Interface Gráfica e Banco de Dados em Java

Introdução

Licença para uso e distribuição

Este material está disponível para uso não-comercial e pode ser derivado e/ou distribuído, desde que utilizando uma licença equivalente.

Atribuição-Uso Não-Comercial-Compartilhamento pela mesma licença, versão 2.5

http://creativecommons.org/licenses/by-nc-sa/2.5/deed.pt

Você pode copiar, distribuir, exibir e executar a obra, além de criar obras derivadas, sob as seguintes condições: (a) você deve dar crédito ao autor original, da forma especificada pelo autor ou licenciante; (b) você não pode utilizar esta obra com finalidades comerciais; (c) Se você alterar, transformar, ou criar outra obra com base nesta, você somente poderá distribuir a obra resultante sob uma licença idêntica a esta.

Sobre o curso

- Aulas expositivas;
- Exercícios de fixação;
- Trabalho prático como avaliação ao final;
- Conteúdo:

Interfaces gráficas:

- Componentes GUI – Parte I
- Componentes GUI – Parte II
- Componentes GUI – Parte III

Banco de dados:

- A API JDBC;
- O *framework* Hibernate.

Sobre o material do curso

- Bibliografia:
  - Deitel & Deitel – Java, Como Programar;
  - Horstmann & Cornell – Core Java 2: Volume II;
  - Bauer & King – Hibernate em Ação.
- Slides:
  - Autoria de Vítor Souza (vitorsouza@gmail.com);
  - Abordagem "*hands on*";
  - Baseado na bibliografia.

Sobre os alunos

- Assume-se que os alunos:
  - Sabem Java Básico;
  - Não conhecem nada sobre interfaces gráficas em Java (AWT, JFC/Swing);
  - Não conhecem nada sobre acesso a bancos de dados em Java (JDBC).
- Recomenda-se aos alunos:
  - Perguntar SEMPRE que houver dúvida;
  - Estudar a bibliografia em casa;
  - Fazer todos os exercícios;
  - Participar de um JUG.

Sobre o instrutor

- Formação:
  - Graduação em Ciência da Computação, com ênfase em Engenharia de Software, pela UFES;
  - Bolsista de Mestrado em Informática na UFES.
- Java:
  - Desenvolvedor Java desde 1999;
  - Especialista em desenvolvimento Web;
  - JUG Leader do ESJUG.
- Profissional:
  - Professor substituto no DI / UFES.

Objetivos do curso

- Tornar os alunos programadores capazes de:
  - Desenvolver interfaces gráficas (janelas) utilizando a API Swing da plataforma Java SE;
  - Construir código Java que consiga conectar-se a diversos sistemas gerenciadores de banco de dados usando o padrão JDBC;
  - Implementar aplicações que utilizem o *framework* Hibernate para mapeamento objeto/relacional;
  - Entender e alterar código existente que faça uso destas APIs.

Introdução à interfaces gráficas

- GUI – *Graphical User Interfaces* = Interfaces Gráficas com o Usuário;
- Linguagens de programação oferecem *toolkits* (kits de ferramentas) para criação de componentes gráficos (*widgets*);
- Os componentes dependem da plataforma (Windows, MacOS, Gnome GTK, KDE QT, etc.).

