UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

FERRAMENTA CONVERSORA DE INTERFACES GRÁFICAS – DELPHI2JAVA-II

FABRICIO FONSECA

BLUMENAU
2005

2005/1-16

FABRICIO FONSECA

FERRAMENTA CONVERSORA DE INTERFACES GRÁFICAS – DELPHI2JAVA-II

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciências da Computação — Bacharelado.

Prof. Mauro M. Mattos , Dr. - Orientador

BLUMENAU
2005

2005/1-16

FERRAMENTA CONVERSORA DE INTERFACES GRÁFICAS – DELPHI2JAVA-II

Por

FABRICIO FONSECA

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Mauro M. Mattos, Dr. – Orientador, FURB
Membro:	Prof. Maurício Capobianco Lopes, FURB
Membro:	Prof. Marcel Hugo, FURB

Blumenau, 12 de julho de 2005

Dedico este trabalho a todos os amigos, especialmente aqueles que me ajudaram diretamente na realização deste. A minha esposa, que soube compreender as horas alocadas no desenvolvimento do mesmo. Aos meus pais que sempre me apoiaram e incentivaram a estudar e lutar pelos meus objetivos.

AGRADECIMENTOS

A Deus, pelo seu imenso amor e graça.

À minha família, pela motivação a qual me fez ultrapassar os obstáculos.

À minha esposa, que soube me compreender nos momentos difíceis.

Aos meus amigos, pela companhia nesta caminhada.

Ao meu orientador, Mauro M. Mattos, por ter acreditado na conclusão deste trabalho.

RESUMO

Este trabalho descreve o desenvolvimento de uma ferramenta capaz de converter formulários desenvolvidos no ambiente Delphi em código Java equivalente, facilitando o processo de migração de aplicações da plataforma Windows para plataforma Java. Utilizando-se do *framework Swing* para a construção das interfaces gráficas, o código Java resultante é um código atualizado perante a plataforma J2EE, a qual disputa grande parte do mercado com a Microsoft .Net.

Palavras-chave: *Software* livre. Java.

ABSTRACT

This work describes the development of a tool capable to convert Delphi forms in equivalent Java code, facilitating the process of migration of the applications Windows platform for Java platform. Using itself of framework Swing for the construction of the graphical interfaces, resultant the Java code is a code brought up to platform J2EE, which dispute great part of the market with the Microsoft Net.

Key-Words: *Free software*. Java.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

AWT – *Abstract Windowing Toolkit*

J2EE – Java 2 *Enterprise Edition*

J2SDK - Java S*oftware Developement Kit*

JLCA – Java *Language Conversion Assistant*

RF – Requisitos funcionais

RNF – Requisitos não funcionais

XML – *eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Atualmente, as plataformas Java 2 *Enterprise Edition* (J2EE) e Microsoft .NET fornecem uma excelente infra-estrutura para o desenvolvimento de vários tipos de aplicações. Um dos principais institutos de pesquisas do mundo, o *Gartner Group* (CESAR, 2003), indica que as duas plataformas de desenvolvimento devem dividir a maior fatia do mercado durante alguns anos, cerca de 80%. Com isto, cada vez mais cresce o interesse de desenvolvedores e empresas em comparar as tecnologias oferecidas pelas duas plataformas para tomarem as suas decisões.

Tecnicamente, as tecnologias J2EE e .NET são bem parecidas. Ambas fornecem arquiteturas de componentes para a construção de sistemas distribuídos, oferecendo recursos similares para o desenvolvimento de aplicações internet e *web services*. Há também os componentes *Enterprise JavaBeans* para J2EE e .NET *Managed Components*, ambos para conexão com bancos de dados e baseados nos mesmos princípios de funcionamento (MACÊDO, 2003).

As duas tecnologias usam também o mesmo conceito de máquina virtual, em que as linguagens de programação são compiladas para um código intermediário (*bytecode* no Java e Microsoft *Intermediary Language* no .NET), com velocidades de execução muito próximas. Contudo, por trás das tecnologias, existem algumas diferenças fundamentais. A Máquina Virtual Java é utilizada para mapear uma mesma linguagem nas mais diversas plataformas. Isso permite que uma aplicação corporativa, desenvolvida originalmente em Windows, possa rodar sem alterações em sistemas operacionais, como Unix ou até em *mainframes*. Para integração com outros sistemas, J2EE oferece facilidades como Java *Connectors* (SIQUEIRA, 2004).

Para uma empresa que hoje utiliza a plataforma Windows e é usuária do ambiente de desenvolvimento Borland Delphi, a adoção da plataforma .NET seria a opção mais cômoda para migrar suas aplicações, isto porque, segundo Borland Software Corporation (2003), as novas versões do Delphi serão voltadas para esta plataforma.

Contudo, mas, supondo que esta mesma empresa deseje ter seu parque industrial voltado para a tecnologia J2EE, além da grande mudança estrutural, as aplicações nela desenvolvidas terão que ser reescritas na linguagem de programação Java.

Além de reescrever os códigos fontes das aplicações, todas as interfaces gráficas (formulários) teriam que ser redesenhadas na nova linguagem, o que, dependendo do ambiente de desenvolvimento a ser escolhido, implicaria em grande dispêndio de tempo e recurso tendo em vista que esses ferramentais não possuem as mesmas facilidades de desenvolvimento que o ambiente Delphi propicia.

Neste sentido, foi desenvolvida uma ferramenta capaz de converter automaticamente os arquivos que descrevem as estruturas de composição dos formulários desenvolvidos em ambientes Delphi (arquivos com extensão DFM) em códigos fonte na linguagem Java.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é desenvolver uma ferramenta para conversão de interfaces gráficas desenvolvidas em ambiente Delphi para a linguagem de programação Java.

Os objetivos específicos do trabalho são:

a) implementar a conversão de arquivos DFM para a linguagem Java;

b) implementar a conversão dos seguintes componentes de interface: *TBitBtn*, *TButton*, *TCheckBox*, *TComboBox*, *TEdit*, *TForm*, *TGroupBox*, *TLabel*, *TListBox*, *TMainMenu*, *TMemo*, *TMenuItem*, *TPageControl*, *TPanel*, *TPopupMenu*,

TProgressBar, *TRadioButton*, *TRadioGroup*, *TRichEdit*, *TSpeedButton*, *TSpinEdit*, *TStringGrid*, *TTabSheet*, *TToolBar* e *TToolButton*;

c) disponibilizar no projeto Java os eventos *onclick*, *onchange*, *onkeypress*, *onenter* e *onexit*, além dos eventos *onCreate* e *onDestroy* pertinentes ao *TForm*, *onPopup* pertinente ao componente *TPopupMenu* e *onCloseup* pertinente ao *TComboBox*, facilitando ao usuário introduzir código específico, isto porque a manipulação de eventos no Java é realizada de forma trabalhosa, tanto quanto a criação de interfaces gráficas.

1.2 ESTRUTURA DO TRABALHO

O capítulo dois apresenta a fundamentação teórica deste projeto, baseado na migração de softwares.

O capítulo três apresenta uma explanação sobre o desenvolvimento de interfaces com o usuário e tratamento de eventos no ambiente Delphi. O capítulo quatro fundamenta o desenvolvimento de interfaces com usuário e tratamento de eventos em Java.

O capítulo cinco descreve a ferramenta Delphi2Java, uma ferramenta comercial atualmente descontinuada, que implementa a mesma funcionalidade que a ferramenta desenvolvida neste projeto.

O capítulo seis descreve o desenvolvimento da ferramenta Delphi2Java-II, apresentando dois estudos de caso. Além disto, realiza o comparativo na conversão de interfaces gráficas entre as ferramentas Delphi2Java e Delphi2Java-II.

Por fim, no capítulo sete, são apresentadas as conclusões e sugestões para a continuidade deste trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Conforme citado anteriormente, atualmente, as plataformas J2EE e Microsoft .NET fornecem uma excelente infra-estrutura para o desenvolvimento de vários tipos de aplicações. Um dos principais institutos de pesquisas do mundo, o *Gartner Group* (CESAR, 2003), indica que as duas plataformas de desenvolvimento devem dividir a maior fatia do mercado durante alguns anos, cerca de 80%. Com isto, cada vez mais cresce o interesse de desenvolvedores e empresas em comparar as tecnologias oferecidas pelas duas plataformas para tomarem as suas decisões.

A realidade é que os softwares tornam-se obsoletos com a evolução das plataformas e não podem aproveitar toda a infra-estrutura oferecida como, por exemplo, a comunicação *web*, além de não se comunicarem com novos sistemas.

Segundo Filho (2003), este contexto faz com que as empresas entrem num dilema: manter os sistemas como estão ou reescrevê-los. Manter é um problema, pois os sistemas atuais não aproveitarão a infra-estrutura da plataforma, e reescrevê-los toma muito tempo e exige que o investimento original seja refeito.

Como uma solução paliativa está a migração de softwares, que pode se tornar uma opção atrativa e evita a defasagem dos sistemas, pois, conforme Filho (2003), migrar pode gerar uma economia de até 85% em relação aos investimentos em reescrever, isto porque a migração exige cerca de 25% do tempo de desenvolvimento original e o custo pode representar cerca de 15% a 20% do custo original, incluindo-se neste percentual o tempo de testes e consultoria.

Além do custo, migrar o software é aproveitar tudo o que já foi desenvolvido especificamente para a empresa, sem ter que começar tudo do zero, tornando o processo de desenvolvimento muito mais rápido.

Contudo, a tecnologia de desenvolvimento de interfaces com o usuário pode tornar-se o fator fundamental na decisão de migrar uma aplicação para a plataforma Windows .Net ou para a plataforma J2EE, já que o desenvolvimento de interfaces gráficas na maioria das vezes consome grande parte do tempo total de desenvolvimento.

2.1 TRABALHOS CORRELATOS

Enquanto a Sun Microsystems incentiva a utilização do Java, oferecendo cursos de programação orientada a objetos, a Microsoft disponibilizou a Java *Language Conversion Assistant* (JLCA) (MICROSOFT CORPORATION, 2002), ferramenta que permite aos desenvolvedores em linguagem Java fazer uma transição para a construção de web *services* em *eXtensible Markup Language* (XML), na plataforma .NET. A ferramenta automatiza o processo de migração da sintaxe da linguagem e da biblioteca de acesso ao código fonte escrito em Java, convertendo-o para a linguagem C#.

