

Diagrama de Classes

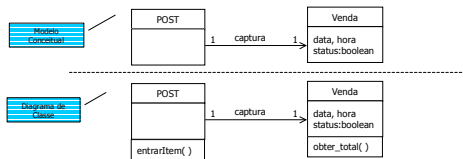
Modelo de classes de especificação
Perspectiva de Projeto

Diagrama de Classes

- Ilustra as especificações de software para as classes e interfaces do sistema.
- É obtido através da adição de detalhes ao modelo conceitual conforme a solução de software escolhida, acrescentando-se:
 - classes de controle e de interface (fronteira);
 - métodos (tipos de parâmetros e de retorno).
 - visibilidade e navegabilidade,
 - inclui dependências,
 - inclui a visão de projeto além da visão de domínio

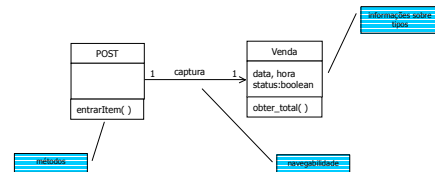
Modelo de Conceitual X Diagrama de Classe

- **Modelo Conceitual:** abstração de conceitos do mundo real
- **Diagrama de Classe:** especificação de componentes de software



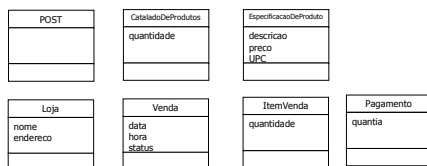
Diagramas de Classe

- Diagrama parcial para as classes **POST** e **Venda** no sistema POST:



POST Criando Diagramas de Classe

- Identificando classes e atributos
 - Observar os DI e Modelo Conceitual



Como construir (2)

- identificar todas as classes participantes da solução através dos diagramas de interação
- desenhar o diagrama
- copiar os atributos
- adicionar os métodos dos diagramas de interação
- adicionar tipos de atributos, parâmetro e retornos de métodos.

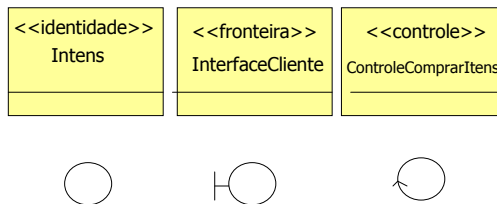
Como construir (2)

- adicionar as associações necessárias a visibilidade
- adicionar navegabilidade que indica a direção de visibilidade por atributo
- adicionar setas pontilhadas indicando visibilidade que não é por atributo
- Métodos ``create" e de acessos aos atributos podem ser omitidos.
- Os tipos podem ou não ser mostrados; classes podem ser detalhadas ao máximo

Identificação de Classes com Estereótipos

- Estereótipo é um classificador. Tipos:
- **Entidade:** representam conceitos do mundo real e armazenam dados que os identificam – como no modelo conceitual
- **Controle:** controlam a execução de processos e o fluxo de execução de todo ou de parte de casos de uso e comandam outras classes
- **Fronteira:** realizam o interfaceamento com entidades externas (atores), contém o protocolo de comunicação com impressora, monitor, teclado, disco, porta serial, modem, etc.

Exemplos no Sistema Posto Comercial



Identificação das Classes de Controle

- Definir pelo menos uma classe do tipo controle para cada caso de uso de forma que ela contenha a descrição e comando do processo associado ao caso de uso.
- Definir classes de controle auxiliares em certos casos de uso que devido à complexidade requeiram a divisão de seu processo em subprocessos. As classes auxiliares seriam controladas pela classe de controle principal

Identificação das Classes de Controle

- Suas principais características são:
 - Cria, ativa e anula objetos controlados
 - Controla a operação de objetos controlados
 - Controla a concorrência de pedidos de objetos controlados
 - Em muitos casos corresponde a implementação de um objeto intangível.
- Gerente de Registro para o Caso de Uso registrarAlunos

Identificação das Classes de Fronteira

- Definir uma classe do tipo fronteira para cada ator que participe do caso de uso, pois cada ator que pode exigir um protocolo próprio para comunicação.
- Uma classe para interface com o usuário, uma classe para interface com a impressora, uma classe para interface com a porta serial, etc.