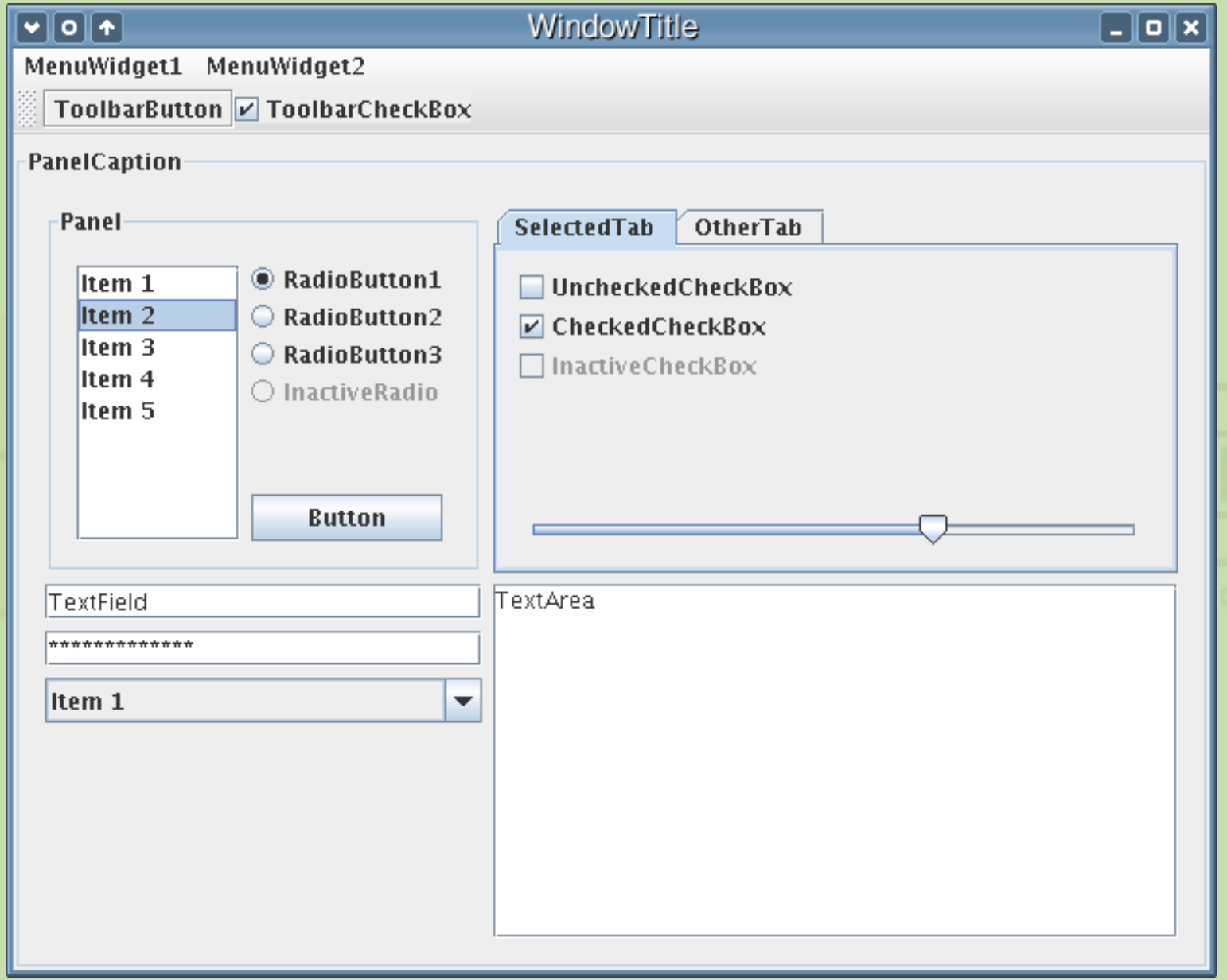
Java *Abstract Window Toolkit* (AWT)

- Em 1995, a Sun criou a API *Abstract Window Toolkit* (AWT) para J2SE 1.0;
- Fina camada de abstração sobre GUI nativa;
  - Alta fidelidade ao *toolkit* nativo;
  - Maior integração com aplicações nativas;
  - Interfaces desenvolvidas em uma plataforma não ficavam bonitas em outras;
  - Contrário ao princípio WORA.

Java Swing

- A partir do Java 1.2, a Sun incluiu na API do Java SE a tecnologia Swing;
- Grande abstração sobre GUI nativa:
  - Escrito em Java puro (usando AWT e Java2D);
  - Aparência consistente em plataformas diferentes;
  - *Look & Feel* plugável;
  - Altamente baseado na arquitetura MVC com projeto altamente orientado a objetos;
  - Perda de desempenho e curva de aprendizado mais íngreme em relação a outros *toolkits*.

Exemplo de janela Swing



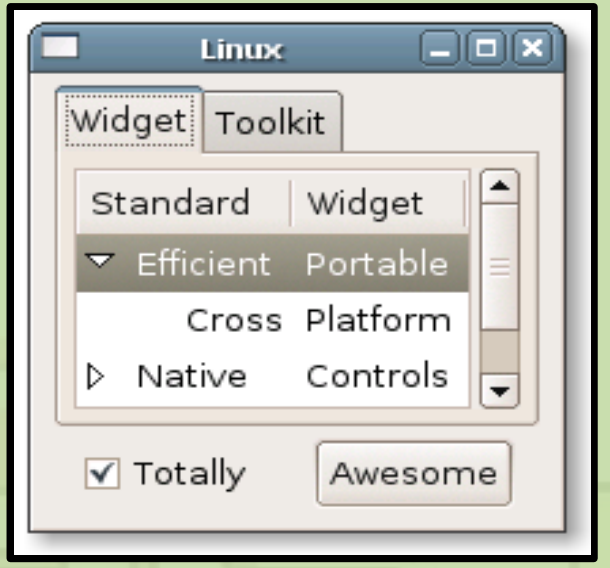
Java Foundation Classes (JFC)