Outra aposta da Microsoft é o J# .NET, ferramenta que compila programas Java para a plataforma .NET. Inclui implementação .NET das bibliotecas de classes Java para tornar mais fácil esta atividade com o mínimo de reescrita (BINSTOCK, 2002).

Durante o levantamento bibliográfico, constatou-se que havia sido desenvolvida uma ferramenta comercial com comportamento similar ao pretendido neste trabalho, denominada Delphi2Java (WINSITE, 1997). Entretanto constatou-se também que a mesma havia sido descontinuada visto que, não há atualmente referência alguma na internet para o novo endereço de alguma companhia que esteja comercializando o software. A descrição de uma cópia *trial* é apresentada no capítulo 5.

3 COMPONENTES DE INTERFACE DO AMBIENTE DELPHI

Uma das principais características do ambiente Delphi é a facilidade de desenvolvimento de aplicações orientadas a formulários.

Os tópicos a seguir relatam sobre arquivos DFM, modelo de interfaces e de eventos do Delphi e os componentes que foram convertidos pela ferramenta (PEIL, 2000?):

3.1 ARQUIVO DFM

É um arquivo binário que contém as propriedades do desenho de um formulário contido em um projeto. As informações dos componentes de um formulário, suas propriedades, localização e nome são armazenados nele, inclusive quais procedimentos de eventos estão sendo utilizados.

Nem todas as informações estão presentes no arquivo DFM, pois algumas informações padrões num objeto passam a fazer parte do arquivo DFM apenas se sofrerem alterações. É o caso dos procedimentos de eventos, que passam a ser indicados no arquivo DFM apenas se o evento estiver sendo utilizado.

Cada linha do arquivo DFM armazena uma informação de algum componente. A linha iniciada pela palavra *object* refere-se à declaração de um objeto, sendo esta palavra seguida do nome do objeto e sua classe. Todas as linhas subseqüentes referem-se a informações do objeto descrito na linha *object*, salvo a linha com a descrição *end* que indica a finalização das informações do objeto. Caso uma nova linha iniciada com a descrição *object* anteceder a finalização do objeto indica que este novo componente está contido no objeto anterior.

Na figura 1 é apresentada a estrutura de um arquivo DFM contendo um formulário com um botão.

C:\Arquivos de programas\Borland\Delphi7\Projects\Unit1.dfm

Unit1

```
1  object Form1: TForm1
2    Left = 256
3    Top = 114
4    Width = 783
5    Height = 540
6    Caption = 'Form1'
7    Color = clBtnFace
8    Font.Height = -11
9    Font.Name = 'MS Sans Serif'
10    OnCreate = FormCreate
11    object Button1: TButton
12      Left = 40
13      Top = 48
14      Width = 75
15      Height = 25
16      Caption = 'Button1'
17    end
18  end
```

Figura 1 – Estrutura de um arquivo DFM

No exemplo demonstrado na figura 1 há as seguintes informações:

a) linha 1: a palavra *object* indica a declaração de um objeto, neste caso um objeto da classe *TForm1* denominado *Form1*;

b) linha 2: a palavra *Left* indica o deslocamento horizontal do objeto. Tomando-se como base um plano cartesiano, é a indicativa de qual ponto em relação ao eixo x está localizado o objeto, sendo que o ponto zero é o ponto inicial do objeto pai em relação ao mesmo eixo;

c) linha 3: a palavra *Top* indica o deslocamento vertical do objeto. Também se considerando como base um plano cartesiano, é a indicativa de qual ponto em relação ao eixo y está localizado o objeto, sendo que o ponto zero é o ponto inicial do objeto pai em relação ao mesmo eixo;

d) linha 4: a declaração *Width* indica, em pontos, qual a largura do objeto;

e) linha 5: a declaração *Height* indica, em pontos, qual a altura do objeto;

f) linha 6: a palavra *Caption* indica qual o título do objeto;

g) linha 7: a palavra *Color* indica qual a cor do objeto;

h) linha 8: *Font.Height* descreve qual o tamanho da fonte do objeto;

i) linha 9: *Font.Name* descreve qual a fonte utilizada do objeto;

j) linha 10: *OnCreate* indica que o evento de mesmo nome está sendo utilizado pela aplicação;

k) linha 11: a palavra *object* indica a declaração de um objeto, neste caso um objeto da classe *TButton* denominado *Button1*. Por esta declaração ser feita antes da finalização do objeto anterior, o novo objeto está contido no mesmo;

l) linha 12: idem ao item b, neste caso referente ao objeto *Button1;*

m) linha 13: idem ao item c, neste caso referente ao objeto *Button1;*

n) linha 14: idem ao item d, neste caso referente ao objeto *Button1;*

o) linha 15: idem ao item e, neste caso referente ao objeto *Button1;*

p) linha 16: idem ao item f, neste caso referente ao objeto *Button1;*

q) linha 17: a palavra *end* indica a finalização de um objeto, sempre o último objeto declarado e ainda não finalizado, neste caso o *Button1*;

r) linha 18: idem ao item q, neste caso finalizando objeto *Form1*.

3.2 MODELO DE INTERFACES DO DELPHI

Os aplicativos para sistema operacional Windows são usualmente baseados em janelas. Deste modo o Delphi disponibiliza a criação dessas janelas através do componente *TForm*, que é o já conhecido formulário. Embora todo formulário seja uma janela, a recíproca não é verdadeira.

Cada formulário pode armazenar uma série de componentes, os quais podem ser

escolhidos através de uma paleta na janela do Delphi. Para inserir um componente em um formulário, basta selecioná-lo na paleta e em seguida clicar sobre o formulário, colocando-o na posição desejada, como pode ser visto na figura 2.

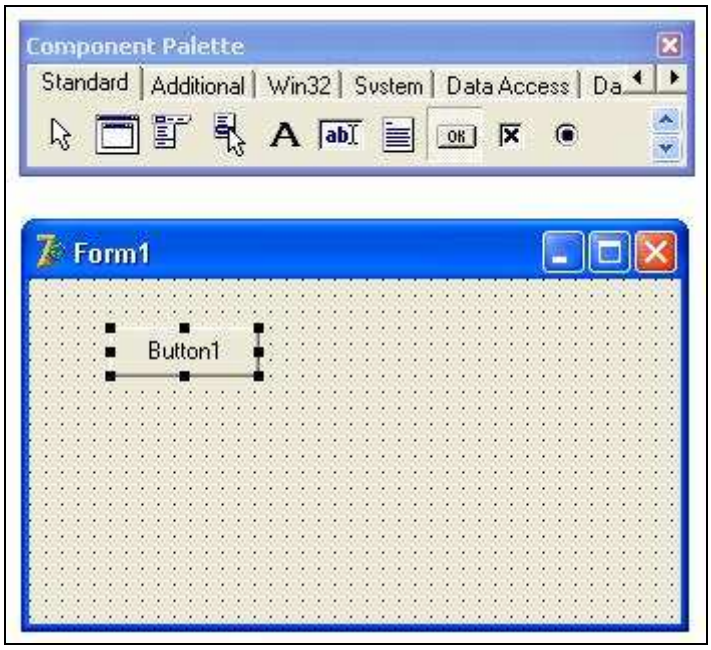

Figura 2 – Inserção de componente em um formulário Delphi

3.3 MODELO DE EVENTOS DO DELPHI

Ao pressionar o mouse sobre um formulário ou sobre algum componente, o sistema operacional informa o aplicativo sobre o evento, enviando uma mensagem a ele. Assim sendo, o Delphi reage ao receber uma notificação de evento, invocando um método de resposta a eventos adequado.

O Delphi define uma série de eventos para cada tipo de componente, ou seja, a lista de eventos de um formulário é diferente da lista de eventos de um botão, embora alguns eventos sejam comuns a ambos. Para visualizar a lista de eventos disponíveis para um componente, basta selecioná-lo e na paleta *Object Inspector* do Delphi posicionar-se na página *Events.* A figura 3 mostra os eventos disponíveis para um componente *TButton*.

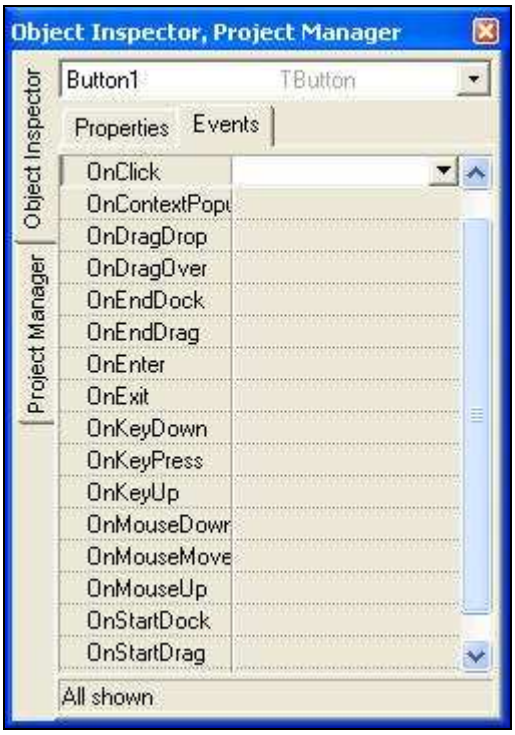

Figura 3 – *Object Inspector* do Delphi

Para associar um código específico para um determinado evento, basta que se dê um duplo clique com o botão esquerdo do mouse sobre a área branca que aparece ao lado do evento selecionado. O Delphi criará um procedimento no código do formulário e abrirá o código fonte da aplicação nessa posição, como pode ser observado na figura 4.

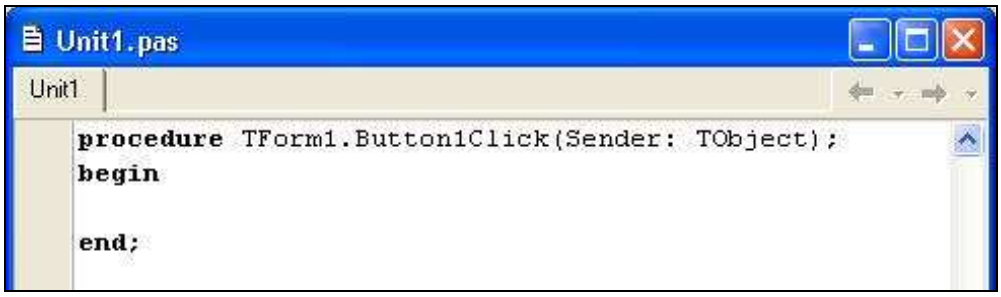

Figura 4 – Procedimento associado ao evento *onClick* de um *TButton* no Delphi

3.4 COMPONENTES DE INTERFACE DELPHI CONVERTIDOS

Embora o Delphi disponibilize algumas dezenas de componentes para construir formulários, a ferramenta desenvolvida neste projeto limitou-se a um grupo de componentes,

selecionando apenas os componentes mais usuais e que possuíam referência no Java, excetuando-se deste grupo os componentes para conexão com banco de dados.

