Engenharia de Software Aula 09

Projeto de Software
Diagrama de Classes

Eduardo Figueiredo

http://www.dcc.ufmg.br/~figueiredo
dcc603@gmail.com

04 Abril 2011

Tópicos da Aula

- Revisão de orientação a objetos
- Projeto orientado a objetos
- Diagrama de Classes

Trabalho Prático (TP)

Diretrizes Gerais

- Grupo de até 5 integrantes
- Tarefas
  - Desenvolver um sistema de software
  - Simular um processo de software
  - Documentar todas as fases de desenvolvimento

Tema do Trabalho

- A escolha do grupo
  - Entretanto, no máximo dois grupos podem fazer o TP sobre um mesmo tema
- Sugestão de temas
  - Locadora, Biblioteca, Oficina de Eletro, Clínica Médica, Construtora, Veículos Policiais, Alarme de Fogo e Invasão, Pedágio, Urna Eletrônica, Copa de 2014.

Pontuação do TP

- 30 pontos, distribuídos entre
  - Modelo de processo e plano de trabalho
  - Documentação de requisitos
  - Projeto arquitetural
  - Projeto detalhado
  - Implementação
- Substitutiva
  - Prova individual envolvendo todo o conteúdo da disciplina
  - Não recomendado (queima cartucho)

Ferramentas Usadas

- Pode se usar qualquer ferramenta ou ambiente (IDE) para desenvolver o TP
- Entregar a documentação em DOC (MS Word 2003) ou PDF
- Entregar a implementação em um arquivo ZIP contendo o código *.JAVA

Apresentação Parcial

- Duração de 8 a 10 minutos
  - O tempo depende do número de grupos
- Andamento do trabalho até a data da apresentação
  - Modelo de processo e cronograma
  - Análise de requisitos
  - Decisões tomadas em relação a arquitetura e ao projeto

Datas Importantes

- Imediatamente (04/04/2012)
  - Início do trabalho
- 23/04/2012
  - Informar grupo e tema escolhido
- 30/05/2012
  - Apresentação parcial
- 15/06/2012
  - Entrega Final

Projeto Orientado a Objetos

Pensar Orientado a Objetos

- Onde quer que você olhe no mundo real, você vê objetos
  - Pessoas, animais, plantas, carros, etc.
- Humanos pensam em termos de objetos
  - Orientação a objetos é alto nível i.e., mais próximo dos humanos que dos computadores

Características de Objetos

- Classificação
  - Animados: possuem vida, se movem...
  - Inanimados: não possuem vida
- Objetos possuem **atributos**
  - Tamanho, forma, cor, peso, etc.
- Objetos exibem **comportamentos**
  - Uma bola rola, um avião voa
  - Uma pessoa anda, fala, pensa, etc.

Classe de Objetos

- Objeto é uma entidade que possui um estado e operações definidas sobre este estado
- Classe é um “esqueleto” para criação (instanciação) de objetos
  - Como a planta é um “esqueleto” para criação de casas

Definições

- Objeto
  - Entidade que descreve uma realidade
- Classe
  - Abstração que define objetos
- Instância
  - Criação de um objeto a partir de uma classe

Comunicação entre Objetos

- A comunicação pode ocorrer de várias formas
  - Envio de mensagens (exemplo, pode ser implementadas por arquivos XML)
  - Invocação de métodos remotos (RMI)
  - Chamada de métodos locais
- Veremos apenas chamada de métodos locais
  - Comunicação síncrona

Projeto Orientado a Objetos

- Maneira natural de visualizar o software
  - Documentação
  - Comunicação entre a equipe
- Modela o software semelhante ao mundo real - usando objetos
- **Objetos** são modelados em termos de seus **atributos** e comportamento (**métodos**)

Dos Requisitos ao Projeto

Desenvolvimento OO

- Análise orientada a objetos
  - Cria um modelo de objetos para o domínio da aplicação (domínio do problema)
- Projeto orientado a objetos
  - Cria um modelo de objetos para implementar requisitos (domínio da solução)
- Programação orientada a objetos
  - Implementa o projeto orientado a objetos usando uma linguagem de programação

Desenvolvimento OO

- A transição entre estágios deve ser contínua e com notações compatíveis
  - Da análise para o projeto
  - Do projeto para a programação

Vantagens de OO

- Facilidade de entendimento
  - Mapeamento de entidades do mundo real para objetos de sistema
- Facilidade de manutenção
  - Mais fácil de alterar pois os objetos são independentes
- Facilidade de reuso
  - Objetos são potencialmente componentes reusáveis

Atividades de Projetar OO

1. Definir o contexto do sistema
2. Projetar a arquitetura
3. Identificar os objetos principais
4. Desenvolver os modelos de projeto
5. Especificar interfaces entre objetos