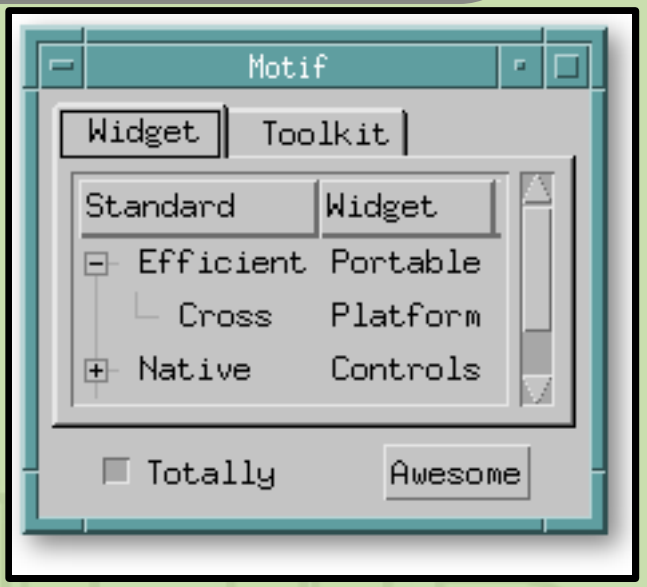
- União das tecnologias AWT, Swing e Java2D;
- Java2D: criação de desenhos em duas dimensões em Java;
- *Framework* oficial provido pela plataforma Java SE para construção de GUIs portáveis.

Standard Widget Toolkit (SWT)

- Na mesma época, IBM Visual Age (IDE) torna-se *open source* – nasce o Eclipse IDE;
- Objetivos: ter *look & feel* mais próximo do nativo e melhor desempenho, mantendo o alto nível;
- Meio termo entre AWT e Swing:
  - Usa JNI para acessar *toolkits* nativos;
  - Implementa seu próprio código quando necessário.
- JFace: classes utilitárias para implementação de tarefas maçantes em SWT.