Utilizando-se do mesmo critério, foram disponibilizados no código Java cinco eventos presentes na maioria dos componentes – *onclick*, *onchange*, *onkeypress*, *onenter* e *onexit* – além dos eventos *onCreate* e *onDestroy* pertinentes ao componente *TForm*, *onPopup* pertinente ao componente *TPopupMenu* e *onCloseup* pertinente ao *TComboBox*, facilitando a inserção de código específico pelo usuário.

Os eventos serão disponibilizados no código Java apenas se na aplicação Delphi o mesmo encontrar-se habilitado, ou seja, se existir um tratamento para o evento na aplicação original.

A figura 5 apresenta os componentes contemplados pela ferramenta Delphi2Java-II.

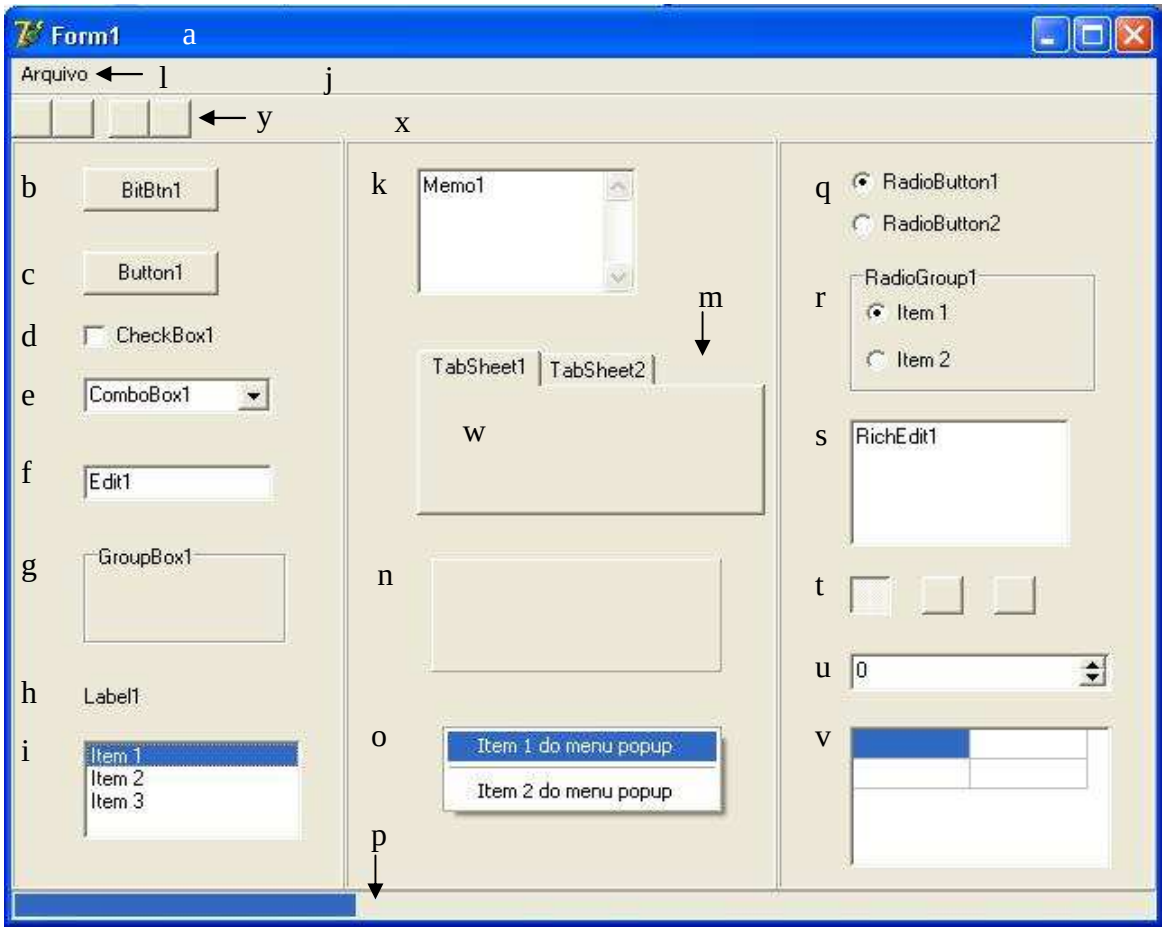

Figura 5 – Componentes Delphi contemplados pela ferramenta Delphi2Java-II

A seguir é feito um breve relato sobre a funcionalidade de cada componente apresentado na figura 5, seus principais eventos e sua correspondência no Java, ressaltando que independentemente de qual seja o evento principal do componente, caso alguns dos eventos citados anteriormente estiverem sendo utilizados, estes serão disponibilizados no código Java:

a) *TForm*: é o formulário, ou seja, uma janela da aplicação. Seus principais eventos são os de criação e destruição, *onCreate* e *onDestroy* respectivamente. No Java, o componente que mais se assemelha é o *JFrame*.

b) *TBitBtn*: é um botão com a opção de adicionar uma figura. Seu principal evento é o *onClick*, e o componente que mais se assemelha no Java é o *JButton*;

c) *TButton*: idem ao *TBitBtn*, mas sem a opção de adicionar figura;

d) *TCheckBox*: é uma caixa de seleção, a qual pode ser marcada ou desmarcada com um simples clique do mouse. Seu principal evento é o *onClick* e assemelha-se ao componente *JCheckbox* do Java;

e) *TComboBox*: uma caixa de seleção com uma lista de opções, a qual se abre ao clicar no componente e se fecha ao selecionar um item da lista. Seus principais eventos são *onClick*, *onChange* e *onCloseup*. O componente ao qual se assemelha no Java é o *JComboBox*;

f) *TEdit*: uma caixa de texto simples com uma única linha. Seu principal evento é o *onKeyPress* e assemelha-se ao componente *JFormattedTextField*.

   O *TEdit* possui uma propriedade que permite substituir os caracteres digitados por um caractere especial, como nos campos tipo senha. Neste caso seu correspondente no Java é o *JPasswordField*;

g) *TGroupBox*: é um painel utilizado para agrupar componentes. Pode ou não possuir título e é contornado por uma linha de baixo relevo. Possui uma lista de eventos,

mas em geral não são utilizados. No Java o componente que se assemelha é o *JDesktopPane*, sendo que deve ser utilizado o tipo de borda *TitledBorder*;

h) *TLabel*: utilizado para colocar um texto simples, em uma única linha, sem a opção de edição, como se fosse um rótulo. Possui uma lista de eventos, mas em geral não utilizados. No Java assemelha-se ao componente *JLabel*;

i) *TListBox*: é uma caixa com uma lista de opções a mostra, sendo que se o tamanho da lista for maior que a caixa, surgirá uma barra de rolagem. Seu principal evento é o *onClick*. No Java o componente que mais se assemelha é o *JList*, o qual deve ser associado a um *JScrollPane* para que seja criada a barra de rolagem;

j) *TMainMenu*: é o menu principal do formulário. Para cada formulário há apenas um *TMainMenu*. Seu único evento é o *onChange*. No Java corresponde-se com o componente *JMenuBar*;

k) *TMemo*: caixa para digitação de texto, podendo ou não ter barras de rolagem. Seu principal evento é o *onKeyPress*. No Java corresponde-se com o componente *JTextArea*, sendo que, assim como o *TListBox*, deve ser associado a um *JScrollPane* para que sejam criadas as barra de rolagem;

l) *TMenuItem*: são as opções do menu principal ou de menus *popup*, aqueles acionados com o botão direito do mouse. Seu principal evento é o *onClick* e no Java possui três correspondentes, sendo o *JMenuItem* para as opções terminais, *JMenu* para as opções que possuam sub opções e *JSeparator* para separadores de opções;

m) *TPageControl*: é um controlador de páginas, onde cada página é um componente do tipo *TTabSheet*. Seu principal evento é o *onChange* e assemelha-se ao componente *JTabbedPane* do Java;

n) *TPanel*: assim como o *TGroupBox*, é utilizado para agrupar componentes, mas não

possui título e sua borda não é identificada por uma linha em baixo relevo. Possui uma lista de eventos, mas em geral não são utilizados. No Java o componente que se assemelha é o *JDesktopPane*, sendo que deve ser utilizado o tipo de borda *CompoundBorder*, a qual deve ser composta por duas bordas do tipo *BevelBorder*;

o) *TPopupMenu*: são menus acionados com o botão direito do mouse. Cada componente pode ter apenas um *TPopupMenu*, mas estes podem estar associados a quantos componentes forem necessários. Seu principal evento é o *onPopup* e no Java assemelha-se ao componente *JPopupMenu*;

p) *TProgressBar*: é uma barra utilizada para mostrar o progresso da execução de algum processo. Seus eventos geralmente não são utilizados. No Java corresponde-se com o componente *JProgressBar*;

q) *TRadioButton*: em resumo é um botão em forma de círculo seguido de um rótulo. São utilizados geralmente em grupos, onde apenas um *TRadioButton* do grupo pode ser selecionado. Seu principal evento é o *onClick* e no Java corresponde-se com o componente *JRadioButton;*

r) *TRadioGroup*: visualmente é semelhante a um *TGroupBox*, sendo que neste caso possui apenas itens os quais se assemelham ao *TRadioButton*. Seu principal evento é o *onClick* e no Java não possui componente correspondente, sendo que sua implementação deve ser feita de modo manual, ou seja, cria-se um *TGroupBox* e para cada item cria-se um *TRadioButton*;

s) *TRichEdit*: Assemelha-se em muito ao *TMemo*, sendo que o *TRichEdit* possui algumas propriedades e eventos a mais que este;

t) *TSpeedButton*: é um botão sem a opção de foco. Possui a opção de manter-se pressionado e pode ser associado a um grupo com outros *TSpeedButton*, onde apenas um desses botões se manterá pressionado. Seu principal evento é o *onClick*

e no Java corresponde-se com o componente *JToggleButton*;

u) *TSpinEdit*: visualmente é semelhante a um *TEdit*, mas o *TSpinEdit* possui uma numeração inteira que pode ser incrementa ou decrementada utilizando-se de dois botões existentes no componente. Seus principais eventos são o *onClick* e *onChange* e no Java corresponde-se com o *JSpinner;*

v) *TStringGrid*: é uma caixa composta de células formando uma grade. Cada célula pode receber um texto. Seus principais eventos são *onClick* e *onKeyPress* e no Java o componente que mais se assemelha é o *JTable*;

w) *TTabSheet*: são as páginas do TPageControl. Possui uma lista de eventos, mas geralmente não são utilizados. No Java não existe um componente correspondente, sendo necessário criar um *JDesktopPane* e associá-lo ao *JTabbedPane* através do método *addTab();*

x) *TToolBar*: são as barras de tarefas. Possuem uma lista de eventos os quais geralmente não são utilizados, pois os eventos são associados aos botões da barra de tarefas. No Java corresponde-se com o componente *JToolBar;*

y) *TToolButton*: são os botões da barra de tarefas. Seu principal evento é o *onClick* e no Java não possui um correspondente direto, mas pode ser comparado a outros dois componentes, sendo o *JButton* para os botões e o *JSeparator* para os separadores de botões.

