Programação Orientada a Objetos - Aula 3:

Introdução a *Smalltalk*

Raphael Mendes de O. Cóbe
raphael@ccsa.ufrn.br

Instituto de Matemática e Estatística - IME
Universidade de São Paulo - USP

Conteúdo

"The purpose of the Smalltalk project is to provide computer support for the creative spirit of everyone." - Daniel Ingalls, 1981.

Implementações (populares) de Smalltalk

- VisualWorks Smalltalk (CinCom):
  - Comercial;
  - Melhor suporte ao desenvolvimento de aplicativos visuais;
  - Suporte Corporativo;
- Squeak Smalltalk:
  - OpenSource;
  - Comunidade ativa;
  - Material disponível para o aprendizado;
  - OLPC Etoys;

Arcabouços e Ferramentas I

- **Monticello Browser**: Sistema de empacotamento e controle de versões (similar ao cvs);
- **SUnit**: Arcabouço para testes de Unidade em Squeak Smalltalk;
- **Refactoring Browser**: Ferramenta que disponibiliza um conjunto vasto de tarefas de refatoração, como:
  - Renomear Variáveis;
  - Renomear Métodos;

Arcabouços e Ferramentas II

- **Magma**: Implementação de um sistema de banco de dados orientado a objetos em Smalltalk;
- **Seaside**: Arcabouço para o desenvolvimento de aplicações web dinâmicas;
  - **ShoreComponents**: Componentes extras para o desenvolvimento de aplicações web com Seaside;
- **Morphic**: Kit para desenvolvimento de GUI para aplicativos desktop;