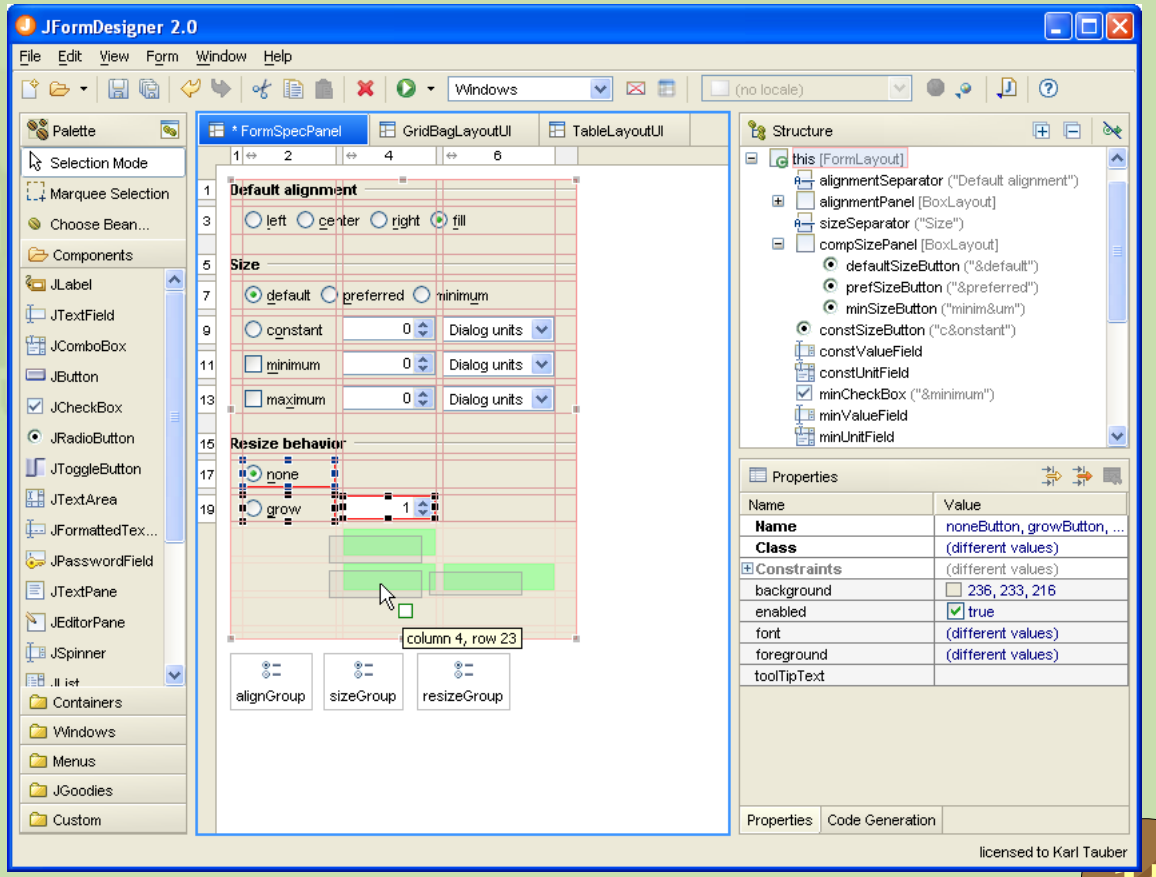
Exemplo de janelas SWT





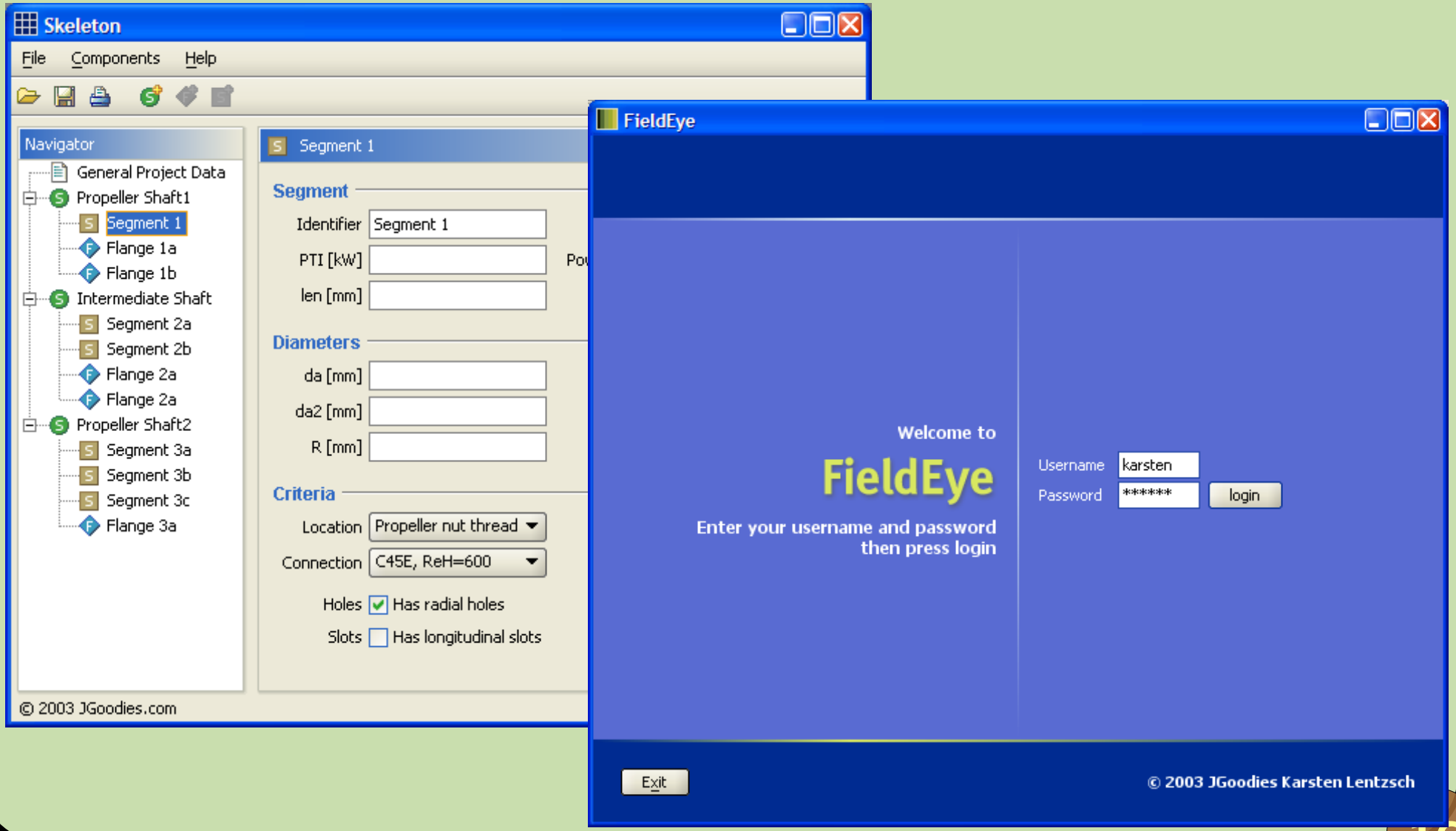
Ferramentas de desenho

- Integradas às IDEs:
  - Netbeans (Matisse);
  - Eclipse (Visual Editor, Matisse4Eclipse);
  - Outras...
- Independentes:
  - Swing Designer;
  - JFormDesigner;
  - FormLayoutMaker;
  - Abeille;
  - Outras...



