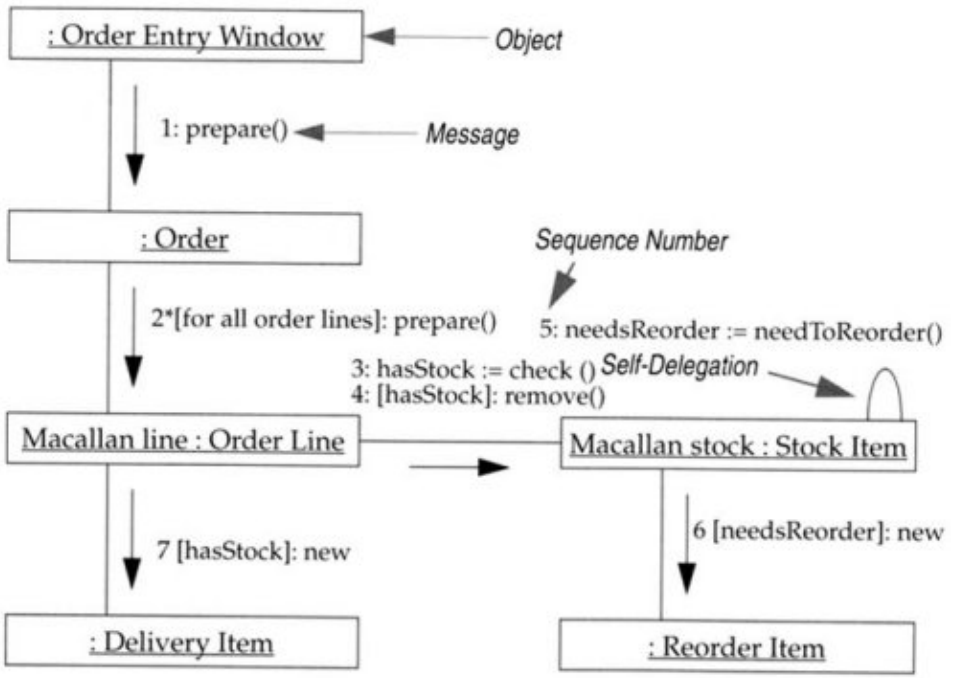


Figure 5-4: *Collaboration Diagram with Simple Numbering*

Padrões

- *Padrões são um repertório de soluções e princípios que ajudam os desenvolvedores a criar software e que são codificados em um formato estruturado consistindo de*
  - *Nome*
  - *Problema que soluciona*
  - *Solução do problema*
- *O objetivo dos padrões é codificar conhecimento existente de uma forma que possa ser reaplicado em contextos diferentes*

Padrões

- *Usualmente não contém novas idéias*
  - *Codifica princípios e conhecimento existente na prática*
- *Padrões tem nomes*
  - *Nomes devem ser sugestivos para o que o padrão representa e se propõe a resolver*
    - *Ex: Expert, Creator, Controller, etc.*
- *Nomear padrões facilita a comunicação*

Padrões GRASP

- *Introduzidos por Craig Larman em seu livro "Applying UML and Patterns"*
- *GRASP: **G**eneral **R**esponsibility and **A**ssignment **S**oftware **P**atterns*
  - *Princípios Gerais para Atribuição de Responsabilidades*
- *Os padrões GRASP descrevem os princípios fundamentais da atribuição de responsabilidades a objetos, expressas na forma de padrões*
- *Esses padrões exploram os princípios fundamentais de sistemas OO*
  - *padrões fundamentais*
  - *padrões avançados*

Padrões GRASP

Padrões básicos

- *Information Expert*
- *Creator*
- *High Cohesion*
- *Low Coupling*
- *Controller*

