

Design e Programação OO

Dalton Serey
© 2004 DSC/UFG

Design OO

- Como projetamos um programa OO?
- Como fazemos um programa OO?

```
public interface Contador {  
    ...  
    public void reset();  
    public void conte();  
    ...  
}
```

Uma rápida revisão...

- O que e pra quê são **interfaces**?
- O que são **classes**?
- Qual a relação entre classes e interfaces?
- O que são **tipos**?
- O que são **nomes**? Quais tipos de nomes há?
- O que são **expressões** e **instruções**?
- O que são **métodos**? E **construtores**?

O que é *design*?

- O *design* de um programa consiste...
 - na enumeração e descrição detalhada das responsabilidades, atributos, métodos e dos relacionamentos das classes que o compõem
- Na prática, fazer *design* é...
 - identificar quais elementos do problema devem ser representados/modelados no programa e por quais elementos de linguagem
- Logo, ao programar estamos fazendo *design*!

Como fazemos *design*?

- Ponto de partida: **especificação de alto nível**
 - um texto descritivo da situação problema
 - um texto descritivo de uma solução imaginada
 - idéias coletadas em conversas com o cliente
 - listas de requisitos
 - etc...

Exemplo de especificação

- Construa um programa em Java que permita manipular expressões polinomiais com qualquer número de variáveis. O programa deve permitir simplificar, avaliar, somar, multiplicar expressões por escalares e por outras expressões. O programa deve permitir declarar e nomear qualquer número de expressões.
 - Um polinômio $p(n)$ de grau d , na variável n , é uma função na forma $a_1 \cdot n^1 + a_2 \cdot n^2 + \dots + a_d \cdot n^d$, onde as constantes a_i são chamadas de coeficientes e $a_d > 0$.

Substantivos são coisas

- Identifique as “**coisas**” da especificação
 - em geral, substantivos do texto
 - apenas os relevantes no problema
- No programa, substantivos serão
 - **atributos primitivos** se forem simples
 - números simples, caracteres, valores verdade
 - **objetos da API** se for possível usá-los
 - strings, números de precisão, datas...
 - **novas classes/objetos**, se forem específicos
 - tipicamente, coisas da aplicação

Versão 1.0

© 2004 Dalton Serey DSC/UFMG

Verbos são métodos

- Identifique as “**ações**” da especificação
- Interessam os verbos que indicam
 - coisas que os objetos devem fazer
 - coisas que devem ser feitas com os objetos
 - coisas e operações que os objetos devem executar uns com os outros
- No programa, verbos serão métodos
 - pode ser bom identificar *sujeito* e *objeto*
 - indicam *classe* e *argumentos* do método

Versão 1.0

© 2004 Dalton Serey DSC/UFMG

Adjetivos são interfaces

- Identifique “**adjetivos**” na especificação
- Em geral, indicam *interfaces*
 - *exemplo: comparável, executável, redutível, etc*
 - obs: substantivos também podem ser interfaces, tais como “contador”, “comparador”
- A questão é: quando devemos usar interfaces ou classes?
 - comece sempre com interfaces, se o design não se sustentar, rebaixe-as para classes

Versão 1.0

© 2004 Dalton Serey DSC/UFMG

Adjetivos são Interfaces

- Interfaces devem expressar o que há de comum entre classes de objetos
- Devemos declarar e usar interfaces sempre que for conveniente separar
 - o código que usa o tipo
 - da forma como ele será implementado

Versão 1.0

© 2004 Dalton Serey DSC/UFMG

De volta ao exemplo...

- Identifiquemos substantivos...
 - Construa um programa em Java que permita manipular expressões polinomiais com qualquer número de variáveis. O programa deve permitir simplificar, avaliar, somar, multiplicar expressões por escalares e por outras expressões. O programa deve permitir declarar e nomear qualquer número de expressões.
 - Um polinômio $p(n)$ de grau d , na variável n , é uma função na forma $a_1*n^1 + a_2*n^2 + \dots + a_d*n^d$, onde as constantes a_i são chamadas de coeficientes e $a_d > 0$.

Versão 1.0

© 2004 Dalton Serey DSC/UFMG

De volta ao exemplo...