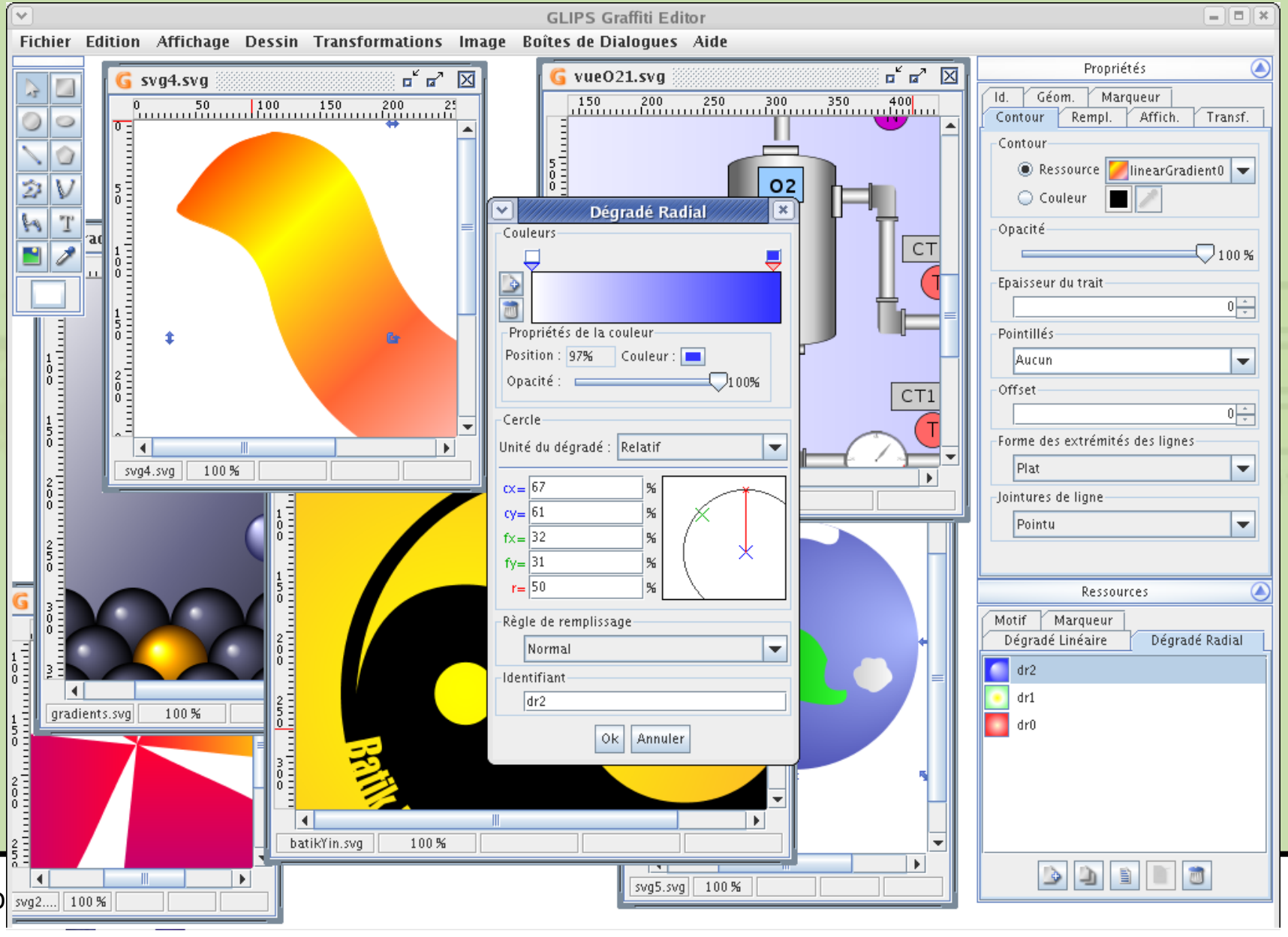
Outras ferramentas

- Ex.: JGoodies (www.jgoodies.com):



Exemplos de aplicações em Swing

- GLIPS (http://glipssvgeditor.sourceforge.net):