4 DESENVOLVIMENTO DE INTERFACES ORIENTADAS A FORMS EM JAVA

Um dos principais objetivos do Java é oferecer um ambiente de desenvolvimento independente de plataforma. A área das interfaces gráficas com usuário é uma das partes mais complicadas da criação de código portável, devido a API do Windows ser diferente da API do Mac, que por sua vez é diferente da API do *Presentation Manager* para OS/2 e assim por diante.

Para resolver este impasse, a Sun Microsystems optou por desenvolver uma API comum, que permite aos aplicativos em Java mesclarem-se uniformemente com os seus adjacentes. Esta API foi nomeada de *Abstract Windowing Toolkit*, ou simplesmente AWT. Posteriormente esta tecnologia ganhou uma extensão, denominada de *Swing*, que define componentes gráficos que utilizam exclusivamente Java, com funcionalidades e aparência independentes do sistema onde a aplicação é executada.

Apesar de toda a evolução do Java, ainda é notória a complexidade que o modelo de interface apresenta ao programador, a qual, dependendo da amplitude da aplicação, implica em grande dispêndio de tempo e recurso.

Os tópicos a seguir apresentam um breve relato sobre os pacotes java.awt e javax.*swing* e descreve o modelo de interface e eventos do Java.

4.1 PACOTE JAVA.AWT

A biblioteca de componentes AWT está definida através das classes do pacote java.awt e seus subpacotes, tais como java.awt.*event* e java.awt.*color*. As classes desses pacotes agrupam as funcionalidades gráficas que estão presentes desde as primeiras versões do Java, e operam tendo por base as funcionalidades do gerenciador de janelas e bibliotecas nativas do

sistema onde a aplicação é executada.

4.2 PACOTE JAVA.*SWING*

O *framework Swing*, associado ao pacote javax.*swing* e seus subpacotes, é uma extensão definida a partir de AWT que define componentes gráficos que utilizam exclusivamente Java, com funcionalidades e aparência independentes do sistema onde a aplicação é executada.

Neste pacote os componentes existentes na AWT sofreram melhoras visuais e alguns ganharam novos eventos, como o *PopupMenu* que recebeu os eventos de *popup* com o componente *JPopupMenu*. Os pacotes de eventos e cores foram mantidos na biblioteca AWT, pois não haveria necessidade de reescrevê-los. Já novos eventos, como os de controle de *popup*, foram desenvolvidos em um novo pacote, o javax.*swing.event.*

Na plataforma Java 2, *Swing* passou a fazer parte da distribuição padrão.

4.3 MODELO DE INTERFACE DO JAVA

A grande maioria dos ambientes para desenvolvimento Java não propicia as mesmas facilidades de desenvolvimento encontradas no ambiente Delphi, implicando em um maior domínio da linguagem em questão por parte do desenvolvedor.

No ambiente Delphi, os controles para criação de componentes visuais e eventos são abstraídos do desenvolvedor, facilitando e agilizando o desenvolvimento de aplicações, resultando em uma maior produtividade com custo menor de tempo. Por sua vez, o Java tem a vantagem de ser uma linguagem portável.

A figura 6 demonstra a criação de um formulário em Java, utilizando-se a extensão

javax.*swing.JFrame*, que é um subpacote do javax.*swing*.

```
public class uForm extends javax.swing.JFrame {

    public uForm() {
        // definir o tamanho do formulário
        // e posição na tela
        setBounds(316, 146, 300, 200);
    }

    //método principal do formulário
    public static void main(String args[]) {
        new uForm().show();
    }
}
```

Figura 6 – Código fonte de um formulário desenvolvido em Java

A figura 7 mostra a execução do código Java exibido na figura 6.

Figura 7 – Resultado da execução do código exibido da figura 6

Assim como no Delphi, cada formulário pode armazenar uma série de componentes, os quais devem ser criados e atribuídos ao formulário ou a algum componente de agrupamento, como um painel, por exemplo, através dos métodos *getContentPane().add() ou add()*, respectivamente. A figura 8 mostra como ficaria o código exibido na figura 6 acrescentando-se a construção de um botão.

```
public class uForm extends javax.swing.JFrame {
    //declarar o botao
    private javax.swing.JButton jBotao;

    public uForm() {
       // definir o tamanho do formulário
       // e posição na tela
       setBounds(316, 146, 300, 200);
       /* retirar o layout do form para
         permitir posicionar os componentes
         onde desejar */
       getContentPane().setLayout(null);

       //instanciar o botão
       jBotao = new javax.swing.JButton();
       //adicionar o botão ao formulário
       getContentPane().add(jBotao);
       //definir tamanho e posição no form
       jBotao.setBounds(120, 70, 81, 26);
       //definir o texto do botão
       jBotao.setText("Clique");
    }

    //método principal do formulário
    public static void main(String args[]) {
        new uForm().show();
    }
}
```

Figura 8– Adicionando um botão em um formulário Java

A figura 9 mostra a execução do código Java exibido na figura 8.

Figura 9 – Resultado da execução do código exibido na figura 8

Além do modelo de desenvolvimento Java demonstrado nas figuras 8 e 9 existem ferramentas Java com recursos visuais para desenvolvimento de formulários, mas estas em geral não apresentam as mesmas facilidades existentes no Delphi e aquelas que mais se assemelham necessitam de um maquinário com boa capacidade de processamento.

4.4 MODELO DE EVENTOS DO JAVA

Como citado na seção 4.3, o Delphi abstrai o controle de eventos dos componentes, deixando a cargo do desenvolvedor apenas implementar o tratamento do evento, que pode ser o clique de um botão, o fechamento de uma janela, entre outros.

O Java por sua vez disponibiliza duas bibliotecas para tratamento de eventos, sendo elas java.awt.*event* e javax.*swing.event*, sendo que uma não substitui a outra, mas sim complementa. Fica então a cargo do desenvolvedor utilizar essas bibliotecas, associando os eventos aos componentes e criando métodos para tratamento dos mesmos.

Na figura 10 é demonstrada a implementação necessária para utilização e tratamento do evento quando pressionado um botão.

```
public class uForm extends javax.swing.JFrame {

    private javax.swing.JButton jBotao;

    public uForm() {
        setBounds(316, 146, 300, 200);
        /*retirar o layout do form para permitir
          posicionar os componentes onde desejar*/
        getContentPane().setLayout(null);

        jBotao = new javax.swing.JButton();
        getContentPane().add(jBotao);
        jBotao.setBounds(120, 70, 81, 26);
        jBotao.setText("Clique");

        //associar o evento de pressão do botão
        jBotao.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                //chamada do método para tratamento do evento
                jBotaoActionPerformed(evt);

            }
        } );
    }

    //método para tratamento do evento de clique do botão
    private void jBotaoActionPerformed(java.awt.event.ActionEvent evt) {
        jBotao.setText("Clicou");
    }

    public static void main(String args[]) {
        new uForm().show();
    }
}
```

Figura 10 – Tratamento de eventos em Java

A figura 11 mostra a execução do código Java exibido na figura 10, após clicar no botão.

Figura 11 – Resultado da execução do código exibido na figura 10

4.5 COMPONENTES DE INTERFACE JAVA CONVERTIDOS

A figura 12 apresenta os componentes Java contemplados pela ferramenta Delphi2Java-II.

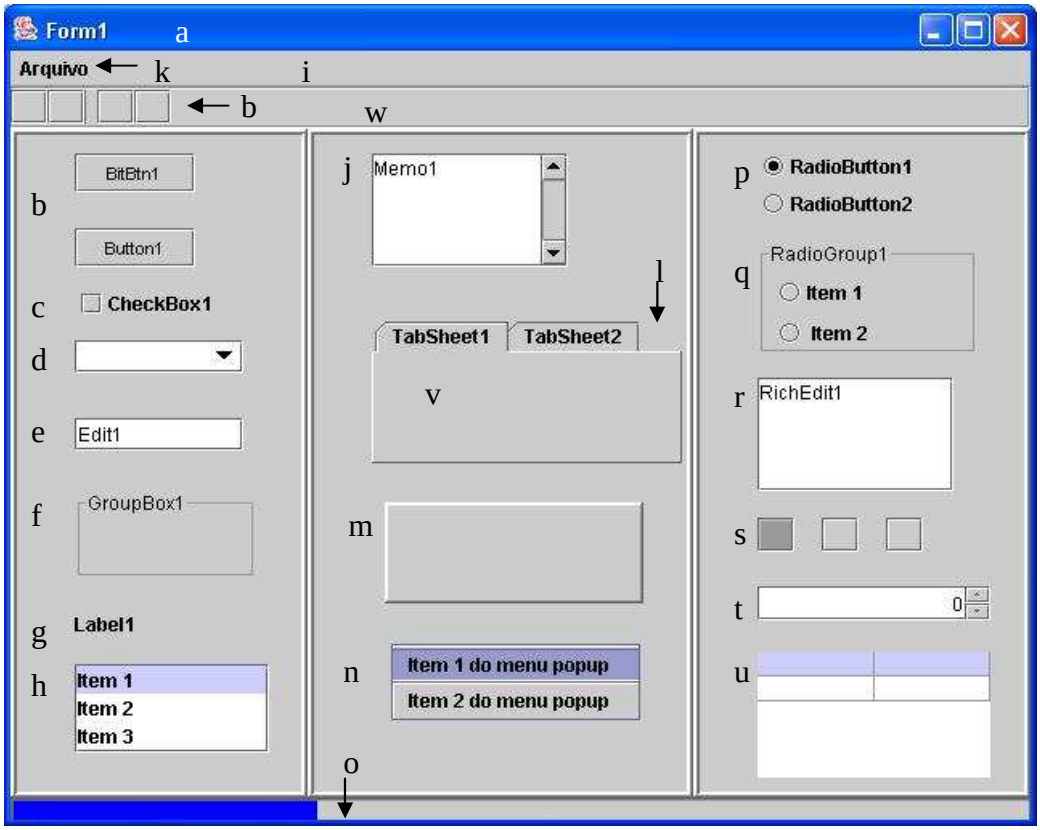

Figura 12 – Componentes Java contemplados pela ferramenta Delphi2Java-II

A seguir são apresentados os componentes da figura 12, sendo que suas funcionalidades e principais eventos são similares aos descritos para o Delphi na seção 3.4:

a) *JFrame*;

b) *JButton*;

c) *JCheckbox*;

d) *JComboBox*;

e) *JFormattedTextField*;

f) *JDesktopPane*, sendo que está utilizando o tipo de borda *TitledBorder*;

g) *JLabel*;

h) *JList*;

i) *JMenuBar*;

j) *JTextArea*;

k) *JMenu*;

l) *JTabbedPane*;

m) *JDesktopPane*, sendo que está utilizando o tipo de borda *CompoundBorder* a qual é composta por duas bordas do tipo *BevelBorder*;

n) *JPopupMenu*;

o) *JProgressBar*;

p) *JRadioButton*;

q) *JDesktopPane* contendo dois *JRadioButton*;

r) *JTextArea*;

s) *JToggleButton*;

t) *JSpinner*;

u) *JTable*;

v) *JDesktopPane* associado ao *JTabbedPane* através do método *addTab()*;

w) *JToolBar*.

5 DELPHI2JAVA

É uma ferramenta comercial, com propósitos semelhantes aos da ferramenta desenvolvida neste projeto.

Recentemente foi localizada uma cópia *trial* desta ferramenta (WINSITE, 1997), a qual se propõe a converter interfaces gráficas, componentes com conexão a banco de dados e códigos fontes além dos tratadores de eventos.

Segundo Spritemaster Development (1997?), esta ferramenta é capaz de converter *procedures* e *functions* e os componentes abrangidos pela ferramenta são:

a) *TBevel*;
b) *TBitBtn*;
c) *TBookmark*;
d) *TBrush*;
e) *TButton*;
f) *TCanvas*;
g) *TCheckBox*;
h) *TComboBox*;
i) *TDataSource*;
j) *TDBCheckbox*;
k) *TDBCombobox*;
l) *TDBEdit*;
m) *TDBGrid*;
n) *TDBListbox*;
o) *TDBMemo*;
p) *TDBNavigator*;

q) *TDBRadiogroup*;

r) *TDBText*;

s) *TEdit*;

t) *TFont*;

u) *TGroupBox*;

v) *TImage*;

w) *TIniFile*;

x) *TLabel*;

y) *TListBox*;

z) *TMainMenu*;

aa) *TMemo*;

bb) *TPanel*;

cc) *TPen*;

dd) *TPopupMenu*;

ee) *TPrinter*;

ff) *TQuery*.

gg) *TRadioButton*;

hh) *TRadioGroup*;

ii) *TScrollBar*;

jj) *TScrollBox*;

kk) *TShape*;

ll) *TStringGrid*;

mm) *TStringList*;

nn) *TStrings*;

oo) *TTable*;

pp) *TTimer*.

Em testes realizados com uma interface disponibilizada pela própria versão *trial*, sem conexões com banco de dados, constatou-se que a ferramenta converteu a interface gráfica e o código fonte, mas apresentou alguns pontos fracos:

a) utiliza apenas componentes da biblioteca AWT;

b) simula alguns componentes não existentes na biblioteca AWT através de um pacote denominado d2j, deixando assim o código fonte dependente de classes proprietárias e não distribuídas juntamente com o Java.

Entretanto constatou-se também que a mesma havia sido descontinuada visto que, não há atualmente referência alguma na internet para o novo endereço de alguma companhia que esteja comercializando o software. Isto explicaria então a utilização da biblioteca AWT em conjunto com o pacote d2j ao invés do *framework Swing.*

O detalhamento dos testes realizados podem ser visualizados na seções 6.4.1 e 6.4.2.

A figura 13 mostra a tela de abertura da ferramenta Delphi2Java.

Figura 13 – Tela de abertura da aplicação Delphi2Java

6 DESENVOLVIMENTO DO TRABALHO - DELPHI2JAVA-II

A motivação para o desenvolvimento do presente projeto surgiu como uma proposta do Prof. Mauro Mattos no sentido de desenvolver-se uma ferramenta que, a partir de formulários gerados no ambiente Delphi pudesse gerar código equivalente em Java tendo em vista facilitar a utilização de aplicações orientadas a formulários nas disciplinas introdutórias de programação orientada a objetos.

A versão de avaliação citada no capítulo anterior foi localizada somente no mês de março de 2005. Contudo, a ferramenta descrita neste trabalho vem sendo desenvolvida desde o início do segundo semestre de 2004.

O presente capítulo descreve a especificação do projeto e detalha aspectos importantes acerca da forma como os modelos de interface foram convertidos. Além disso, detalha a funcionalidade da ferramenta de conversão.

6.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

O sistema apresenta os seguintes requisitos funcionais (RF):

a) os arquivos convertidos devem preservar o nome dos objetos presentes no formulário original e suas principais propriedades, tais como largura, altura, cor, descrição ou texto (quando houver) e posição, entre outras propriedades peculiares a cada objeto. Para a criação dos objetos na linguagem Java, serão utilizadas classes da biblioteca *Swing* e para implementação de cores e eventos será utilizada a biblioteca AWT, ambas de propriedade da Sun Microsystems;

b) o aplicativo deve possuir duas opções de saída de dados, sendo uma apenas em memória, isto é, o código resultante da conversão será mostrado na tela do usuário

em uma área específica do aplicativo. Outra opção é que este código resultante da conversão seja gravado fisicamente, ou seja, serão gerados arquivos com a extensão java;

c) quando utilizada a opção para gravar o código fisicamente, o aplicativo deve gerar dois arquivos, sendo um para a criação dos componentes e atribuição das suas propriedades e outro, sendo uma extensão do primeiro, para manipulação dos eventos, facilitando a inserção de código específico pelo usuário, visto que um formulário pode conter muitos componentes, o que geraria um código Java muito extenso;
d) a ferramenta deve garantir ao usuário que o resultado é realmente uma cópia mais próxima possível do original desenvolvido em Delphi (respeitadas as configurações visuais de cada ambiente);
e) o sistema deve possibilitar que o usuário selecione vários arquivos para conversão.

O aplicativo deverá atender os seguintes requisitos não funcionais (RNF):

a) a ferramenta deve ser de simples manuseio, ou seja, as operações devem ser realizadas de uma forma simples e ágil, com um mínimo de necessidade de configurações adicionais, tornando o processo de conversão viável em um ambiente de produção;
b) os arquivos DFM apresentam a mesma estrutura independente da versão do Delphi, sendo assim, poderão ser convertidos formulários desenvolvidos em qualquer uma das versões do Delphi;
c) a versão Java utilizada será a 1.4, pois é uma das mais atuais e contempla a biblioteca *Swing*, a qual também está presente nas versões superiores a 1.2, compatibilizando o código resultante com essa versão.

6.2 ESPECIFICAÇÃO

Para o desenvolvimento dos diagramas foi escolhida a ferramenta JUDE-Community 1.5.2, criada por Eiwa System Management (2005), por ser gratuita, de fácil utilização e por disponibilizar vários recursos para o desenvolvimento de diagramas, não só de classes, mas também de seqüência, de caso de uso entre outros. Possibilita ainda a exportação dos diagramas para arquivos de imagem com extensão .JPG.

A estrutura da ferramenta desenvolvida neste projeto é apresentada nas seções 6.2.1 e 6.2.2.

6.2.1 Modelo estático

O sistema foi desenvolvido tendo-se como base cinco classes principais, sendo elas:

a) TMainObj: é a classe principal do sistema, onde foi implementada toda a estrutura necessária para gerar o código Java;

b) TMainObj_BK: a indicação BK no nome da classe provém da palavra *BackGround*, ou seja, a cor de fundo ou cor do objeto. A classe TMainObj_BK é uma classe derivada da classe TMainObj com a implementação de métodos para conversão de cores do ambiente Delphi para o ambiente Java. Tanto no ambiente Delphi quanto no ambiente Java as cores são manipuladas utilizando-se a paleta de cores *Red Green Blue* (RGB). No ambiente Java trabalha-se com uma escala numérica de 0 a 255 para cada uma dessas três cores. Já no Delphi a informação da cor utilizada por um componente é armazenada em modo hexadecimal, no formato “$00000000”, sendo que os três primeiros caracteres são descartados, e os seis

caracteres restantes é a representação em hexadecimal das cores azul, verde e vermelho, ou seja, BGR.

c) TMainObj_BK_TM: a sigla TM no nome da classe é o indicativo de tamanho. Esta é uma classe derivada da classe TMainObj_BK com a implementação de métodos para geração de código Java de objetos que possuam a opção de definição de tamanho;

d) TMainObj_BK_TM_FG: a sigla FG no nome da classe provém da palavra *ForeGround*, utilizada para indicar a cor da fonte. Esta é uma classe derivada da classe TMainObj_BK_TM, com a implementação de métodos para geração de código Java de objetos que possuam texto com a opção de definição da cor deste;

e) TMainObj_BK_TM_FG_CPT: a indicação CPT no nome da classe provém da palavra *Caption*, ou seja, título. Esta é uma classe derivada da classe TMainObj_BK_TM_FG, com a implementação de métodos para geração de código Java de objetos que possuam títulos ou rótulos.

Para cada componente do ambiente Java foi construída uma classe derivada de uma das cinco classes principais, como pode ser visualizado nas figuras 14 e 15.