Padrões avançados

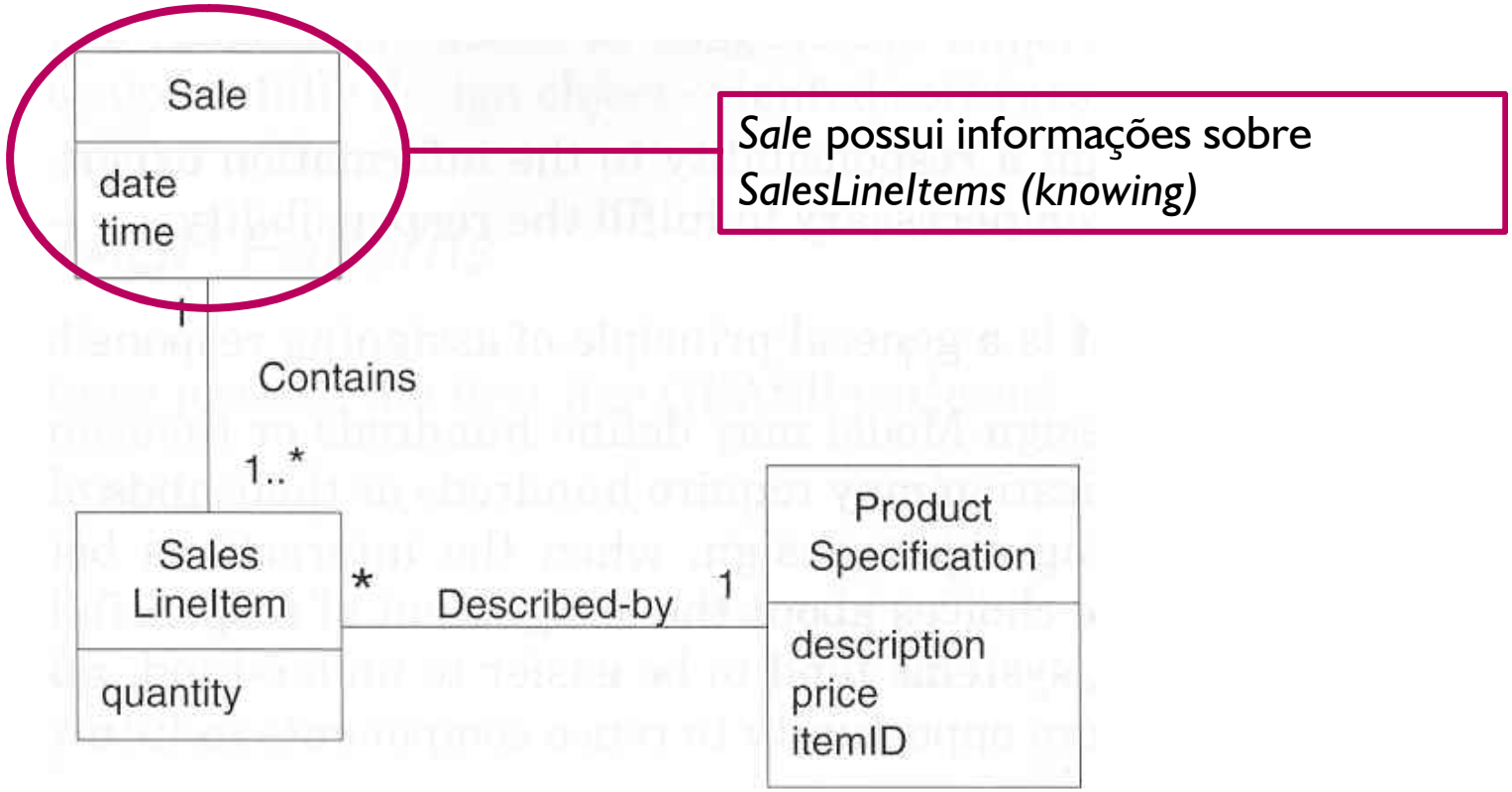
- *Polymorphism*
- *Pure Fabrication*
- *Indirection*

Expert (especialista de informação)

- *Problema*
  - *Precisa-se de um princípio geral para atribuir responsabilidades a objetos*
  - *Durante o design, quando são definidas interações entre objetos, fazemos escolhas sobre a atribuição de responsabilidades a classes*
- *Solução*
  - *Atribuir uma responsabilidade ao especialista de informação: classe que possui a informação necessária para cumpri-la*
  - *Comece a atribuição de responsabilidades ao declarar claramente a responsabilidade*