- Identifiquemos substantivos...
 - Construa um programa em Java que permita manipular **expressões polinomiais** com qualquer **número de variáveis**. O programa deve permitir simplificar, avaliar, somar, multiplicar expressões por **escalares** e por outras expressões. O programa deve permitir declarar e nomear qualquer **número de expressões**.
 - Um **polinômio de grau d** , na variável n , é uma **função**
 $p(n) = a_1*n^1 + a_2*n^2 + \dots + a_d*n^d$, onde as **constantes a_i** são chamadas de **coeficientes** e $a_d > 0$.

Versão 1.0

© 2004 Dalton Serey DSC/UFMG

De volta ao exemplo...

- Como modelar os conceitos abaixo?
 - expressões polinomiais
 - número de variáveis
 - variáveis
 - escalares
 - número de expressões
 - polinômio de grau d
 - função
 - constantes chamadas de coeficientes

Versão 1.0

© 2004 Dalton Serey DSC/UFG

De volta ao exemplo...

- Identifiquemos verbos...
 - Construa um programa em Java que permita **manipular** expressões polinomiais com qualquer número de variáveis. O programa deve permitir **simplificar**, **avaliar**, **somar**, **multiplicar** expressões por escalares e por outras expressões. O programa deve permitir **declarar** e **nomear** qualquer número de expressões.
 - Um polinômio $p(n)$ de grau d , na variável n , é uma função na forma $a_1*n^1 + a_2*n^2 + \dots + a_d*n^d$, onde as constantes a_i são chamadas de coeficientes e $a_d > 0$.

Versão 1.0

© 2004 Dalton Serey DSC/UFG

De volta ao exemplo...

- manipular
- simplificar
- avaliar
- somar
- multiplicar
- declarar
- nomear

Versão 1.0

© 2004 Dalton Serey DSC/UFG

Objetos da solução

- São objetos **específicos do programa**
- Não são identificados na especificação
- Exemplos:
 - interface com o usuário
 - controle da aplicação
 - armazenamento de coleções ou BDs
- Não é raro que ao executar, o programa precise de **uma única instância** deles...

Versão 1.0

© 2004 Dalton Serey DSC/UFG

Detalhando o design

- Declare as classes envolvidas
- Atribua **responsabilidades**
 - determine o papel de cada classe na aplicação
 - identifique atributos (parte do estado geral)
- Organize os métodos identificados
 - aloque-os para as classes adequadas
 - coloque-os junto aos dados necessários
 - identifique suas assinaturas

Versão 1.0

© 2004 Dalton Serey DSC/UFG

Detalhando o design

- Na prática, esta etapa pode ser feita em Java
 - inicialmente, crie as classes vazias
 - coloque os atributos
 - coloque *stubs* para os métodos
 - a cada passo documente com javadoc!

Versão 1.0

© 2004 Dalton Serey DSC/UFG

Valide o design

- Antes de escrever código
 - faça um desenho do sistema em execução
 - coloque os objetos da aplicação e seus *links*
 - avalie o *design*, simulando uma execução
- Sugestão: uso de CRC cards p/ *design*
 - classes
 - responsibilities
 - collaborations

Versão 1.0

© 2004 Dalton Serey DSC/UFG

Escreva testes

- Inicie escrevendo **clientes de teste**
- É uma classe usada apenas para testar o serviço de classes provedoras
- Isto nos obriga a (re)pensar:
 - as assinaturas dos métodos e construtores
 - valores e tipos de retorno
 - condições de contorno
- Só então, comece a escrever a aplicação!

Versão 1.0

© 2004 Dalton Serey DSC/UFG

O processo é cíclico

- Todo o processo descrito é cíclico:
 - identificamos elementos na especificação
 - os mapeamos e modelamos em OO/Java
 - detalhamos métodos e assinaturas
 - validamos o *design* proposto
 - escrevemos testes
 - escrevemos código
 - e repetimos todo o processo...

Versão 1.0

© 2004 Dalton Serey DSC/UFG

Fazer design não é fácil!

- Princípios que podem ajudar:
 - O princípio KISS
 - "Keep it Short and Simple"
 - ou se você preferir, "Keep It Simple, Stupid!"
 - A Lei de Demétrio
 - "fale apenas com seus amigos mais próximos"
 - Minimize acoplamento, maximize coesão
- Não se preocupe, fazer design só se aprende pra valer através de experiência...

Versão 1.0

© 2004 Dalton Serey DSC/UFG

Exercícios...

- Construa o programa especificado ao longo deste material.
- Construa um programa em Java que permita jogar o conhecido [jogo da velha](#).

Versão 1.0

© 2004 Dalton Serey DSC/UFG