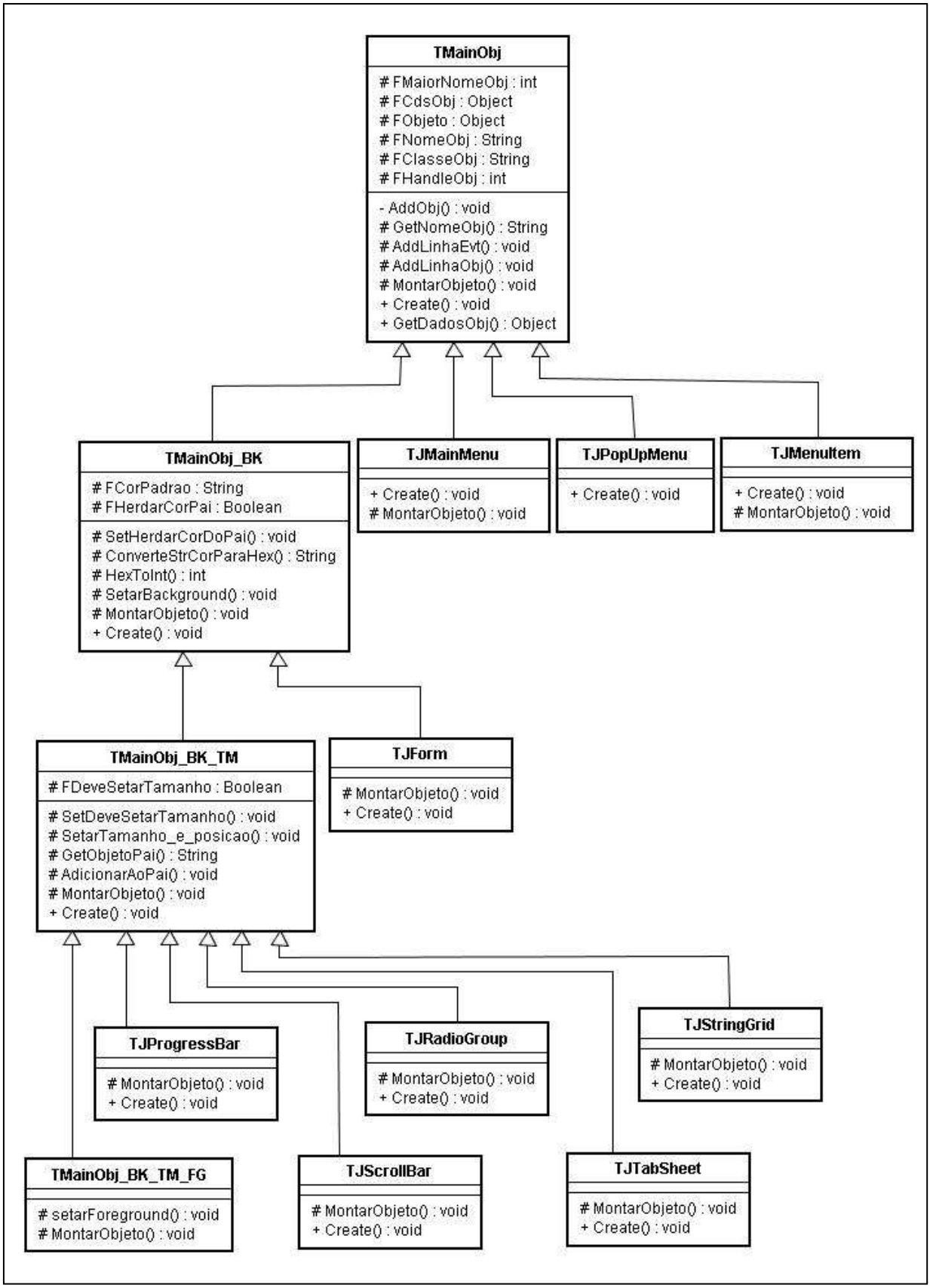

Figura 14 – Diagrama de classes da ferramenta Delphi2Java-II

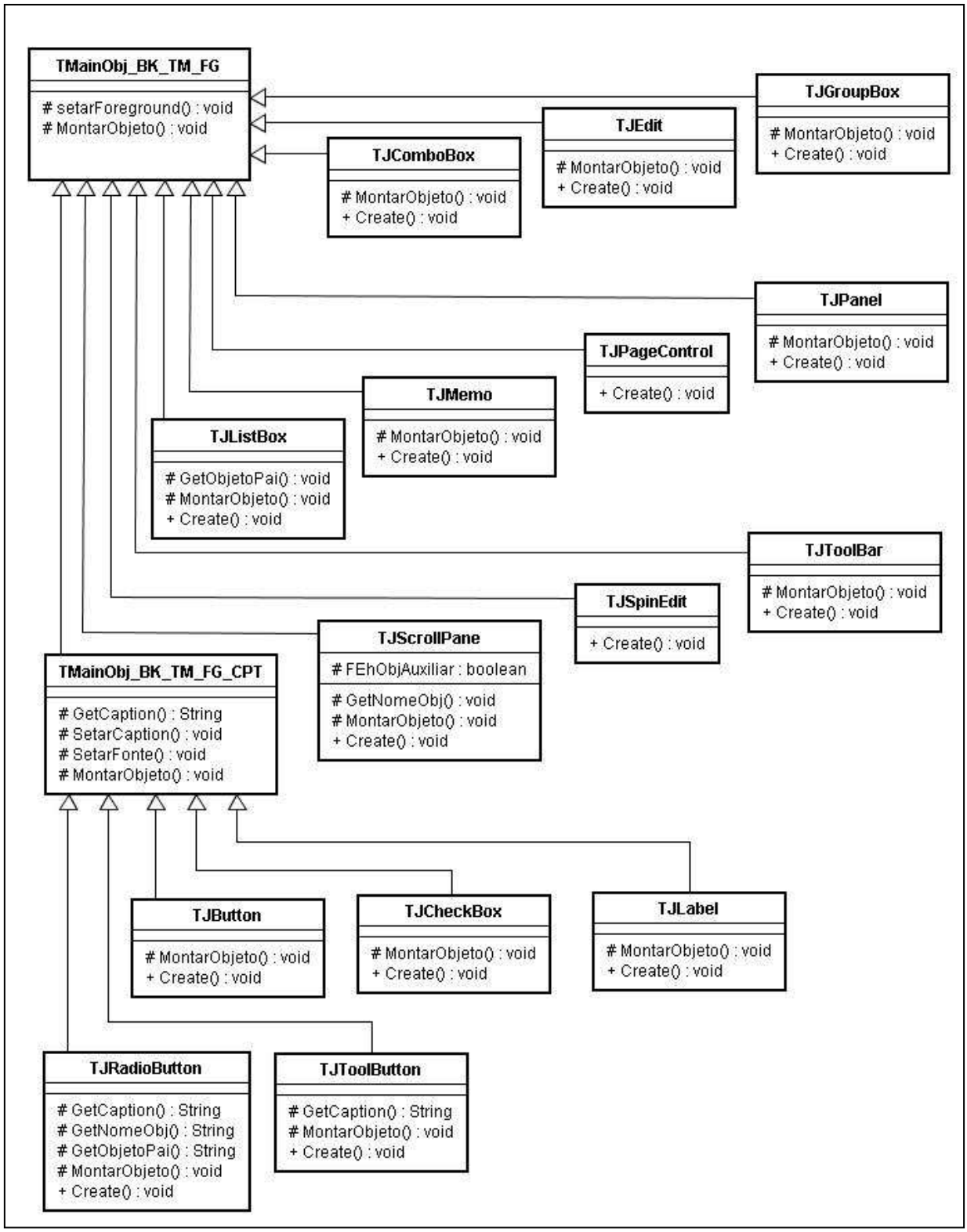

Figura 15 – Continuação do diagrama de classes da ferramenta Delphi2Java-II

6.2.2 Modelo dinâmico

A figura 16 apresenta, através de dois casos de uso, as duas opções de utilização da ferramenta Delphi2Java-II.

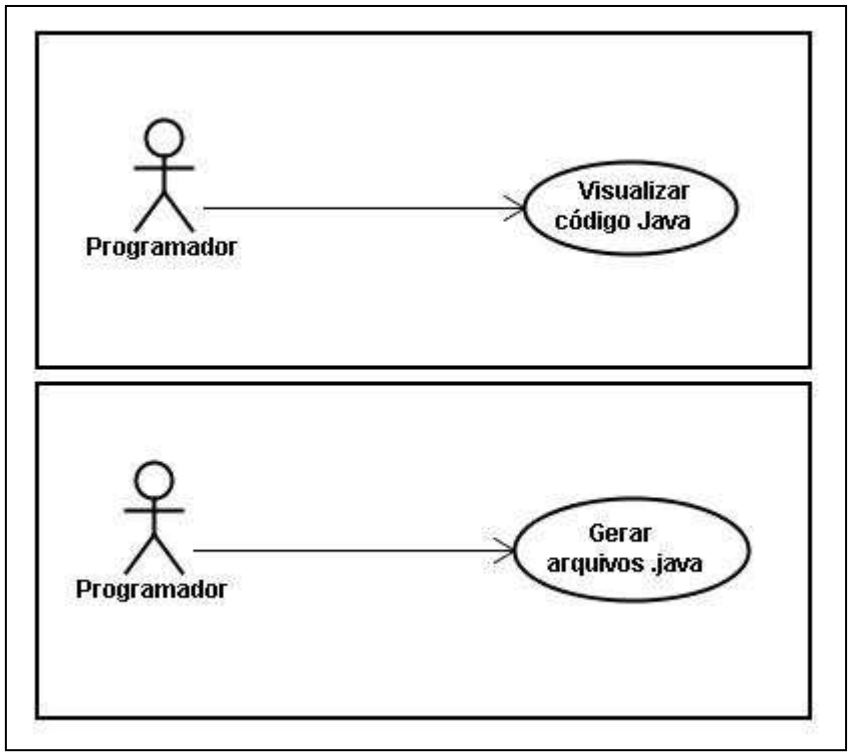

Figura 16 – Diagramas de caso de uso da ferramenta Delphi2Java-II

Nas figuras 17 e 18 são apresentados, através de diagramas de seqüência, as ações a serem tomadas para visualizar o código Java e gerar os arquivos com extensão java, respectivamente.