Paralelo e Iterativo

- As atividades não necessariamente são sequenciais
- Geralmente é feito de forma iterativa
  - Define-se parte do contexto do sistema
  - Projeta-se parte da arquitetura
  - Identifica-se alguns objetos
  - Modela-se estes objetos
  - Define-se suas interfaces

Definir o contexto do sistema

- Objetivo: compreensão do software que está sendo desenvolvido e de seu ambiente externo
- Técnicas adotadas
  - Diagramas de Casos de Uso
  - Descrição dos Cenários
- Ao definir o contexto, pode-se identificar alguns objetos do domínio

Projetar Arquitetura

- Primeiro passo do projeto de sistema
- O projeto arquitetural envolve
  - Identificação dos componentes principais do sistema (sub-sistemas)
  - Definição das interfaces de comunicação entre os componentes
- Regra geral: modelar de 5 a 9 subsistemas

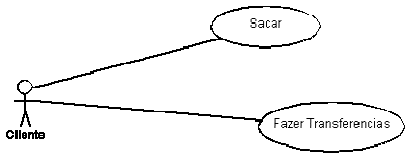
Identificar os objetos principais

- Identificação de objeto é um processo iterativo
  - É improvável que você faça certo na primeira vez
- Na verdade, identifica-se as classes de objetos
- Não há fórmula mágica para a identificação de objetos

Uma abordagem para identificação

- Análise gramatical baseada em
  - Descrição em linguagem natural do sistema
  - Descrição dos cenários de uso
- Como proceder
  - Substantivos são objetos ou atributos
  - Verbos são métodos
  - Refinar e definir novos objetos usando o conhecimento do domínio do sistema

Diagrama de Casos de Uso



Exemplo de Cenário

- Nome do Cenário: Sacar
- Ator: Cliente
- Pré-condição: Conta e senha validadas
- Fluxo normal
  1. Entrar com valor do saque
  2. Confirmar dados e operação
  3. Debitar valor da conta do cliente
- Fluxos alternativo: Saldo insuficiente
  3.1 Apresentar aviso ao cliente
- Pós-condição: Valor sacado é debitado do saldo do cliente

Exemplo de Cenário

- Nome do Cenário: Sacar
- Ator: Cliente
- Pré-condição: Conta e senha validadas
- Fluxo normal
  1. Entrar com valor do saque
  2. Confirmar dados e operação
  3. Debitar valor da **conta** do **cliente**
- Fluxos alternativo: Saldo insuficiente
  3.1 Apresentar aviso ao cliente
- Pós-condição: Valor sacado é debitado do saldo do cliente

Potenciais objetos do sistema

Exemplo de Cenário

- Nome do Cenário: Sacar
- Ator: Cliente
- Pré-condição: Conta e senha validadas
- Fluxo normal
  1. Entrar com **valor** do saque
  2. Confirmar **dados** e operação
  3. Debitar **valor** da conta do cliente
- Fluxos alternativo: Saldo insuficiente
  3.1 Apresentar aviso ao cliente
- Pós-condição: Valor sacado é debitado do saldo do cliente

Potenciais atributos dos objetos

Exemplo de Cenário

- Nome do Cenário: Sacar
- Ator: Cliente
- Pré-condição: Conta e senha validadas
- Fluxo normal
  1. **Entrar com valor** do saque
  2. Confirmar dados e operação
  3. **Debitar valor** da conta do cliente

Potenciais métodos dos objetos

- Fluxos alternativo: Saldo insuficiente
  3.1 Apresentar aviso ao cliente
- Pós-condição: Valor sacado é debitado do saldo do cliente

Modelos de projeto

- Fazem a ligação entre requisitos (problema) e implementação (solução)
- Mostram os objetos ou as classes de objetos e os relacionamentos entre essas entidades
- Devem incluir detalhes suficientes para os programadores tomarem decisões

Várias visões

- Para evitar modelos complexos, eles são quebrados em diversas visões
  - Modelos estáticos descrevem a estrutura estática das classes
  - Modelos dinâmicos descrevem as interações dinâmicas entre os objetos
- O modelo estático mais utilizado é o **Diagrama de Classes**

Especificar interfaces entre objetos

- Especificação de interfaces permite que objetos e componentes sejam projetados em paralelo
- Objetos podem ter várias interfaces
  - Cada interface é um ponto de vista dos métodos fornecidos
  - A UML usa diagramas de classe para especificação de interfaces (semelhante a classes)

Diagrama de Classes

A estrutura do projeto

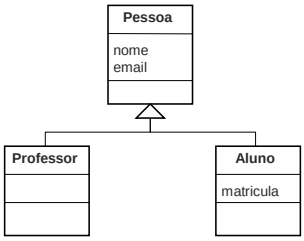
Alguns Diagramas UML

- Diagramas Estruturais (Estáticos)
  - Diagrama de Classes
  - Diagramas de Objetos
  - Diagrama de Caso de Uso
  - Diagrama de Componentes, etc.
- Diagramas Dinâmicos
  - Diagrama de Seqüência
  - Diagrama de Estados
  - Diagrama de Atividades
  - Diagrama de Colaboração, etc.

