


Projeto Orientado a Objetos


Diagrama de Classes

Engenharia de Software II




Projeto OO

- Nesta fase é desenvolvida uma solução lógica baseada no paradigma de orientação a objetos – objetos, mensagens, classes, métodos,
 - “Fazer Certo a Coisa” – projetar de maneira competente uma solução que satisfaça os requisitos
- Os dois artefatos principais a serem desenvolvidos são:
 - Diagramas de Interação
 - Diagramas de Classe de Projeto




Classes de Análise para Classes de Projeto

- Modelo Conceitual ⇒ abstrações de conceitos, ou objetos, do mundo real
 - conceitos são também chamados de classes conceituais
- Diagrama de Classes de Projeto ⇒ definição de classes como componentes de software
 - classes de software
- Na prática, o Diagrama de Classes pode ser construído à medida que a fase de projeto avança, a partir dos Diagramas de Colaboração




Classes de Análise para Classes de Projeto

- Diagrama de Classes de Projeto ⇒ refinamento e transformações
 - classes de fronteiras
 - classes de controle
 - classes de entidade (negócio)



Classes de Análise para Classes de Projeto

- Classes de fronteiras <<boundary>>
 - Comunicação entre o sistema e os atores
 - Realizam a interface com o usuário, interface com sistemas externos, comunicação com dispositivos do sistema
 - Ex: *FormulárioInscrição, LeitoraCartões, SistemaFaturamento*



Classes de Análise para Classes de Projeto

- Classes de Controle <<control>>
 - Ponte de comunicação entre classe de fronteira e classe de entidade
 - Decidem o que o sistema deve fazer quando ocorre um evento externo relevante
 - Ex: *ControladorInscrição, GerenciadorContas, ControladorReservasCarros*

Classes Análise --> Classes de Projeto Classes de Fronteira

- Durante a análise, considera-se que há uma única classe de fronteira para cada ator.
- Algumas dessas classes podem resultar em várias outras.
 - classes de interface com usuário
 - classes de interface com equipamentos
 - classes de interface com outros sistemas
 - Ex: Equipamentos: uma ou mais classes para encapsular o protocolo de comunicação do equipamento.

Classes Análise --> Classes de Projeto Classes de Entidade

- A maioria permanece na passagem da análise para o projeto.
- As classes de entidades que devem ser armazenadas de modo persistente devem ser identificadas.
 - Classes persistentes e transientes
- Analisar relacionamentos entre as classes
 - visibilidade, navegabilidade, multiplicidade

Classes Análise --> Classes de Projeto Classes de Controle

- No refinamento, as classes de controle devem estar envolvidas com a coordenação
 - realização é responsabilidade das classes de entidade
 - Ex. coordenação:
 - preenchimento de controle das interfaces
 - autenticação de usuários
 - controle de acesso a funcionalidades do sistema
- Uma classe de controle de análise pode resultar em duas ou mais classes de projeto
 - Alta coesão

Classes Análise --> Classes de Projeto Outras Classes

- Em alguns casos, torna-se necessário acrescentar classes relacionadas a requisitos não funcionais
 - distribuição do sistema e comunicação de suas partes
 - segurança, autenticação, autorização
 - controle de transações, registro de operações realizadas
 - armazenamento persistente da informação

Classes Análise --> Classes de Projeto Outras Classes

- Pode utilizar classes pré-existentes
 - biblioteca de classes
 - coleção de classes com objetivo específico
 - JDBC e Swing
 - padrões de projeto
 - descrição de uma solução para um problema recorrente
 - classe DAO (Data Access Object), VH (View Helpers)
 - frameworks
 - semelhante a um padrão, mas apresenta implementação
 - JUnit

Criando Diagramas de Classes

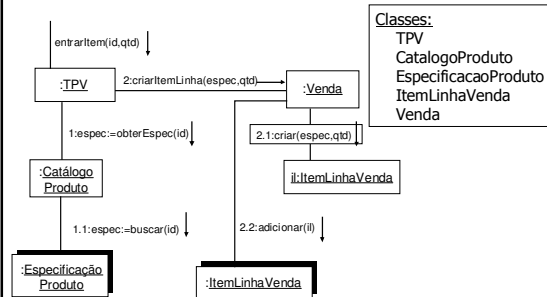
1. Identificar classes a partir dos diagramas de colaboração e ilustrá-las
2. Acrescentar os atributos identificados no Modelo Conceitual
3. Acrescentar os métodos provenientes dos diagramas de colaboração
4. Acrescentar tipos de atributos, parâmetros e retornos de métodos
5. Acrescentar as associações e navegabilidade (visibilidade por atributo)
6. Indicar relacionamentos de dependência (visibilidade não implementada por atributo)