Expert

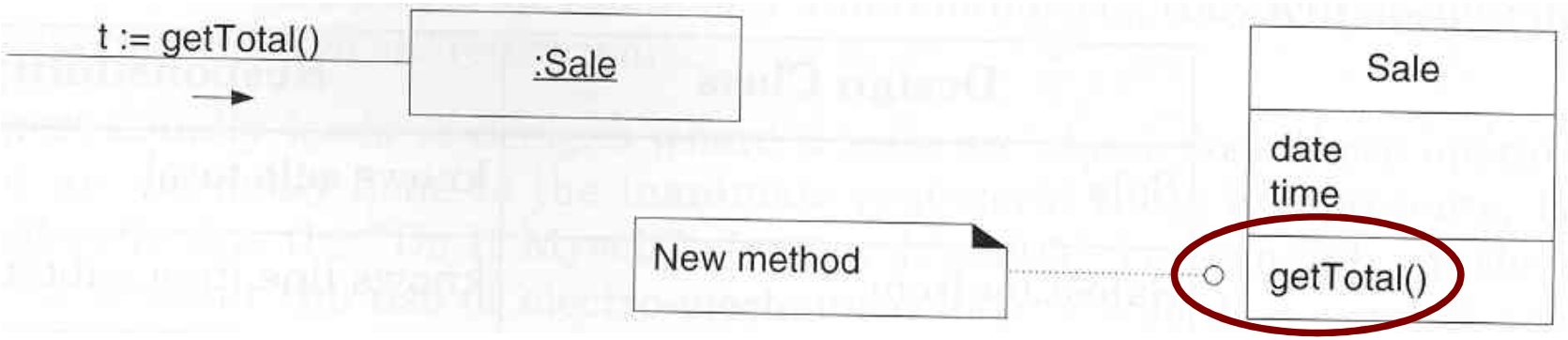
- *No sistema abaixo, uma classe precisa saber o total geral de uma venda (Sale). Que classe deve ser a responsável?*



Fonte: Livro Larman

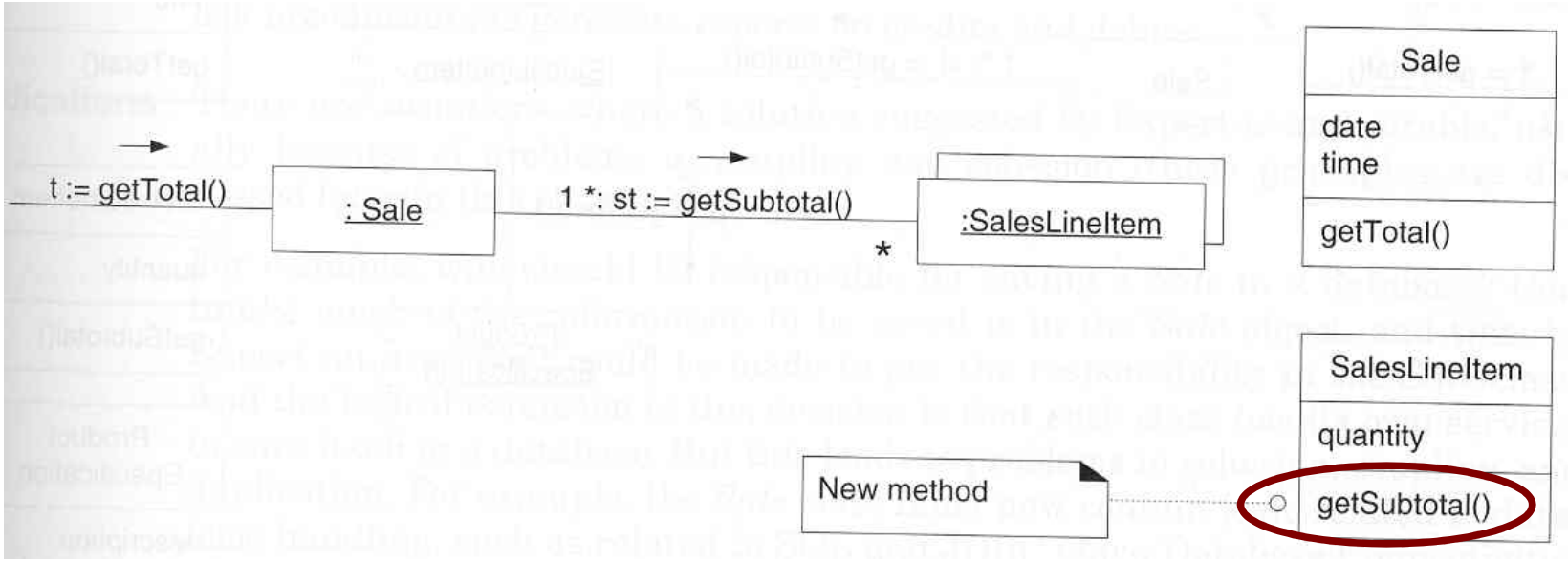
Expert

- *A nova responsabilidade é conduzida por uma operação no driagrama de interação*
- *Um novo método é criado*