Alguns Diagramas UML

- Diagramas Estruturais (Estáticos)
  - **Diagrama de Classes**
  - Diagrama de Caso de Uso
  - Diagramas de Objetos
  - Diagrama de Componentes, etc.
- Diagramas Dinâmicos
  - Diagrama de Seqüência
  - Diagrama de Estados
  - Diagrama de Atividades
  - Diagrama de Colaboração, etc.

Primeiro Diagrama de Classes



Outro Diagrama de Classes

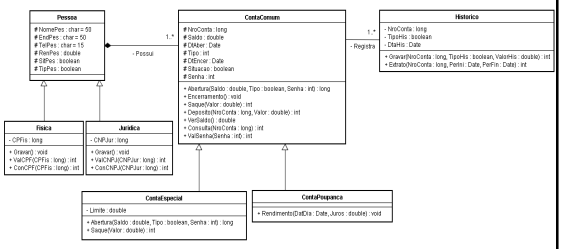


Diagrama de Classes

- Serve de apoio para a maioria dos outros diagramas
- Define a estrutura das classes do sistema
- Estabelece como as classes se relacionam

Diagrama de Classes

- É o mais importante e o mais utilizado diagrama da UML
- Permite a visualização das classes que compõem o sistema
- Representa
  - Atributos e métodos de uma classe
  - Os relacionamentos entre classes

Diagrama de Classes

- Apresenta uma visão estática de como as classes estão organizadas
- Preocupação com a estrutura lógica

Atributos

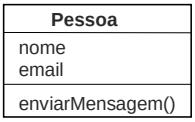
- Permite a identificação de cada objeto de uma classe
- Os valores dos atributos podem variar de instância para instância
- Atributos devem conter o tipo de dados a ser armazenado
  - Byte, boolean, int, double, char, String, etc.

Métodos

- São apenas declarados neste diagrama
  - Diagrama de Classes não define a implementação
- Outros diagramas permitem modelar o comportamento interno dos métodos
  - Diagrama de Sequência
  - Diagrama de Atividades

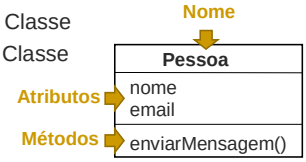
Representação de uma Classe

- Uma classe é representada por um retângulo com três divisões:
  - Nome da Classe
  - Atributos da Classe
  - Métodos da Classe



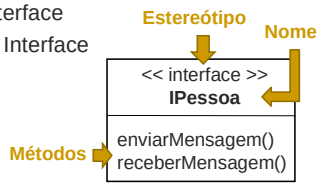
Representação de uma classe

- Uma classe é representada por um retângulo com três divisões:
  - Nome da Classe
  - Atributos da Classe
  - Métodos da Classe



Representação de uma interface

- Uma interface é semelhante a uma classe, mas não tem atributos
- Uma interface possui
  - Nome da Interface
  - Métodos da Interface
- Estereótipo
  - interface



Tipos de visibilidade

- Pública (+)
  - O atributo ou método pode ser utilizado por qualquer classe
- Protegida (#)
  - Somente a classe ou sub-classes terão acesso
- Privada (-)
  - Somente a classe terá acesso

Tipos de visibilidade

- Pública (+)
  - O atributo ou método pode ser utilizado por qualquer classe
- Protegida (#)
  - Somente a classe ou sub-classes terão acesso
- Privada (-)
  - Somente a classe terá acesso

Pessoa
nome - email
+ enviarMensagem()

Comunicação entre Objetos (I)

- Conceitualmente, objetos se comunicam através da troca de mensagens
- Mensagens definem:
  - O nome do serviço requisitado
  - A informação necessária para a execução do serviço
  - O nome do requisitante.

Comunicação entre Objetos (II)

- Na prática, mensagens são geralmente implementadas como chamadas de métodos
  - Nome = o nome do método
  - Informação = a lista de parâmetros
  - Requisitante = o método/objeto que realizou a chamada

Relacionamento

- Classes possuem relacionamentos entre elas (para comunicação)
  - Compartilham informações
  - Colaboram umas com as outras
- Principais tipos de relacionamentos
  - Associação
  - Agregação / Composição
  - Herança
  - Dependência

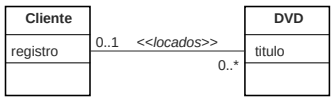
Associações

- Descreve um vínculo entre duas classes
  - Chamado **Associação Binária**
- Determina que as instâncias de uma classe estão de alguma forma ligadas às instâncias da outra classe

Multiplicidade

0..1	No máximo um. Indica que os Objetos da classe associada não precisam obrigatoriamente estar relacionados.
1..1	Um e somente um. Indica que apenas um objeto da classe se relaciona com os objetos da outra classe.
0..*	Muitos. Indica que podem haver muitos objetos da classe envolvidos no relacionamento
1..*	Um ou muitos. Indica que há pelo menos um objeto envolvido no relacionamento.
3..5	Valores específicos.