Identificação das Classes de Fronteira

- Exemplos: Interface tipo Janela, Protocolo de Comunicação, Interface de Impressão, Sensores, etc.
- Classes: Formulário em Branco e Sistema de Cobrança

Identificação dos Métodos

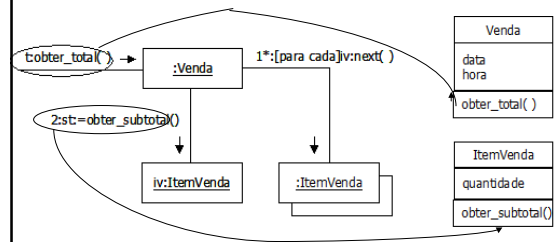
- Os métodos são acrescentados nas perspectivas de especificação e de implementação e são derivados a partir dos diagramas de interação: colaboração e sequências, na fase de projeto detalhado.
- É útil distinguir operações de métodos. Operações é algo que se evoca sobre um objeto (a chamada do procedimento). Para realizar uma operação a classe implementa um método (o corpo do procedimento)

Identificação dos Métodos

- É útil distinguir operações que alteram ou não o estado (atributos) de uma classe
- Não alteram: query, métodos de obtenção, getting methods
- Alteram: modificadores, métodos de atribuição ou fixação, setting methods

Métodos a partir dos DI

- Se uma classe recebe uma mensagem A, ela deverá executar em resposta uma operação A que é implementada por um método A.



Métodos

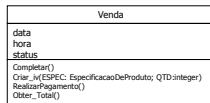
- Os métodos são acrescentados na perspectiva de implementação e são derivados a partir dos diagramas de interação: colaboração e sequências.
- É útil distinguir operações de métodos. Operações é algo que se evoca sobre um objeto (a chamada do procedimento). Para realizar uma operação a classe implementa um método (o corpo do procedimento)

Métodos

- É útil distinguir operações que alteram ou não o estado (atributos) de uma classe
- Não alteram: query, métodos de obtenção, getting methods
- Alteram: modificadores, métodos de atribuição ou fixação, setting methods

Criando Diagramas de Classe para o Sistema POST

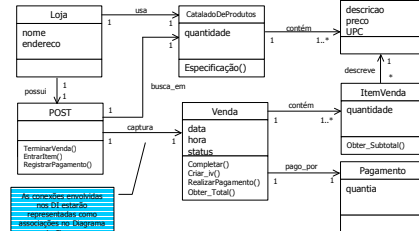
- Adicionando informação sobre o tipo dos atributos
 - Opcional
 - Grau de detalhe dependente do público-alvo.



Criando Diagramas de Classe para o Sistema POST

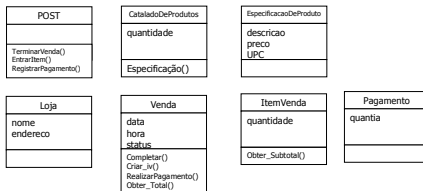
- Adicionando associações e navegabilidade

– Investigar os DI



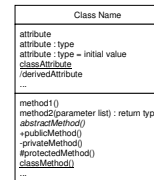
Criando Diagramas de Classe para o Sistema POST

- Adicionando nomes aos métodos
 - Observe as mensagens dos DI



Características dos Elementos de Classe

- UML oferece notação rica para descrever características como visibilidade, valores iniciais, etc.
- No sistema POST: todos os atributos são privados e todos os métodos são públicos



Outros aspectos

Visibilidade

Habilidade de um objeto A ver ou fazer referência a um outro objeto B. Para A enviar uma msg para B, B deve ser visível a A.

- por atributo: B é um atributo de A
- por parâmetro: B é um parâmetro de um método de A
- localmente declarada: B é um objeto local de um método de A; uma instância local é criada, ou ele será um valor de retorno
- global: B é visível globalmente; usa-se uma variável global para armazenar uma instância - pouco recomendada

Diagrama de Colaboração - Visibilidade

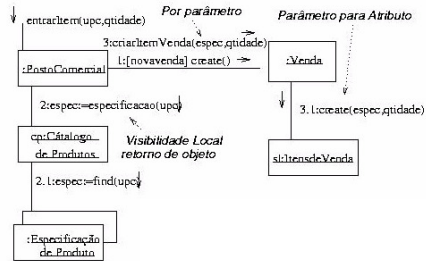


Diagrama de Colaboração - Visibilidade

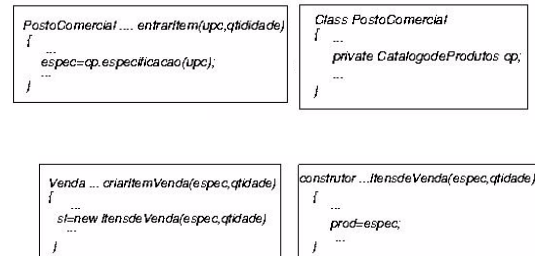


Diagrama de Colaboração - Inicializando

Como se realizam as operações iniciais da aplicação?