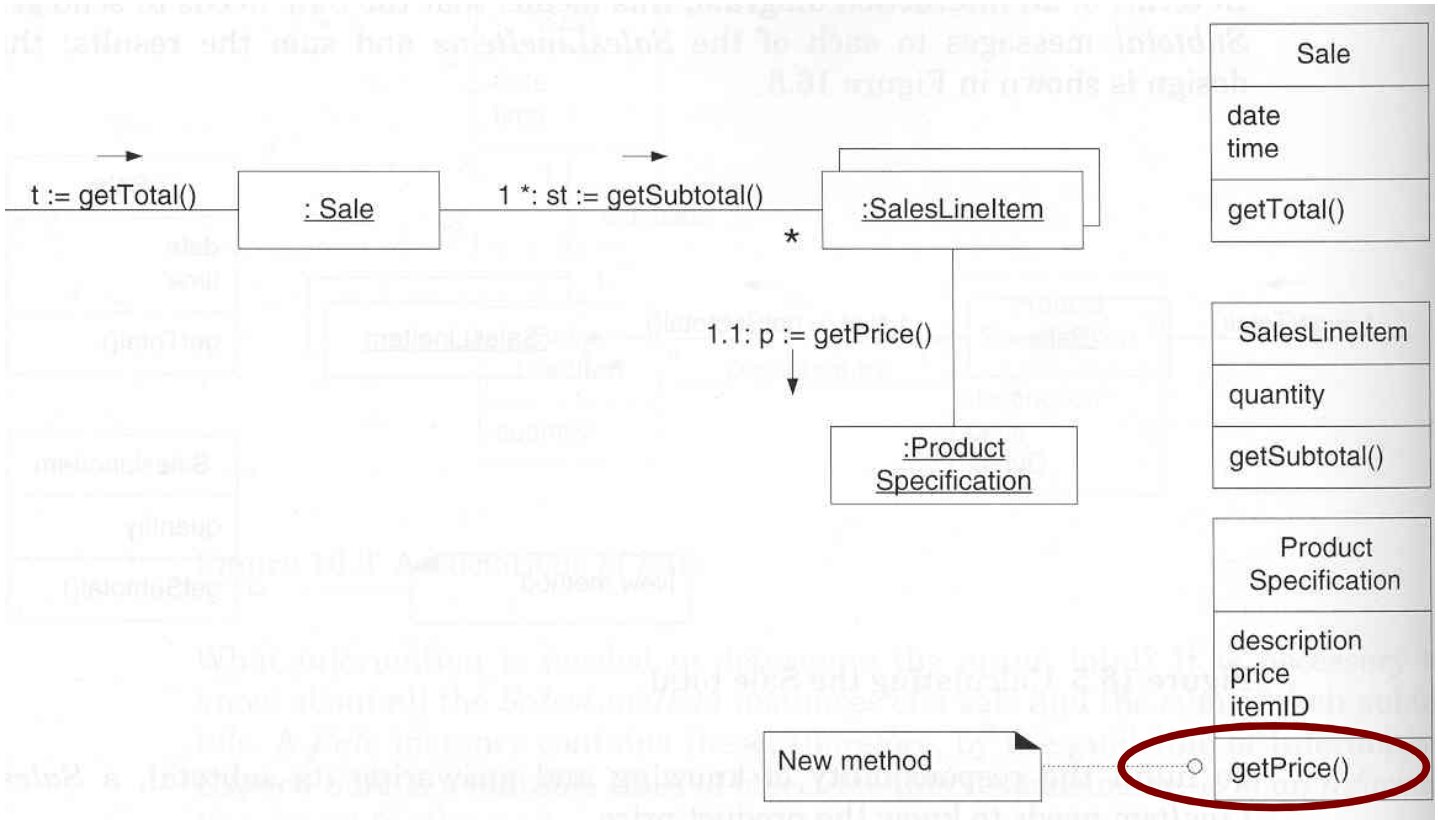
Expert

- *Mas como a classe Sale vai calcular o valor total?*
  - *Somando os subtotais. Mas quem faz isto?*
- *A responsabilidade para cada subtotal é atribuída ao objeto* SalesLineItem (item de linha do pedido)