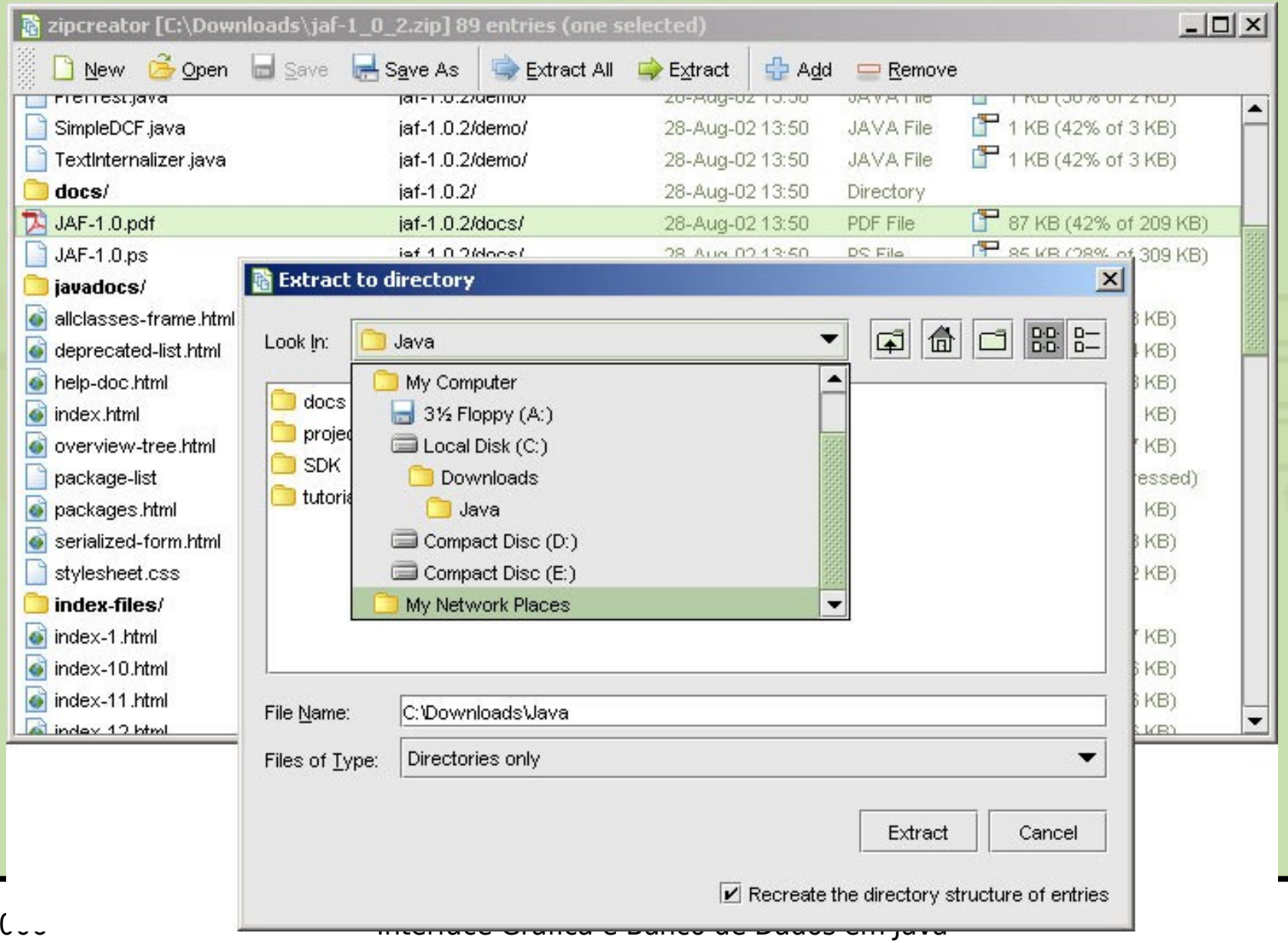


Exemplos de aplicações em Swing

- Jake2 (http://bytonic.de):