Componentes

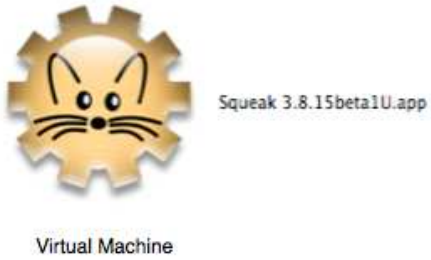


Figura: Máquina Virtual

Figura: Código Fonte

Figura: Imagem + arquivo .changes

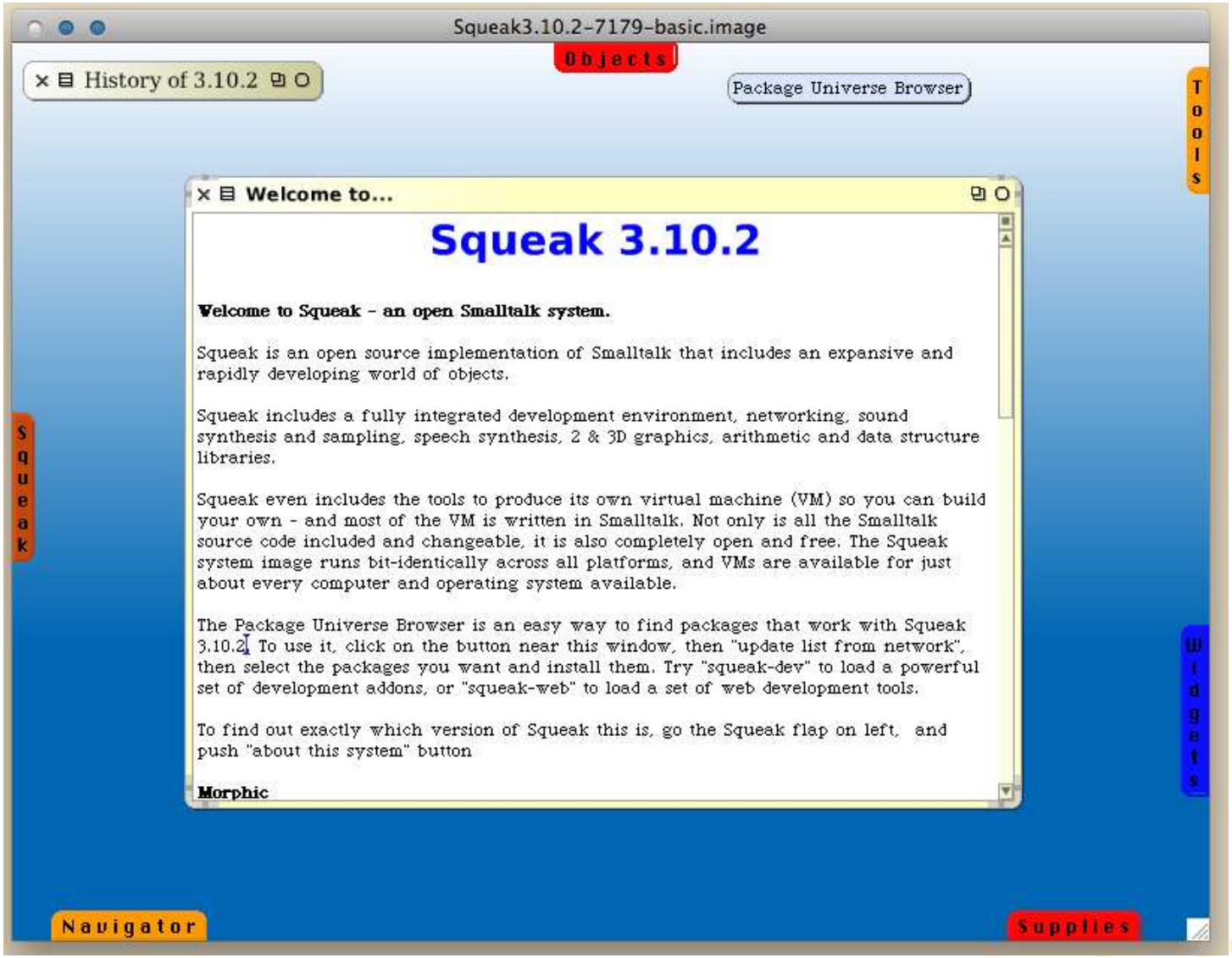
Componentes do Squeak I

- Máquina Virtual Squeak
  - Parte do Sistema que é específica para cada plataforma;
  - Disponível para um grande número de plataformas;
    - Mac OS, Windows, Unixes (Linuxes, BSDs, Solaris);
  - Desenvolvida, testada e simulada em Squeak Smalltalk;
- Código Fonte do Squeak:
  - Código fonte das partes do Squeak que não modificam com frequência;
  - Pequenas diferenças entre versões;

Componentes do Squeak II

- Imagem e arquivo *.changes*:
  - Uma "fotografia" do sistema congelada no tempo de um Squeak rodando;
  - Arquivo *.image*:
    - Armazena o estado de *todos* os objetos do sistema;
    - Compilador, ambiente de desenvolvimento;
    - Inclusos Métodos e Classes (**Eles também são Objetos!**);
  - Arquivo *.changes*
    - Registro das mudanças realizadas na Imagem;
    - Salvação em caso de interrupção abrupta da máquina virtual;

A Cara do Squeak!



Preparando o Ambiente

Figura: Renomeando botões do Mouse

Demonstrar os diferentes menus.

Salvando e reiniciando uma sessão do Squeak

- Modificações são gravadas no arquivo *.image* e *.changes*;
- Ao abrir novamente o Squeak, esse deve se encontrar no mesmo estado antes de fechá-lo;
- Arquivos *.image* não são as únicas formas de compartilhar código em Squeak;
- O arquivo *.changes*:

```
----STARTUP----an Array(1 March 2010 7:13:38 pm)
    as Squeak3.10.2-7179-basic.image!

Time now.!

Workspace open.!

----QUIT----an Array(1 March 2010 7:30:12 pm)
    Squeak3.10.2-7179-basic.image priorSource: 1846835!
```

Demonstrar salvar uma nova imagem e ver novo arquivo *.changes*.