Criando Diagramas de Classes

1. Identificar classes a partir dos diagramas de colaboração e ilustrá-las
2. Acrescentar os atributos identificados no Modelo Conceitual
3. Acrescentar os métodos provenientes dos diagramas de colaboração
4. Acrescentar tipos de atributos, parâmetros e retornos de métodos
5. Acrescentar as associações e navegabilidade (visibilidade por atributo)
6. Indicar relacionamentos de dependência (visibilidade não implementada por atributo)

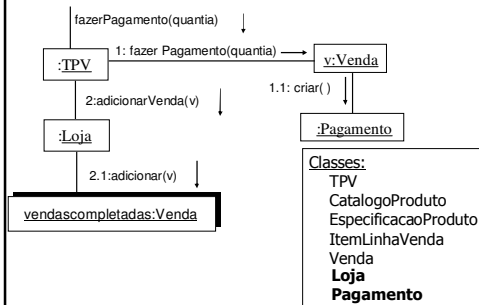
Identificando Classes...

- Diagramas de colaboração:



Identificando Classes...

- Diagramas de colaboração:



Criando Diagramas de Classes

1. Identificar classes a partir dos diagramas de colaboração e ilustrá-las
2. Acrescentar os atributos identificados no Modelo Conceitual
3. Acrescentar os métodos provenientes dos diagramas de colaboração
4. Acrescentar tipos de atributos, parâmetros e retornos de métodos
5. Acrescentar as associações e navegabilidade (visibilidade por atributo)
6. Indicar relacionamentos de dependência (visibilidade não implementada por atributo)

Conceitos (Modelo Conceitual)

TPV	CatálogoProdutos	Loja	EspecificaçãoProduto
		endereço nome	descrição preço CUP

Classes de software (Diagrama de Classes de Projeto)

TPV	CatálogoProdutos	Loja	EspecificaçãoProduto
		endereço nome	descrição preço CUP

Conceitos (Modelo Conceitual)

Pagamento	Venda	ItemLinhaVenda
quantia	data hora	quantidade

Classes de software (Diagrama de Classes de Projeto)

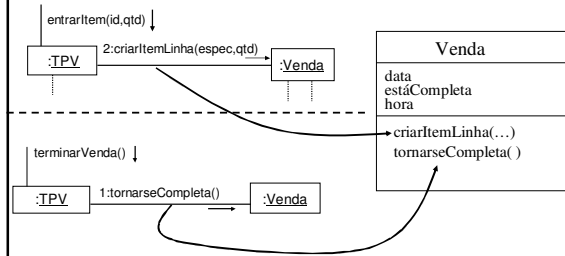
Pagamento	Venda	ItemLinhaVenda
quantia	data hora estaCompleta	quantidade

identificado durante a criação dos diagramas de colaboração

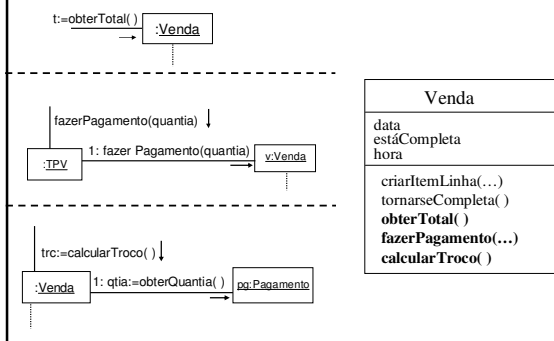
Criando Diagramas de Classes

1. Identificar classes a partir dos diagramas de colaboração e ilustrá-las
2. Acrescentar os atributos identificados no Modelo Conceitual
3. Acrescentar os métodos provenientes dos diagramas de colaboração
4. Acrescentar tipos de atributos, parâmetros e retornos de métodos
5. Acrescentar as associações e navegabilidade (visibilidade por atributo)
6. Indicar relacionamentos de dependência (visibilidade não implementada por atributo)

Acrescentando métodos...



Acrescentando métodos...

