

Revisão : Paradigma de Orientação a Objetos

Karin Becker
Engenharia de Software N
Instituto de Informática - UFRGS

Conceitos

- ♦ Objeto: módulo construtor básico
 - sistema é uma coleção de objetos cooperantes que se comunicam para atingir um objetivo
- ♦ Objeto: visão unificada dos aspectos estáticos e dinâmicos
 - estrutura (estado): dados
 - comportamento: operações
- ♦ cooperação entre objetos é do tipo “cliente-servidor”
 - cada objeto põe à disposição da comunidade um conjunto de serviços
 - » servidor: serviços
 - objetos requisitam serviços a outros objetos
 - » cliente: requisições
 - princípio da terceirização

Conceitos: Objeto

- ♦ Objeto
 - identidade:
 - » imutável ao longo de toda vida do objeto
 - » atribuído pelo sistema, inacessível, usado internamente
 - estado (estrutura)
 - » valores assumidos por propriedades em um dado momento
 - » pode variar ao longo da vida do objeto
 - comportamento
 - » serviços (operações) que o objeto sabe realizar
 - » alguns serviços resultam na alteração do estado do objeto
 - unidade existente exclusivamente em **tempo de execução**

Três janelas (objetos)

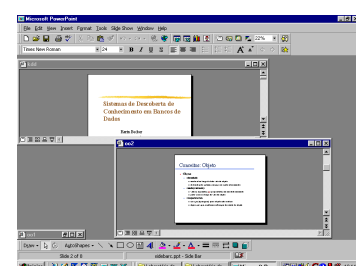
Janela do powerpoint

Janela de um arquivo

Janela de outro arquivo

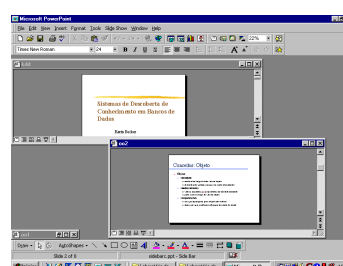
identidades diferentes, desconhecidas externamente
Cada qual com seu estado

Comportamento da janela powerpoint



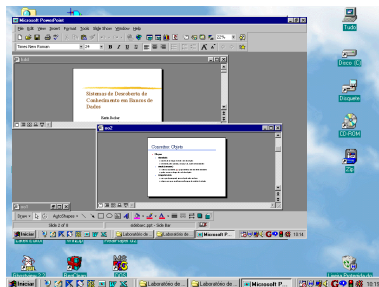
abrir
fechar
minimizar
maximizar
alterar tamanho
incluir subjanela
excluir subjanela
etc

Comportamento da janela powerpoint



alterar tamanho

Comportamento da janela powerpoint



Novo estado estado

Conceitos: Classe

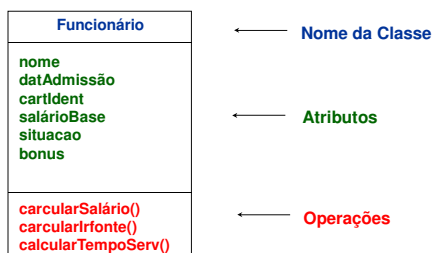
- ♦ Classe
 - descrição das características comuns a vários objetos
 - estrutura
 - » propriedades relevantes
 - » atributos, variáveis
 - comportamento
 - » operações
 - unidade de **descrição e execução**
 - todo objeto é instância de uma classe



são instâncias da classe PESSOA

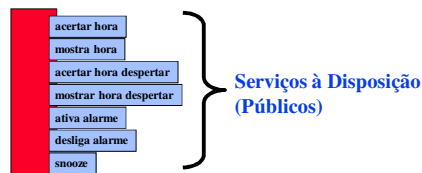
==
têm o mesmo comportamento e estrutura

Representação de uma Classe: UML



Conceitos: Abstração de Dados

- ♦ Origem: Tipos Abstratos de Dados (ADT)
- ♦ abstração de dados
 - define uma estrutura de dados através das operações que a manipulam
 - **INTERFACE PÚBLICA**
- ♦ Exemplo: DESPERTADOR