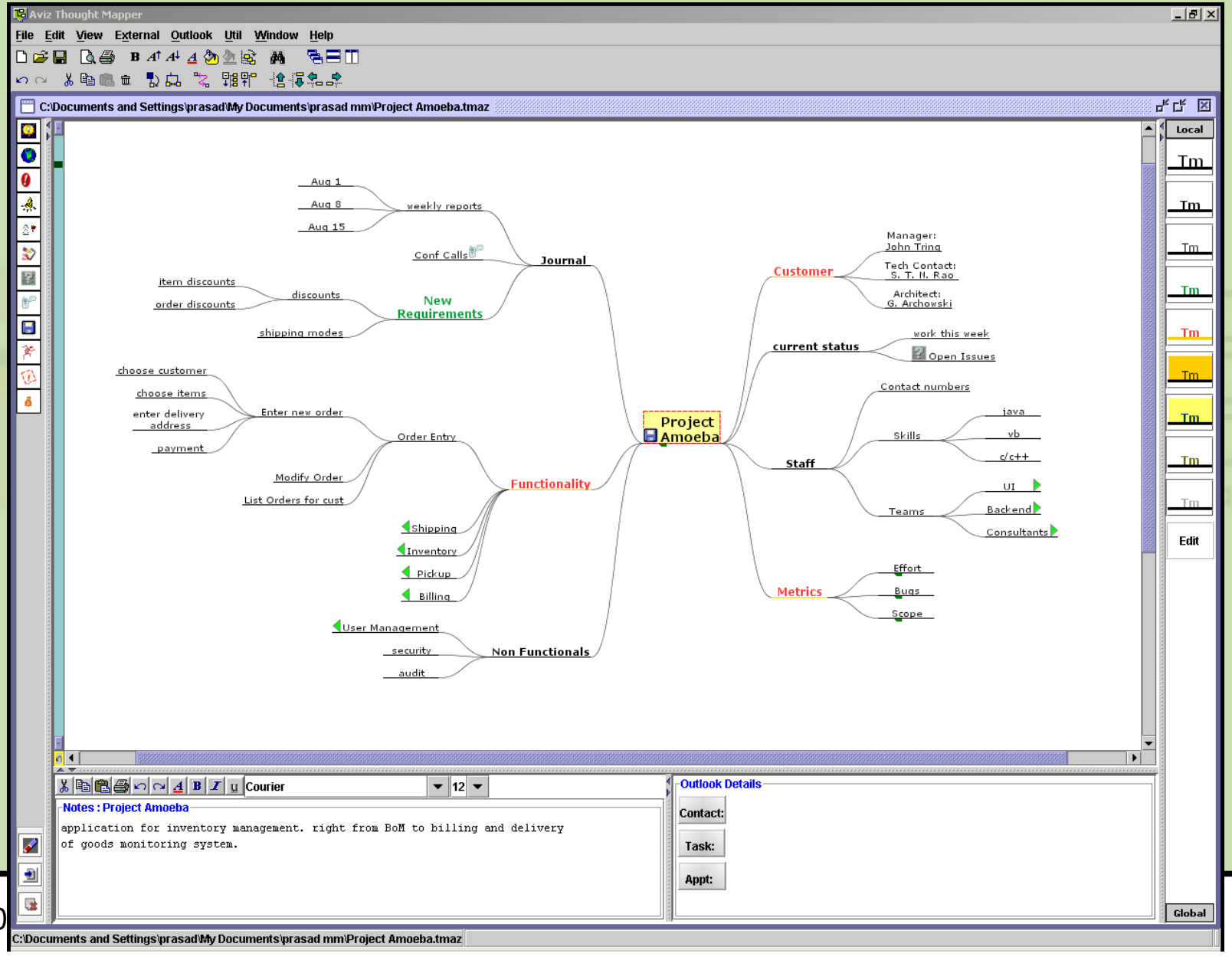
Exemplos de aplicações em Swing

- ZipCreator (http://www.zipcreator.com):