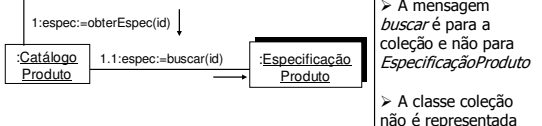


Cuidados com a inserção de métodos

- Linguagens de programação distintas podem ter sintaxes distintas para métodos
 - recomendável: usar sintaxe básica UML
nomeMétodo(Par₁, Par₂, ... Par_n)
- Interpretação do método **criar()**: dependente da linguagem de programação

Cuidados com a inserção de métodos (cont.)

- Mensagens para multiobjetos: não incluir métodos correspondentes a mensagens enviadas a classe coleção
 - tratamento da mensagem depende da implementação - a coleção pode ser representada, por exemplo, por um set em C++



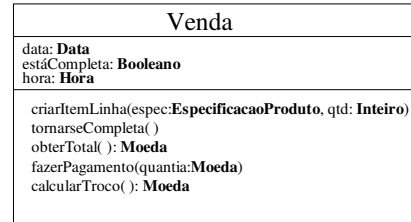
Criando Diagramas de Classes

1. Identificar classes a partir dos diagramas de colaboração e ilustrá-las
2. Acrescentar os atributos identificados no Modelo Conceitual
3. Acrescentar os métodos provenientes dos diagramas de colaboração
4. Acrescentar tipos de atributos, parâmetros e retornos de métodos
5. Acrescentar as associações e navegabilidade (visibilidade por atributo)
6. Indicar relacionamentos de dependência (visibilidade não implementada por atributo)

Acrescentando tipos...

- Os tipos dos atributos e métodos podem, opcionalmente, ser acrescentados aos diagramas de classes de projeto
 - se uma ferramenta CASE for utilizada para geração automática de código, os tipos detalhados são necessários
 - se o diagrama for usado exclusivamente por desenvolvedores de software, o excesso de informação pode "poluir" o diagrama e dificultar seu entendimento
 - incluir informação de tipo quando relevante/necessária

Exemplo



Criando Diagramas de Classes

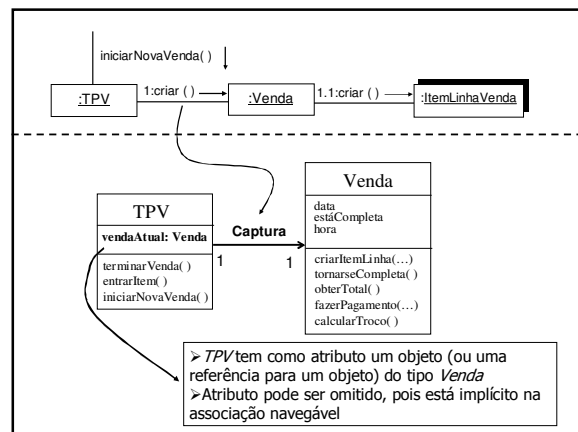
- Identificar classes a partir dos diagramas de colaboração e ilustrá-las
- Acrescentar os atributos identificados no Modelo Conceitual
- Acrescentar os métodos provenientes dos diagramas de colaboração
- Acrescentar tipos de atributos, parâmetros e retornos de métodos
- Acrescentar as associações e navegabilidade (visibilidade por atributo)
- Indicar relacionamentos de dependência (visibilidade não implementada por atributo)