Conceitos: Information Hidding

- ♦ mascaramento (ocultamento) de informações
- ♦ **Interface**
 - serviços que o objeto sabe oferecer (O QUÊ)
 - operações
 - público
- ♦ **Implementação**
 - implementação da estrutura e do comportamento (COMO?)
 - » variáveis
 - » código para operações da interface (método)
 - » operações internas (privadas)
- ♦ **Foco no projeto da Interface das classes**
 - **Coesão, acoplamento**
 - **Reuso**
 - **extensibilidade**

Conceitos: mensagem

- ♦ mensagem
 - único meio de comunicação entre objetos
 - requisição de um serviço
 - analogia: chamada de subrotina
- ♦ mensagem
 - receptor
 - operação
 - parâmetros (opcional)
- ♦ um objeto só responde às mensagens que entende
 - operação definida na interface pública
- ♦ Para enviar uma mensagem, é necessário conhecer o objeto receptor !!

mensagens

Receptor operação argumentos



mostrar hora



acertar hora (4, 32, pm)



mostrar data

Conceitos: Mensagens

- é o receptor da mensagem que decide como respondê-la
 - depende da classe do objeto receptor da mensagem
 - depende da implementação dada à operação



deslocar



deslocar



deslocar



deslocar

Conceitos: Herança

- permite definir uma nova classe tomando outra existente como base
 - define uma hierarquia de classes
 - super-classe / subclasse
 - subclasse herda todas as variáveis, operações e métodos (implementação) definidos em sua superclasse
 - direto: descritos diretamente na superclasse
 - indireto (transitivo): herdado pela superclasse de seus ancestrais
 - alterações podem ser feitas na subclasse
 - novas variáveis e operações
 - código para operações herdadas
 - economia de descrição, facilidade de gerenciamento de estrutura/comportamento compartilhado, reuso

Conceitos: Herança

Classe Relógio

hora
minuto

acertaHora
mostraHora
marcaHora

```
acertaHora(novaHora, novoMin)
  hora := novaHora;
  minuto:= novoMinuto;
  self mostraHora;
end
```

variável
operação pública
operação privada

Conceitos: Herança

Classe Relógio
hora minuto
acertaHora mostraHora marcaHora

Classe Despertador subclasse de Relógio
horad minutod
acertaHora acertaHoraDespertar mostraHoraDespertar ativaAlarme desligaAlarme marcaHora desperta

- herdado: despertador tem hora e minuto, mostra e acerta hora
- despertador tem novas características
- despertador modifica parte de comportamento herdado

Conceitos: Herança

Classe Despertador subclasse de Relógio

horad
minutod


acertaHora
acertaHoraDespertar
mostrarHoraDespertar
ativaAlarme
desligaAlarme
marcaHora
desperta


sobreescreta ("overriding"):
substituição de código herdado


```
acertaHora(novaHora, novoMin)
  hora := novaHora;
  minuto := novoMin;
  self mostraHora;
  if hora = horad and minuto = minutod
  then self desperta;
end
```

Conceitos: Polimorfismo

- ♦ polimorfismo
 - conceito de programação OO
 - uma variável pode representar diferentes objetos
 - linguagens tipadas impõem restrições à capacidade polimórfica


umObjeto := 

umObjeto := 


umObjeto := 

Conceitos: Ligação Dinâmica

- ♦ ligação dinâmica
 - quem enviou a mensagem não se preocupa em como o objeto responderá
 - o objeto **receptor** sempre decide como responder à mensagem recebida em **tempo de execução**
 - a resposta depende da classe do objeto receptor da mensagem

umObjeto := 

umObjeto acertarHora

umObjeto := 

umObjeto acertarHora

Programação Orientada a Objetos

- ♦ mais ou menos fiel aos conceitos básicos
- ♦ idiosincrasias
 - tipadas vs. não tipadas
 - puras vs. híbridas
 - herança simples vs. múltipla
- ♦ Exemplos
 - smalltalk
 - » pura, não tipada, herança simples
 - Java
 - » pura, tipada (não parametrizada), herança simples *
 - C++
 - » híbrida, tipada (parametrizada), herança múltipla
 - Pascal OO, Eiffel, etc ...

Alguns mitos da OO

- ♦ “usei uma linguagem (notação, metodologia) OO, e meu programa (sistema, modelagem) é orientado a objetos”
- ♦ “sistemas OO são mais modulares e fáceis de manter”
- ♦ “a grande vantagem da OO é reuso”
- ♦ “minhas classes são reusáveis”
- ♦ “a base da reusabilidade é herança”

Uma visão crítica ...