Expert

- *O subtotal depende do preço. O objeto ProductSpecification é o especialista que conhece o preço, portanto a responsabilidade é dele.*



Expert

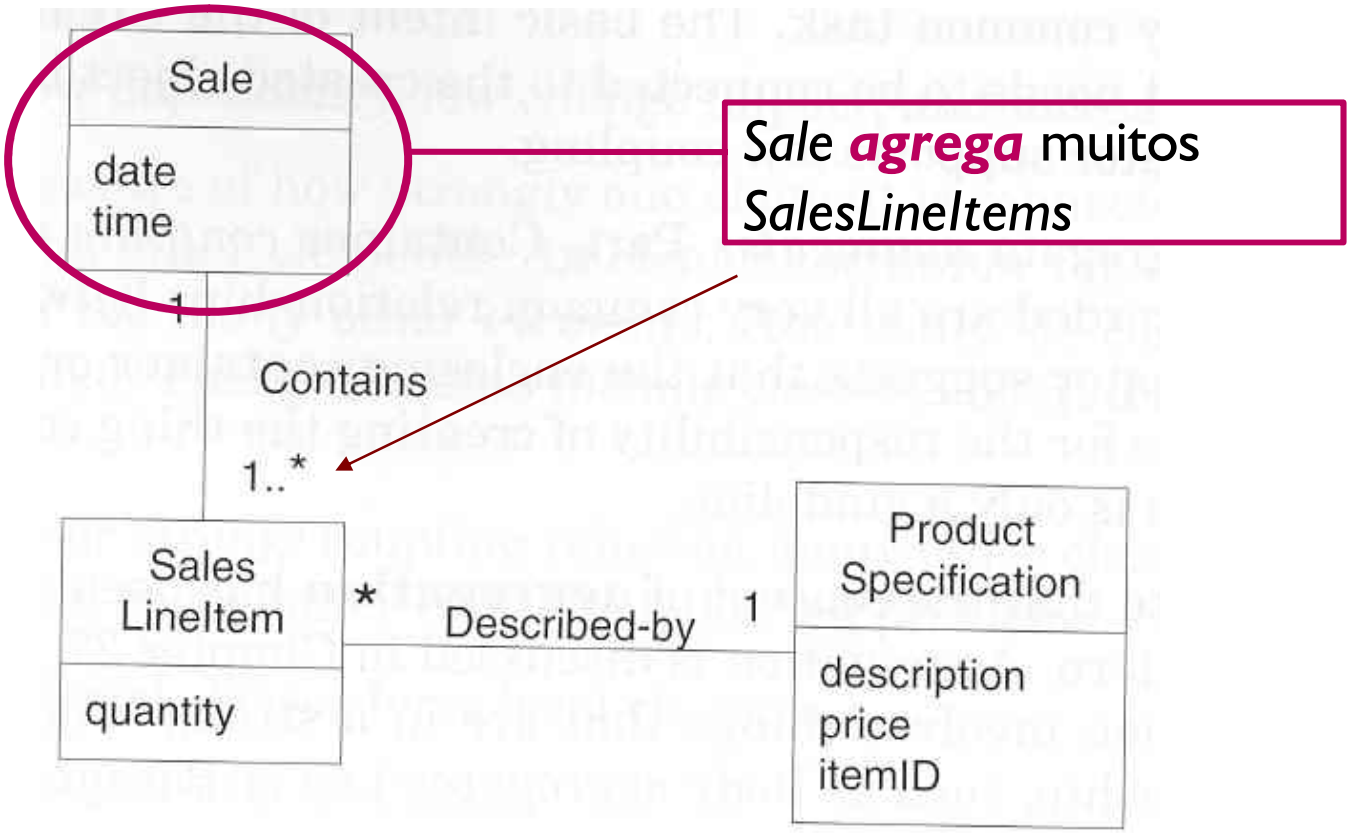
- *Para satisfazer a responsabilidade de informar o total de venda*
  - *3 responsabilidades foram atribuídas para 3 classes*
  - *Venda: conhece o total de venda*
  - *LinhaItemVenda: conhece o subtotal de cada item*
  - *Produto: conhece o preço do produto*