Representação de Associação



Agregação

- Tipo especial de associação
- Demonstra que as informações de um objeto precisam ser complementadas por um objeto de outra classe
- Associação Todo-Parte
  - objeto-todo
  - objeto-parte

Representação de Agregação

- Um losango na extremidade da classe que contém os *objetos-todo*

Composição

- Uma variação do tipo agregação
- Representa um vínculo mais forte entre objetos-todo e objetos-parte
- Objetos-parte **têm** que pertencer ao objeto-todo
  - O todo não existe (ou não faz sentido) sem a parte

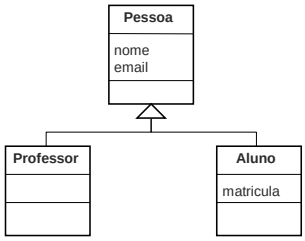
Representação da Composição

- Um losango preenchido
  - Da mesma forma que na Agregação, deve ficar ao lado do objeto-todo

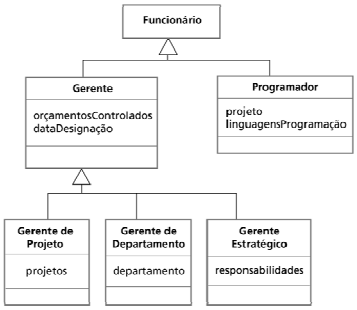
Especialização / Generalização

- Identificar super-classe (geral) e sub-classes (especializadas)
  - Semântica “é um”
  - Tudo que a classe geral pode fazer, as classes específicas também podem
- Atributos e métodos definidos na classe-mãe são **herdados** pelas classes-filhas

Especialização / Generalização



Hierarquia de generalização



Vantagens da herança

- O gráfico de herança é uma fonte de conhecimento sobre o domínio do sistema
- É um mecanismo de abstração usado para classificar entidades
- Mecanismo de reuso em vários níveis
  - Como projeto e programação

Problemas com herança

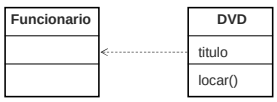
- Classes de objetos não são auto-contidas
  - Não podem ser compreendidas sem referência às suas super-classes
- Reusar gráficos da fase de análise pode ser ineficiente
  - Os gráficos de herança na análise, projeto e implementação têm diferentes funções (devem ser refinados)

Dependência

- Tipo menos comum de relacionamento
- Identifica uma ligação fraca entre objetos de duas classes

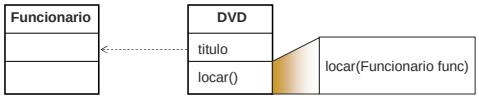
Dependência

- Representado por uma reta tracejada entre duas classes
- Uma seta na extremidade indica o dependente



Dependência

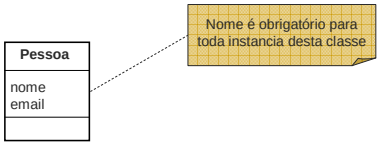
- Representado por uma reta tracejada entre duas classes
- Uma seta na extremidade indica o dependente



Notas

- Informativos
  - Função de comentários em classes, métodos ou atributos
  - Informa restrição de funcionalidade
  - Indicar condições para os relacionamentos, etc.

Notas



Resumo da Aula

- Um projeto orientado a objetos é obtido iterativamente pelas atividades
  - Definição de escopo, projeto de arquitetura, identificação dos objetos, desenvolvimento de modelos e especificação de interfaces
- Descrições textuais e cenários podem ajudar na identificação dos objetos
  - Substantivos e verbos

Resumo da Aula

- Definições de Diagramas de Classes
  - Classes e interfaces
  - Métodos e atributos
  - Visibilidade
  - Multiplicidade
  - Notas
- Relacionamentos
  - Herança, Associação, Agregação, Composição e Dependência

Referências

- Ian Sommerville. **Engenharia de Software**, 9a. Edição. 2011.
  - Cap. 7: Seções 7.1
- BOOCH, G., RUMBAUGH, J., JACOBSON, I. **UML, Guia do Usuário**. Rio de Janeiro: Campus, 2000.
  - Capítulos 4, 8 e 9