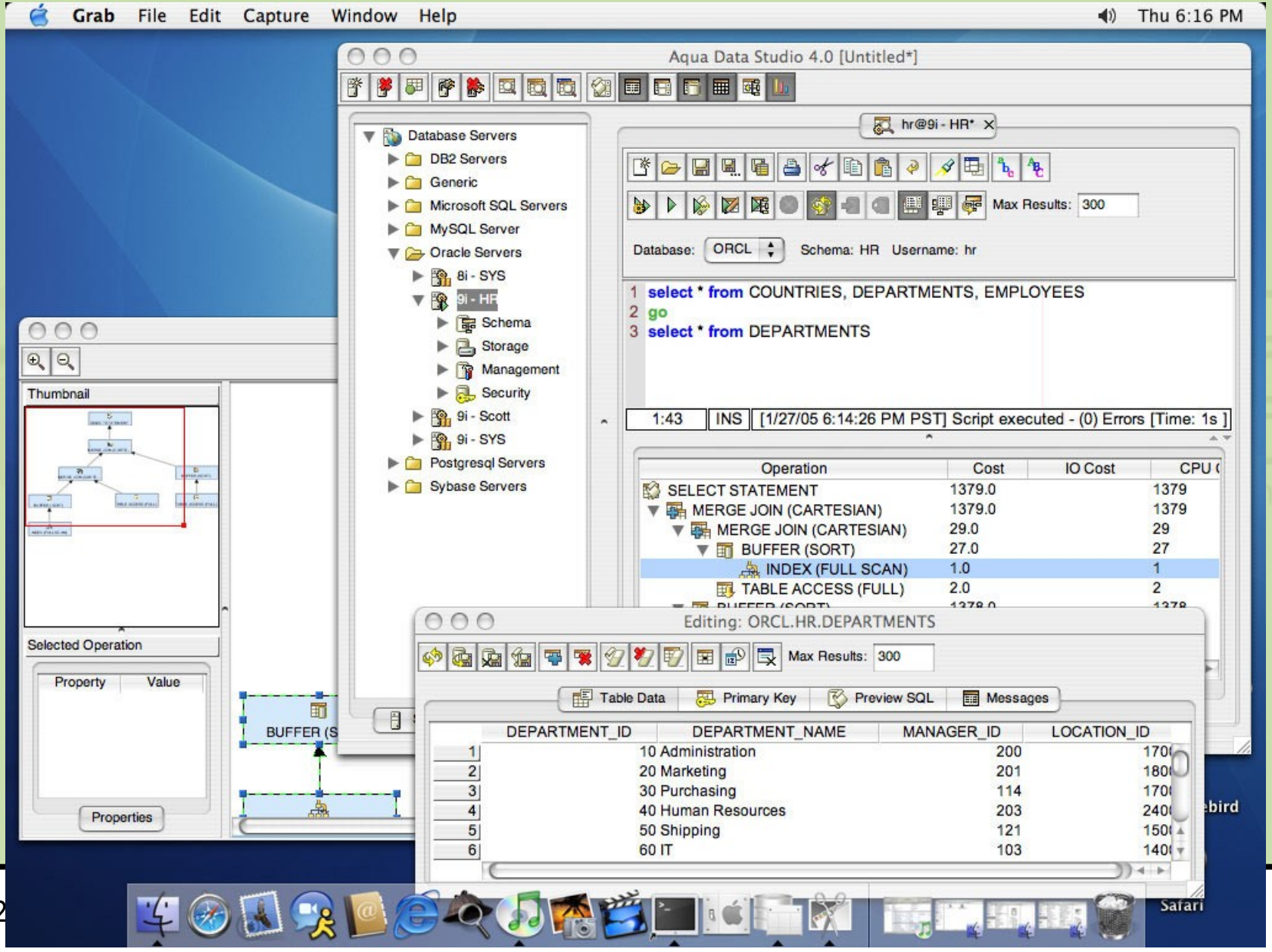
Exemplos de aplicações em Swing

- Thought Mapper (http://www.avizsoft.com):



Exemplos de aplicações em Swing

- Aqua Data Studio (http://www.aquafold.com):



Exemplos de aplicações em Swing

- E muitas outras...
  - Azureus (BitTorrent);
  - Programa de declaração de ajuste anual do IR;
  - NetBeans, JasperReports, Java Web Start, ...;
  - Etc.
- Veja mais em:
  - Swing Sightings: http://java.sun.com/products/jfc/tsc/sightings/
  - Java.com: http://www.java.com/

Foco do curso

- A tecnologia JFC/Swing e Java2D;
- Construção de interfaces gráficas sem auxílio de ferramentas de desenho;
- Construção de interfaces gráficas na ferramenta Matisse, integrada ao NetBeans.

Introdução ao acesso a dados

- Praticamente todo sistema precisa de armazenar dados em memória secundária (persistente);
- Possibilidades:
  - Diretamente em arquivos (texto ou binário);
  - Sistemas Gerenciadores de Banco de Dados (SGBD).
- SGBDs são mais usados em grandes sistemas de informação. Existem SGBDs:
  - Relacionais;
  - Objeto/Relacionais;
  - Orientados a objeto.

SGBD Relacionais

- Tecnologia criada nos anos 70;
- Forte base teórica – álgebra relacional;
- Indústria forte: Oracle, Microsoft, IBM e várias opções open-source;
- Padrão do mercado, mesmo depois de vários anos do surgimento dos SGBDs OO.

Acesso a um SGBDR

- Linguagens de programação oferecem bibliotecas de conexão com alguns SGBDR;
- Java tem a API JDBC (Java DataBase Conectivity):
  - Definição de interface genérica para acesso;
  - Implementação fornecida pelos fabricantes, por meio de *drivers*;
  - Vários níveis de compatibilidade com a API: 1 – 4.
- Java SE inclui:
  - A API JDBC (interfaces);
  - Uma implementação para ODBC.

A incompatibilidade de paradigmas

- Orientado a objetos x Relacional;
  - Granularidade, herança, identidade, associações, navegação no grafo de objetos.
- Opções para persistência:
  - Codificação manual de comandos SQL com JDBC;
  - Serialização;
  - Enterprise JavaBeans gerenciados por *container*;
  - SGBDs orientados a objetos;
  - Mapeamento objeto/relacional (ORM).

Mapeamento objeto/relacional

- ORM (*Object/Relational Mapping*);

  Persistência automatizada e transparente de objetos de um aplicativo Java para as tabelas em um banco de dados relacional, usando metadados que descrevem o mapeamento entre os objetos e o banco de dados.

- Frameworks: Hibernate, OJB, Torque, Castor, Cayenne, etc.

Razões para usar ORM

- Produtividade: elimina a necessidade de escrever código SQL;
- Manutenção: quanto menos linhas de código, maior a manutenibilidade;
- Desempenho: os criadores dos *frameworks* ORM entendem muito mais de BD do que a maioria dos desenvolvedores de aplicativos;
- Independência de fabricante: geração automática de vários dialetos de SQL.

Foco do curso

- A API JDBC para acesso à bancos de dados via consultas SQL;
- O uso do *framework* ORM Hibernate.

Revisão de Java Básico