Ferramentas Básicas para um *Olá Mundo*

- Workspace:
  - São ferramentas importantes para executar trechos de código;
  - Utilizados para "testes";
  - Podem instanciar objetos e enviar mensagens para os mesmos;
  - Armazenar descrições textuais sobre o que se passa no ambiente de desenvolvimento;
- Transcript:
  - Utilizado para registrar mensagens de depuração;
  - É uma espécie de console do sistema (System.out do java);
  - Extremamente lento;
  - Não thread-safe;

Demonstrar Workspace e Transcript;

O que aconteceu?

- Avaliação da nossa primeira expressão Smalltalk.
- Enviamos a mensagem `show: 'Olah Mundo'` para o objeto Transcript;
- Enviamos a mensagem `cr` para o objeto Transcript;
- O objeto Transcript decide o que fazer com essa mensagem:
  - Procurar em seus métodos;
  - Encontrar métodos equivalentes e executá-los;

Exibir implementação do método `show:`;

Sintaxe do Squeak Smalltalk

- Sintaxe minimal;
- Existem apenas 6 palavras-chave (pseudo-variáveis) na linguagem:
  - *self*, *super*, *nil*, *true*, *false* e *thisContext*;
- Não existe sintaxe para estruturas de controle;
- Não existe sintaxe para a definição de novas classes;
- Tudo é realizado através de troca de mensagens;
- Não existência de instruções *If*, por exemplo;
  - Envio de mensagens para objetos Boolean:
  - `ifTrue:` e `ifFalse:`;

Elementos sintáticos

- Inteiros, Pontos Flutuantes;
- Símbolos:
  - `#MAC0441`
- Arrays Dinâmicos:
  - {1. 3. 5+5}
- Arrays de Tempo de compilação:
  - #(1 #(3 3) 5+5)
- Comentário:
  - "Comentario";
- Definição de variáveis:
  - `|var1 var2|`
- Atribuição:
  - `var1:=1.`

Demonstrar declaração e atribuição de variáveis.

Envio de Mensagens I

- Mensagens Unárias:
  - Não aceitam parâmetros;
  - Exemplos:
    - `23 factorial`
    - `10 negated`
    - `'banana' reverse`
    - `-20 isZero`
- Mensagens Binárias:
  - Recebem um único parâmetro;
  - Um ou mais dos Símbolos $+ - /\ * \ <>= @\%|\& \ ? \ ,$
  - Exemplos:
    - `354 + 43`
    - `20 - 2`
    - `74 * 3`
    - `'Small' , 'talk'`

Envio de Mensagens II

- Mensagens com seletores ou palavras chave:
  - Sintaxe: `umObjeto seletor1: valor1 seletor2: valor2 seletor3: valor3;`
  - Exemplos:
    - `substring := 'Alan Kay' copyFrom: 3 to: 4`
    - `2 raisedTo: 6 modulo: 10;`
    - `#(1 2 #(3 3)) at: 3.`

Mostrar Exemplos de envio de Mensagens;

Precedência de Operadores I

- Smalltalk não define uma ordem de precedência de operadores matemáticos;
- Operadores são apenas mensagens como outras quaisquer;
- Existem 4 regras básicas:
  1. As mensagens são sempre processadas da esquerda para a direita;
  2. Mensagens unárias $->$ mensagens binárias $->$ mensagens com Seletores;
  3. A atribuição ($:=$) é sempre processada por último;
  4. Parênteses podem ser utilizados para mudar a ordem de envio das mensagens;

Precedência de Operadores II

- Exemplo:
  - 2+1*3
  - Em linguagens com precedência de operadores: 5;
  - Em Smalltalk: 9
    - A mensagem + é enviada ao objeto 2 com o objeto 1 como parâmetro;
    - Em seguida a mensagem * é enviada ao resultado da operação anterior, com o objeto 3 como parâmetro;

Mostrar Exemplo de precedência de operadores; Mostrar Inconsistências em operações aritméticas;

Mensagens Encadeadas

- Envio de várias mensagens para vários objetos;
- Exemplos:
  - Qual o resultado da avaliação da expressão: `3 + 3 negated * 10`?
  - Quais as diferenças entre as expressões:
    - `'MAC0441' copyFrom:5 to:6 asNumber negated.`
    - `('MAC0441' copyFrom:5 to:6 asNumber) negated.`
    - `('MAC0441' copyFrom:5 to:6) asNumber negated.`
  - Qual delas *faz sentido*?