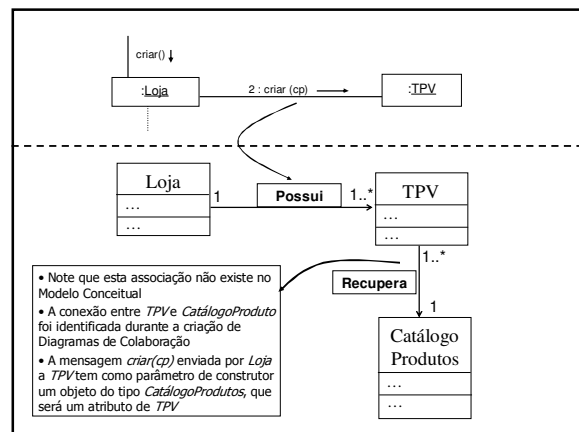
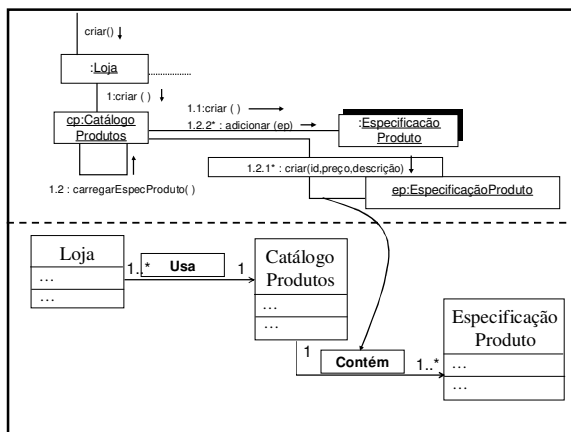
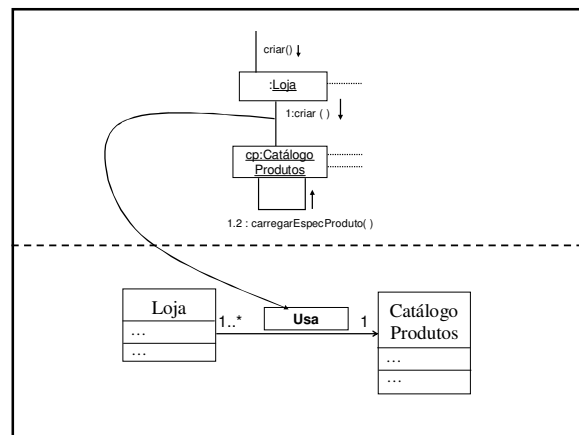
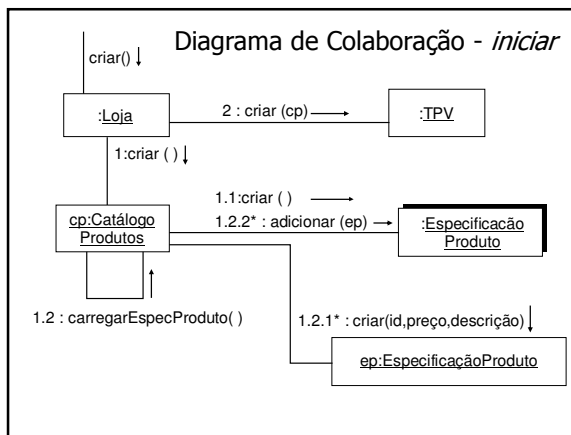
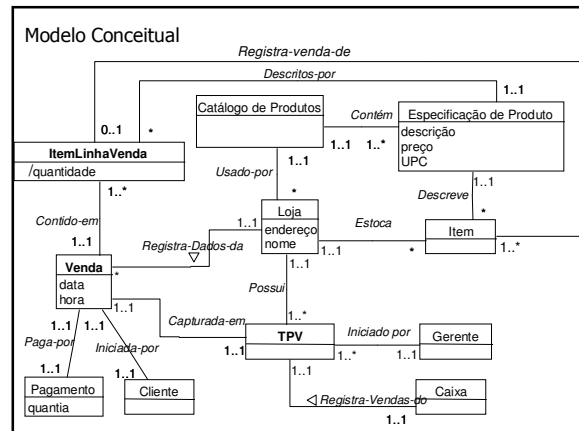
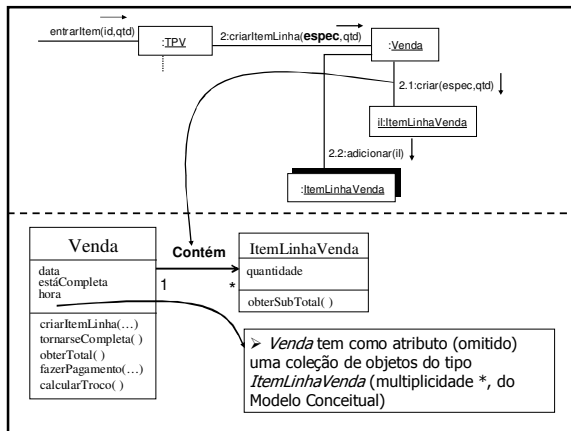
Associações e Navegabilidade