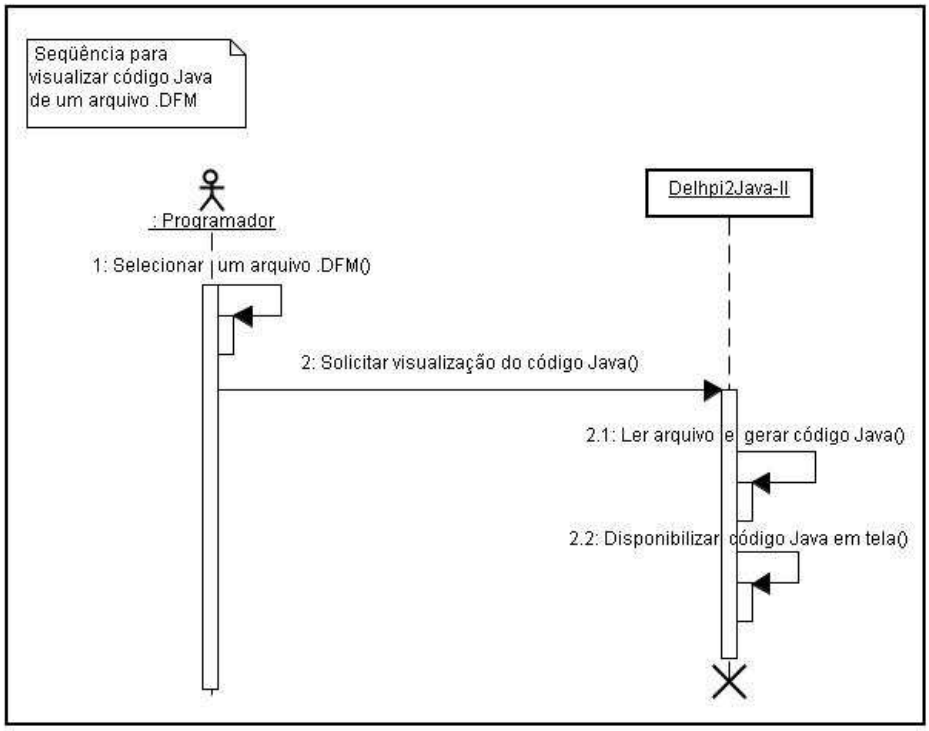

Figura 17 – Diagrama de seqüência para visualização do código Java

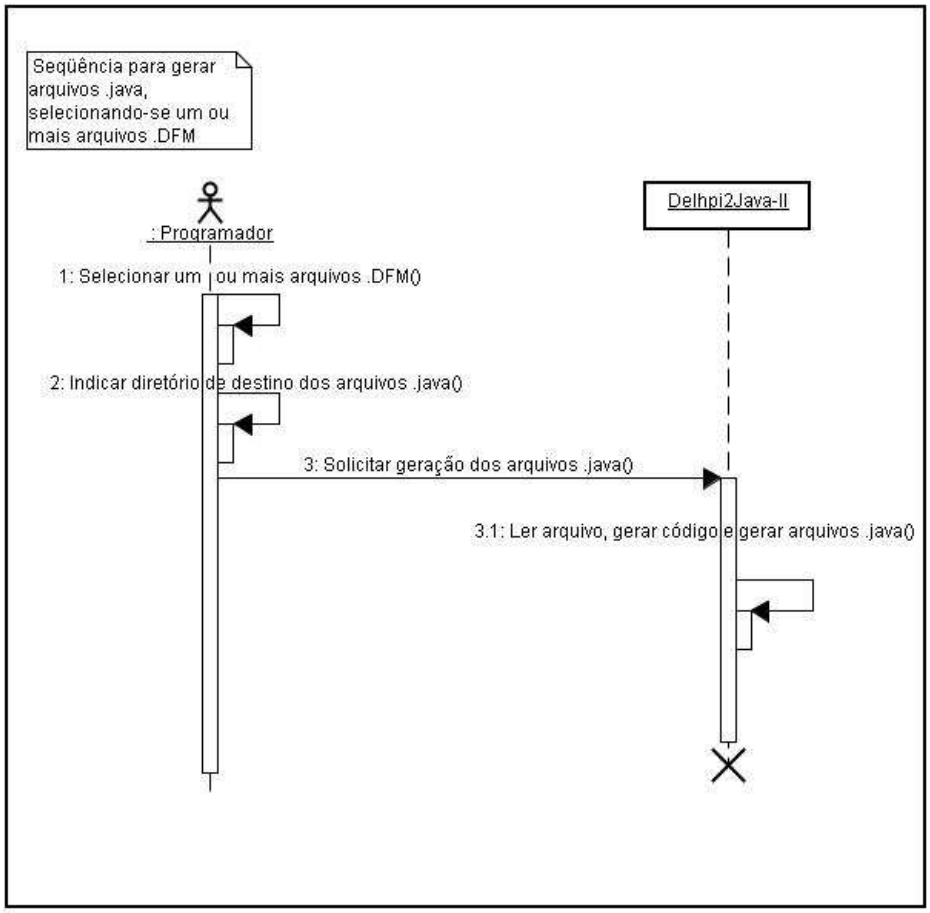

Figura 18 – Diagrama de seqüência para geração dos arquivos com extensão java

6.3 IMPLEMENTAÇÃO

O propósito da ferramenta Delphi2Java-II é, de uma maneira geral, ler arquivos com informações pertinentes a formulários desenvolvidos no ambiente Delphi e com base nessas informações gerar código Java que resulte numa interface semelhante a original.

O processo de construção e validação da ferramenta foi divido basicamente em quatro etapas, sendo iniciado pela leitura do arquivo, seguido da geração do código Java e geração dos arquivos com extensão java e por fim os testes. Estes processos foram executados de forma cíclica, ou seja, para cada componente foi implementada a leitura de suas informações, sua correspondência no Java e realizados testes de validação.

A ferramenta foi desenvolvida utilizando o ambiente de programação Borland Delphi 7 e para os testes com os códigos fonte Java resultantes, foi utilizado o ambiente JCreator LE 3.1 juntamente com o Java S*oftware Developement Kit* (J2SDK), ou simplesmente, ambiente de desenvolvimento Java, versão 1.4.2.

Por se tratar de um arquivo texto, a leitura do arquivo DFM é realizada de forma seqüencial, através dos métodos *AssignFile(), Reset(), ReadLn()*e *CloseFile()*. Na figura 19 é mostrado parte do código fonte que realiza a leitura deste arquivo.

```
AssignFile(ArqDfm, Arquivo);{associar o arquivo DFM.
                            (A variável Arquivo contém o nome do arquivo
                            com o caminho do diretório, ex.: C:\Teste\unit1.dfm)
Reset(ArqDFM); //abrir o arquivo DFM
while not eof(ArqDFM) do begin //enquanto não for o final final do arquivo
  Readln(ArqDfm, dadosLinha);  //ler a linha do arquivo
  //se for a 1ª linha, então verificar se está declarado um objeto
  if (slArquivo.Count = 0) and (pos('OBJECT',UpperCase(dadosLinha)) =0 ) then begin
    MessageDlg('Arquivo corrompido: '+Arquivo, mtInformation, [mbOK], 0);
    exit;
    end;
  //se for a linha da declaração do objeto então guardar o nome e a classe
  if (UpperCase(copy(dadosLinha,1,6)) = 'OBJECT')  then begin
    VerificarMultiplasLinhas;
    {como está lendo um novo objeto, caso haja outro objeto lido,
    então despejar as informações no TClientDataSet}
    if slArquivo.Count > 0 then
      AddObjeto;
    //guardar o nome
    nomeObj := copy(dadosLinha,8,length(dadosLinha));
    //guardar o nome da classe
    nomeClasse:= UpperCase(Trim(copy(nomeObj,pos(':', nomeObj) + 2, length(nomeObj))));
    nomeObj := Trim(copy(nomeObj,1, pos(':', nomeObj) - 1));
```

Figura 19 – Código fonte demonstrando a leitura de um arquivo DFM

Como apresentado na seção 3.1, cada linha do arquivo DFM possui uma informação referente ao objeto, ou seja, se é a declaração do componente, de seus atributos ou a descrição de finalização do mesmo. Após lidas as propriedades do objeto, essas informações são adicionadas a um componente denominado *TClientDataSet*, o qual se assemelha ao componente *TTable*, mas sem a conexão com banco de dados.

Após toda a leitura dos objetos, o componente *TClientDataSet* é percorrido como se fosse um *array* e, para cada componente é verificado sua classe e se esta possui a conversão implementada na ferramenta Delphi2Java-II. Caso a classe do componente esteja na lista das classes convertidas, é gerado um código Java para criação deste componente com as características originais, senão o objeto é adicionado à área de *log* dos componentes não convertidos. Este código é então adicionado a um componente do tipo *TRichEdit*, o qual ao final do processo conterá todo o código Java gerado. Parte do código fonte que realiza estas operações pode ser vista na figura 20.

```
cdsObjetos.First; //ir para o 1º registro do TClientDataSet
while not cdsObjetos.Eof do begin //executar um loop até o último registro
  //buscar o indexador da classe do objeto
  case GetIndexClasse(cdsObjetos.FieldByName('CLASSE').AsString) of
     //se for um formulário (TForm)...
    idxTform : with TJForm.Create(cdsObjetos, MaxEspaco) do begin //criar o objeto
                 LancarDados(GetDadosObj); //despejar as informações no TRichEdit
                 Free; //destruir o objeto
                 end;
    //se for um painel (TPanel)...
    idxTpanel:  with TJPanel.Create(cdsObjetos, MaxEspaco) do begin //criar o objeto
                  LancarDados(GetDadosObj);  //despejar as informações no TRichEdit
                  Free;//destruir o objeto

  ...


  else {se a classe do objeto não está sendo tratada na conversão
        então adicionar o objeto no log dos componentes não convertidos}
      mmLog.Lines.Add(nomeUnit +
                      ' - Objeto: '+cdsObjetos.fieldByName('NOME').AsString +
                      ' - Classe: '+cdsObjetos.fieldByName('CLASSE').AsString);
    end;//end do "Case GetIndexClasse..."

  //passar para o próximo registro
  cdsObjetos.Next;
  end;
```

Figura 20 – Código fonte demonstrando a geração do código Java

E por fim, a figura 21 mostra a criação dos arquivos com extensão java, sendo um arquivo contendo o código Java para a criação dos componentes e outro com a implementação dos eventos. Para tanto são utilizados os métodos *AssignFile(), ReWrite(), WriteLn() e CloseFile()*.

uMain.pas

uMain

```
procedure TFrmConversorJava.Exportar(Arquivo: String);
var
  sDir: String;
  arq : TextFile;
begin
  sDir := GetDiretorio;
  barraStatus.Panels[0].Text := 'Gerando arquivo '+Arquivo+ '.java';
  //gerar unit com a criação do objetos
  if reObjetos.Lines.Count > 0 then begin
    assignFile(arq, sDir + Arquivo + 'FRM.java');
    Rewrite(arq);
    Writeln(arq, reObjetos.text);
    CloseFile(arq);

    //gerar unit com a implementação dos eventos
    if opGerarEvt.Checked then begin
      assignFile(arq, sDir + Arquivo + '.java');
      Rewrite(arq);
      Writeln(arq,reEventos.text);
      CloseFile(arq);
      end;
    end;

  barraStatus.Panels[0].Text := 'Arquivo gerado';
end;
```

Figura 21 – Criação dos arquivos com extensão java

6.3.1 Operacionalidade da implementação

A ferramenta foi desenvolvida objetivando uma utilização prática, sem necessidade de configurações adicionais, para tanto foi utilizado apenas um formulário, o qual agrupa as informações necessárias para o processo de conversão.