Mensagens em Cascata

- Forma de encurtar envio de mensagens consecutivas para o mesmo objeto;
- Omissão do nome do Objeto;
- Separação de mensagens por ;
- A mensagem yourself;

Demonstrar o uso do operador de encadeamento;

Identificando Mensagens

- Envio de mensagem é composto de um receptor, um seletor de mensagem e parâmetros opcionais;

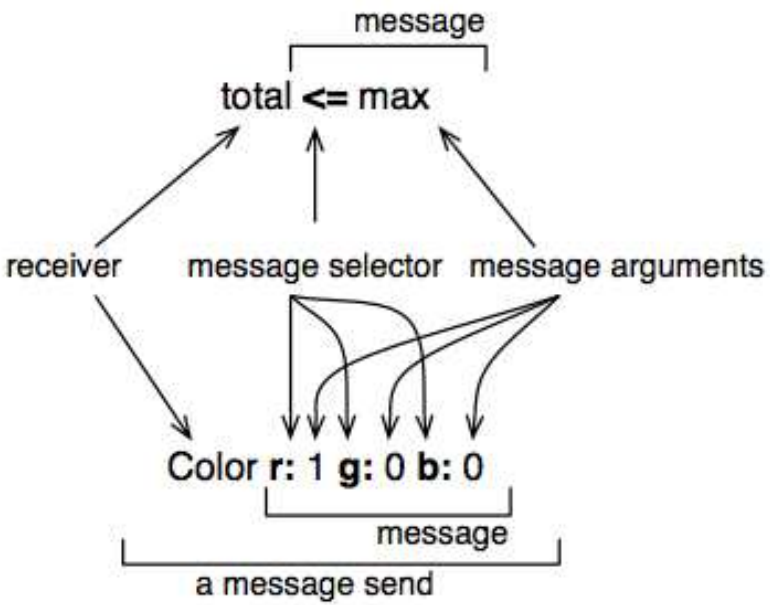


Figura: Mensagens em Smalltalk

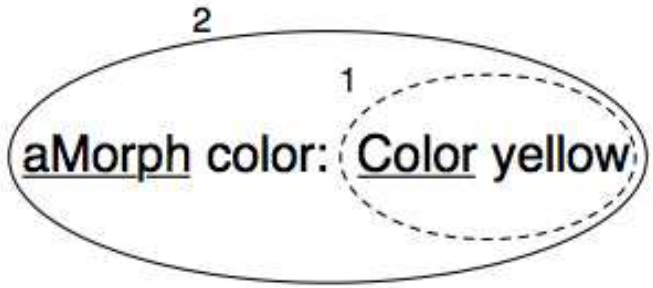


Figura: Mensagens em Smalltalk

Modelo de Objetos do Smalltalk I

- 5 Regras:
  1. **Tudo é um Objeto;**
  2. **Todo objeto é uma instância de uma classe;**
  3. **Toda Classe possui uma Superclasse;**
  4. **Tudo acontece por envio de mensagens;**
  5. **O lookup de métodos segue a árvore hierárquica;**

Modelo de Objetos do Smalltalk II

- Todos os Métodos são Públicos;
- Métodos são agrupados em Protocolos;
- Protocolos Comuns: *accessing*, *printing*, *testing*, *comparing*, ...
- O protocolo *private*;
- Métodos podem ler qualquer variável de instância;
- Novas Classes são normalmente criadas enviando a mensagem `subclass: instanceVariableNames:`

Mostrar Criação de uma Classe Simples;

Exportando definições (File-out/File-in)

- Categorias podem ser escritas como Arquivos de Texto;
- Operação *fileOut* no System Browser;
- Categorias podem ser Carregadas a partir do File Browser;

Demonstrar FileOut/FileIn;