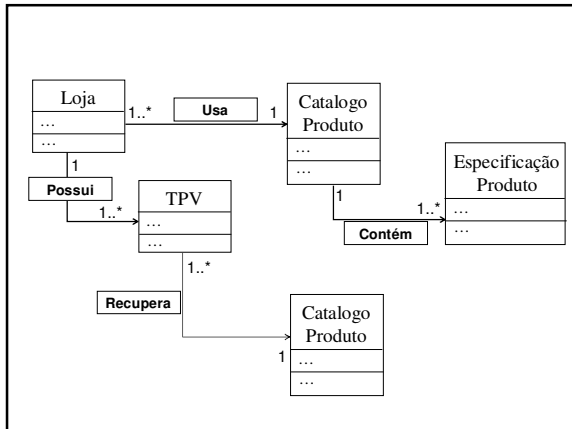
- Associações e navegabilidade entre classe são indicadas pelos diagramas de colaboração
 - Navegabilidade indica possibilidade de navegação unidirecional por meio de uma associação entre classes
 - geralmente implica visibilidade por atributos
- A multiplicidade e os nomes das associações são retirados do Modelo Conceitual
- Notação: seta contínua

Associações e Navegabilidade

- Indícios de associação e com presença de navegabilidade:
 - A envia mensagem para B
 - A cria B
 - A precisa manter uma conexão com B





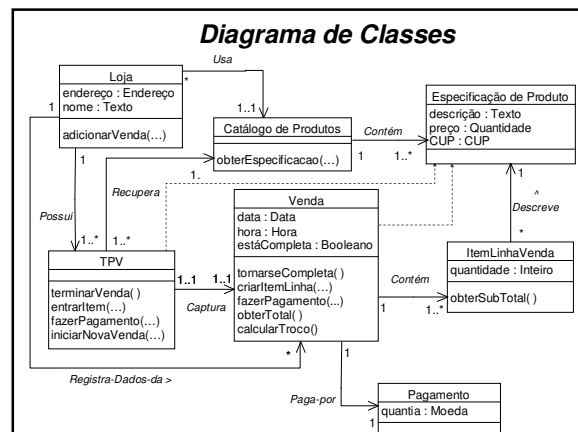
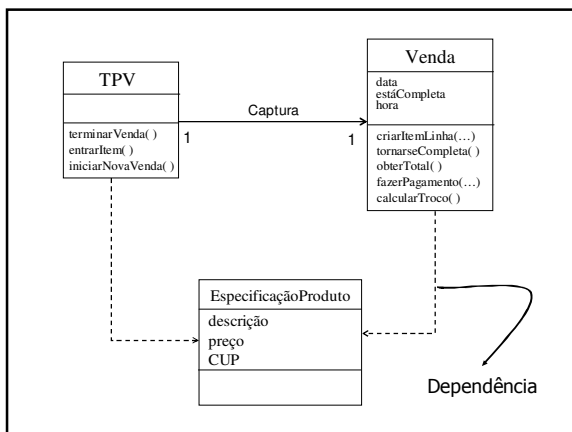
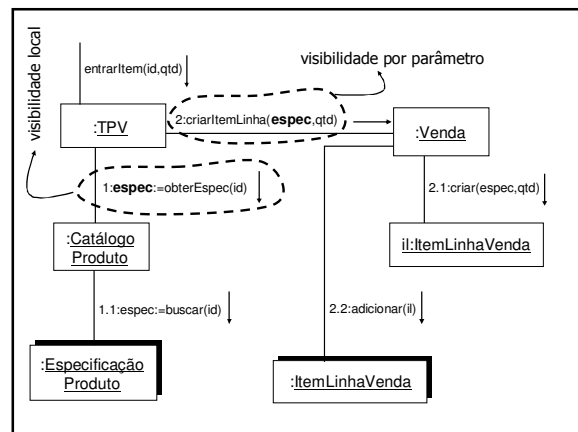


Criando Diagramas de Classes

1. Identificar classes a partir dos diagramas de colaboração e ilustrá-las
2. Acrescentar os atributos identificados no Modelo Conceitual
3. Acrescentar os métodos provenientes dos diagramas de colaboração
4. Acrescentar tipos de atributos, parâmetros e retornos de métodos
5. Acrescentar as associações e navegabilidade (visibilidade por atributo)
6. Indicar relacionamentos de dependência (visibilidade não implementada por atributo)

Relacionamentos de Dependência

- No Diagrama de Classes, o relacionamento de dependência representa a visibilidade entre classes que não é implementada por atributo
 - visibilidade por parâmetro
 - visibilidade local ou global
- Um objeto de uma classe A tem conhecimento (enxerga) um objeto da classe B
- Notação: seta tracejada



Notação para detalhes dos métodos e atributos - UML