- Cria-se um caso de uso startUp.
- Seu diagrama de colaboração deve ser criado por último, representando o que acontece quando o objeto inicial do problema é criado.
- Quem deveria ser o objeto inicial do sistema?
 - classe representando toda a informação lógica do sistema
 - classe representando a organização
 - usar Padrões Alta Coesão e Baixo Acoplamento

Diagrama de Colaboração - Inicializando

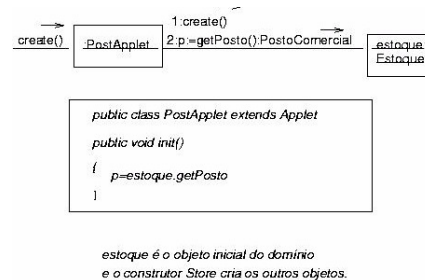


Diagrama de Colaboração - Inicializando

Criação dos Objetos Iniciais do Domínio

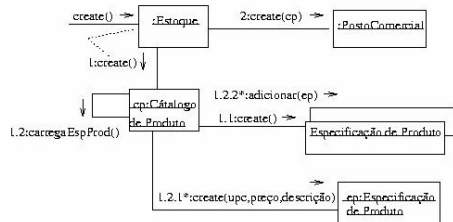


Diagrama de Colaboração - Inicializando

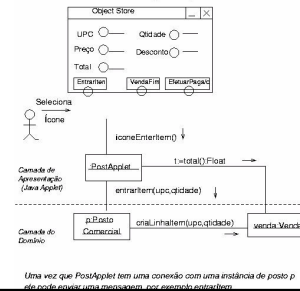
- Conectando a camada de apresentação com a do domínio
- Uma operação de startUp pode ser:
 - uma mensagem "create" para o objeto inicial;
 - se o objeto inicial é o controlador, uma mensagem "run" para um objeto inicial é enviada.

Diagrama de Colaboração - Inicializando

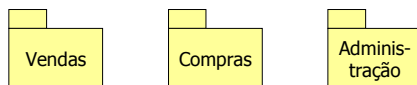
- Se uma interface do usuário estiver envolvida ela é responsável por iniciar a criação do objeto inicial e outros associados.
- Objetos da camada de apresentação não devem ter responsabilidades lógicas. Das nossas escolhas resultarão extensibilidade, clareza e manutenibilidade

Diagrama de Colaboração - Inicializando

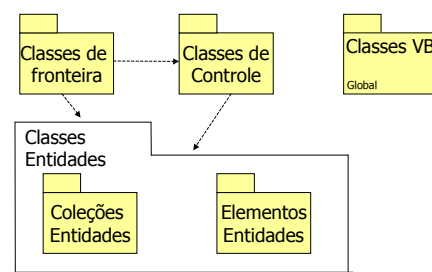
Conectando a Camada de Apresentação ao do Domínio



Organização de Classes em Pacotes Lógicos



Organização de Classes em Pacotes Lógicos



Referências

- Boock, G. and Rumbaugh, J. *The Unified Modeling Language User Guide*. Addison-Wesley, 1999
- Arlow, J. and Neustadt, I. *UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design*, 2nd Edition, The Addison-Wesley Object Technology Series, 2005.
- Rumbaugh, J.; Jacobson, I. and Boock, G. *The Unified Modeling Language Reference Manual*, 2nd Edition, The Addison-Wesley Object Technology Series, 2004.
- Boock, G.; Rumbaugh, J. and Jacobson, I. *Unified Modeling Language User Guide*, 2nd Edition, The Addison-Wesley Object Technology Series, 2005.
- Jacobson, I.; Boock, G. and Rumbaugh, J., *Unified Software Development Process*, Addison-Wesley, Janeiro 1999.
- Larman, C. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design* Prentice-Hall, New Jersey - USA, 1997
- Bezerra, E. *Princípios de Análise e Projeto com a UML*, ed. Campus-Elsevier, 2003.