- ♦ OO não é sinônimo de bom
- ♦ nenhuma tecnologia por si só garante contra o mau uso que pode ser feito dela
- ♦ reuso não vem de graça : reusar tem que ser mais fácil e rápido que desenvolver do zero
- ♦ fazer bons sistemas OO é difícil: desenvolver componentes reutilizáveis é mais ainda!

UML : Diagramas Fundamentais para a OO

- ♦ Diagrama de classes
 - Classes
 - Interfaces das classes
 - Estado e comportamento
 - Relacionamentos estruturais (estáticos) entre objetos de classes
- ♦ Diagramas de Interação
 - Troca de mensagens
 - Diagrama de sequência
 - » Sequência das mensagens
 - Diagrama de colaboração
 - » Como os objetos colaboram baseado no conhecimento que um tem do outro

Diagrama de Classes : Exemplo

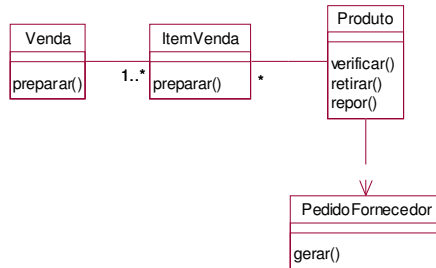


Diagrama de Sequencia : Exemplo

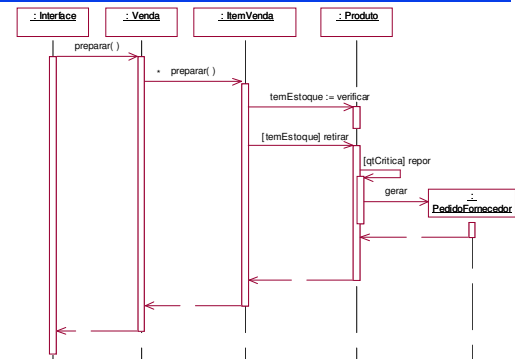


Diagrama de Colaboração : Exemplo

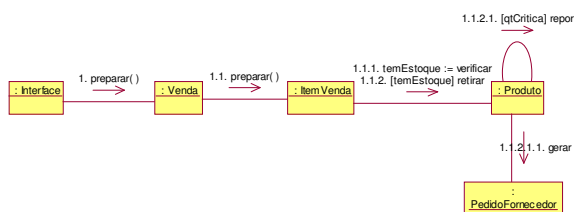
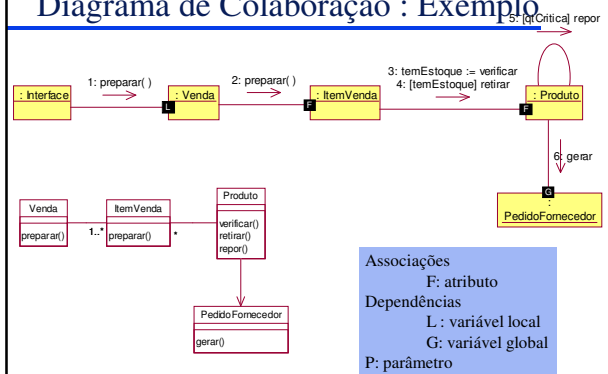


Diagrama de Colaboração : Exemplo



Diagramas da UML

- Modelos de software
 - Diferentes pontos de vista (estrutural vs. comportamental)
 - Complementares
 - Definem um nível de abstração
- Diferentes fases do desenvolvimento de software são baseados no mesmo tipo de diagrama
 - Níveis de detalhe diferentes
 - Propósitos de representação diferentes
- A UML é uma notação flexível e expressiva
 - “assim a nível de proposta, cada um faz o que gosta” (João Bosco)

UML 2.0

- Há 2 documentos que descrevem UML 2.0
 - UML 2.0 Superstructure, que descreve os elementos para modelagem de estrutura e comportamento dos sistemas
 - UML 2.0 Infrastructure – metalinguagem e elementos para definição e adaptação de UML
 - » orientada a DESENVOLVEDORES DE FERRAMENTAS
- Pacote UML2 inclui ainda:
 - Object Constraint Language (OCL), linguagem para especificação de restrições entre modelos
 - Diagram Interchange (XMI) para intercâmbio de modelos UML entre ferramentas