Creator

- ***Problema***: *Que classe deve ser responsável pela criação de uma nova instância de uma classe?*
- ***Solução***: *Atribua a B a responsabilidade de criar A se:*
  - *B* agrega *A objetos*
  - *B* contém *A objetos*
  - *B* guarda instâncias de *A objetos*
  - *B* possui dados para inicialização *que será passado para A quando ele for criado.*

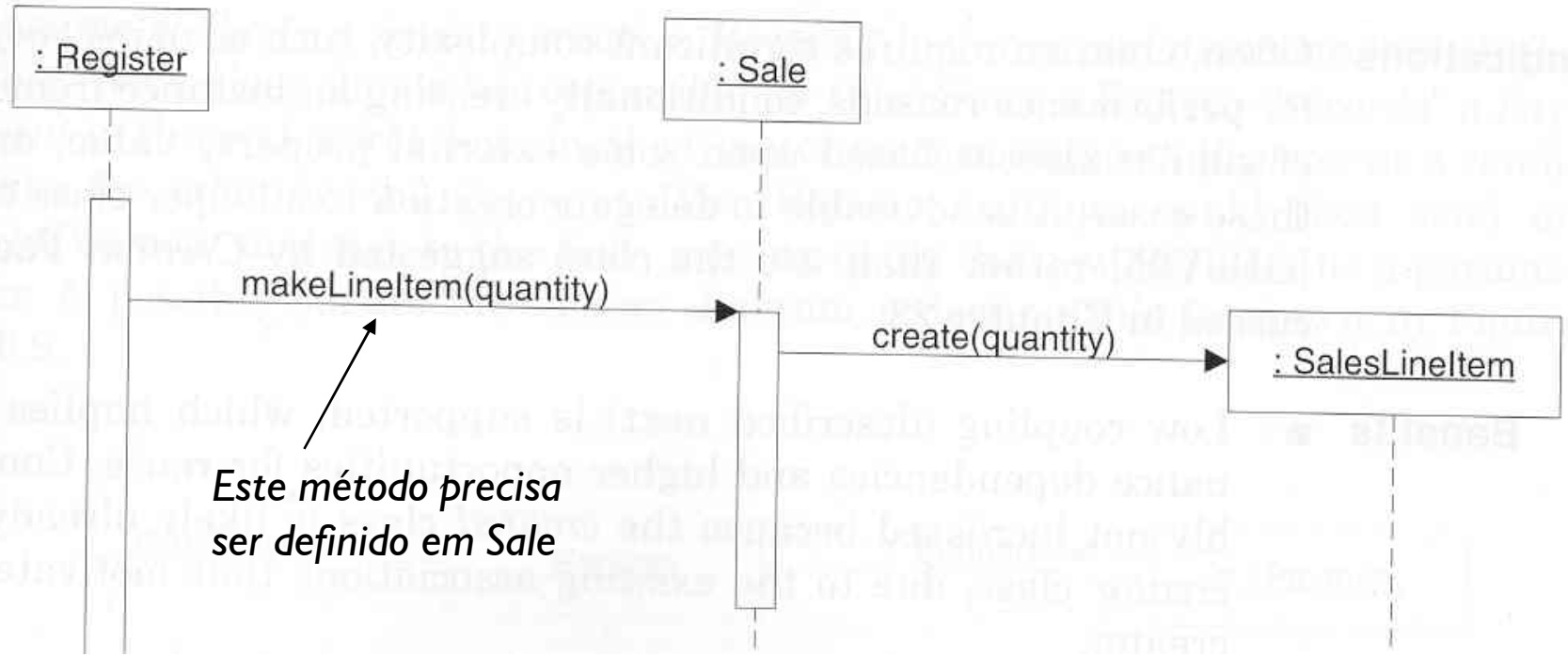
Creator

- *Que classe deve ser responsável por criar uma instância do objeto SalesLineItem abaixo?*



Creator

- *A nova responsabilidade é conduzida por uma operação em um diagrama de interações*
  - *Um novo método é criado na classe de design para expressar isto.*



Exercícios

- *1) Implemente em Java o código correspondente a responsabilidade de calcular o total da venda*
- *2) Implemente em Java o código correspondente a responsabilidade de criar items de venda*

Leitura Sugerida

- *Livro: Utilizando UML e Padrões, Craig Larman,*
  - *Ler capítulo 18 até pagina 201*