A figura 22 apresenta o formulário da aplicação Delphi2Java-II, o qual recebeu marcadores alfabéticos a fim de facilitar a identificação de suas funcionalidades.

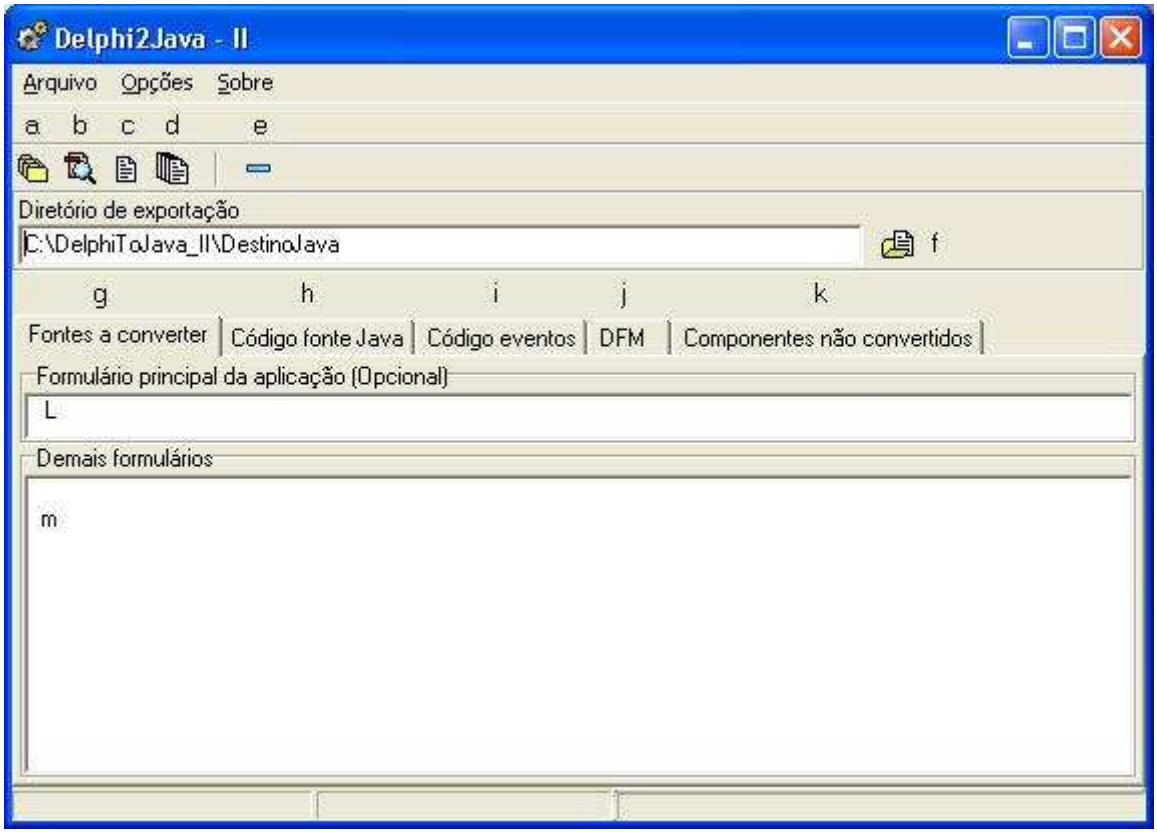

Figura 22 – Ferramenta Delphi2Java-II

O formulário da ferramenta Delphi2Java-II, apresentado na figura 22, possui as seguintes funcionalidades:

a) botão utilizado para selecionar os arquivos para conversão;

b) botão utilizado para converter um arquivo selecionado e disponibilizar o código Java resultante para visualização;

c) botão com a função de converter um arquivo selecionado e gerar os arquivos com extensão java;

d) botão com a função de converter todos os arquivo da lista, itens k e l, e gerar arquivos com extensão java;

e) botão utilizado para limpar a lista dos arquivos selecionados, itens k e l;

f) botão utilizado para selecionar o diretório onde serão gravados fisicamente os arquivos com extensão java;

g) página onde estão os arquivos selecionados para conversão;

h) página utilizada para visualizar o código Java pertinente a criação dos objetos;

i) página utilizada para visualizar o código Java pertinente ao tratamento de eventos;

j) página para visualização do arquivo DFM após conversão do arquivo selecionado;

k) página para visualizar os componentes não convertidos;

l) área utilizada para indicar qual o formulário principal da aplicação, pois este deve receber o comando *System.exit*(0) no evento de saída do formulário, o qual fechará aplicação e não somente o formulário. Para colocar um arquivo nesta área basta executar um duplo clique sobre um arquivo presente na lista de arquivos citada no item l. Para retornar o arquivo para a lista de arquivos (item l), basta executar um duplo clique sobre o arquivo;

m) lista de arquivos a serem convertidos. Para adicionar itens a esta lista basta utilizar o botão citado no item a.

6.4 ESTUDO DE CASO

Tendo em vista validar a ferramenta, foram desenvolvidos em Delphi uma série de exemplos de interfaces e os mesmos foram convertidos através do Delphi2Java-II. Dois desses teste estão descritos nas seções 6.4.1 e 6.4.2.

6.4.1 TESTE COM EXEMPLO DA VERSÃO TRIAL DO DELPHI2JAVA

A figura 23 mostra uma interface desenvolvida em Delphi, disponibilizada pela versão *trial* da ferramenta Delphi2Java. A figura 24 demonstra a conversão desta interface pela versão *trial* da ferramenta Delphi2Java e a figura 25 demonstra a conversão desta mesma interface pela ferramenta Delphi2Java-II. A fim de facilitar a comparação entre as figuras foi

adicionada uma numeração ao lado dos componentes.

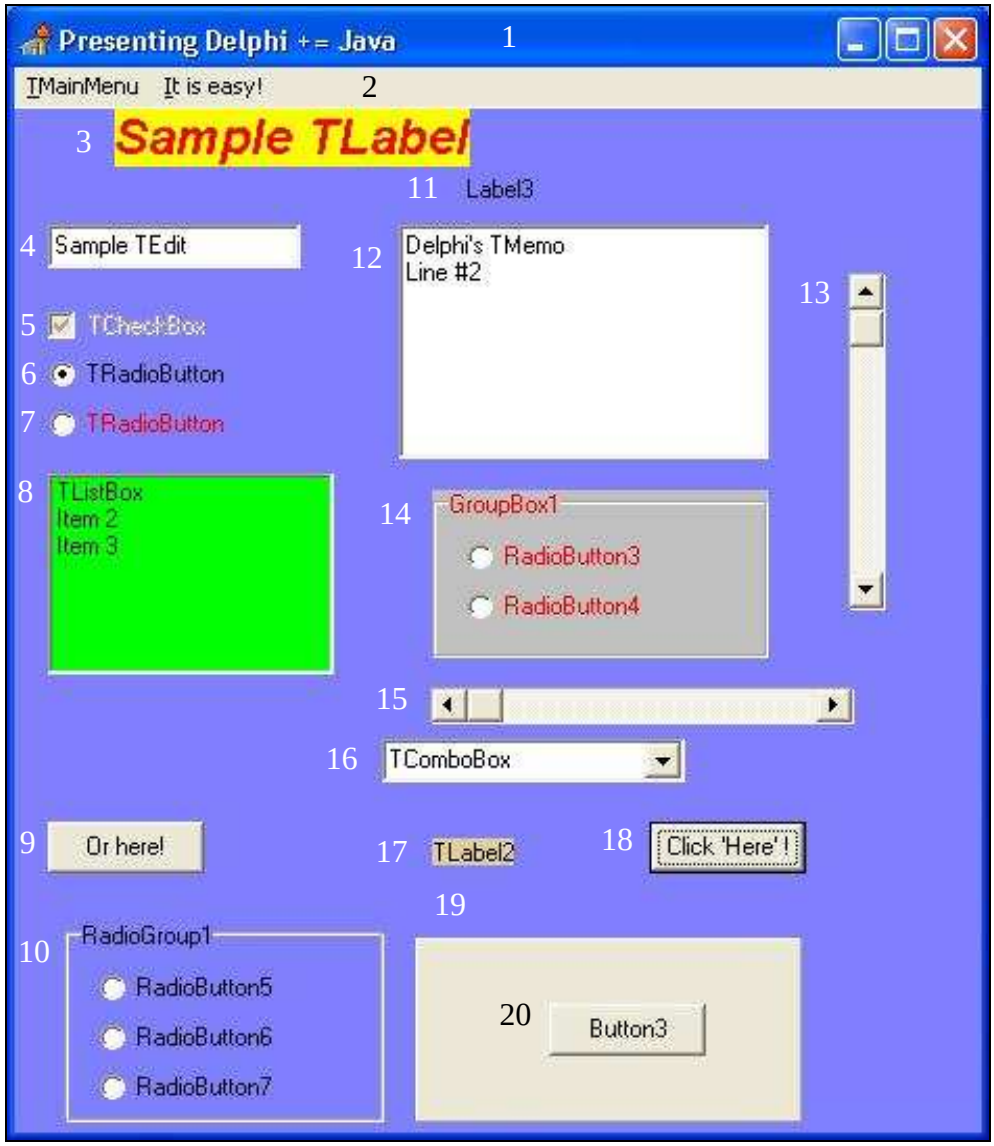

Figura 23 – Interface Delphi disponibilizada pela ferramenta Delphi2Java

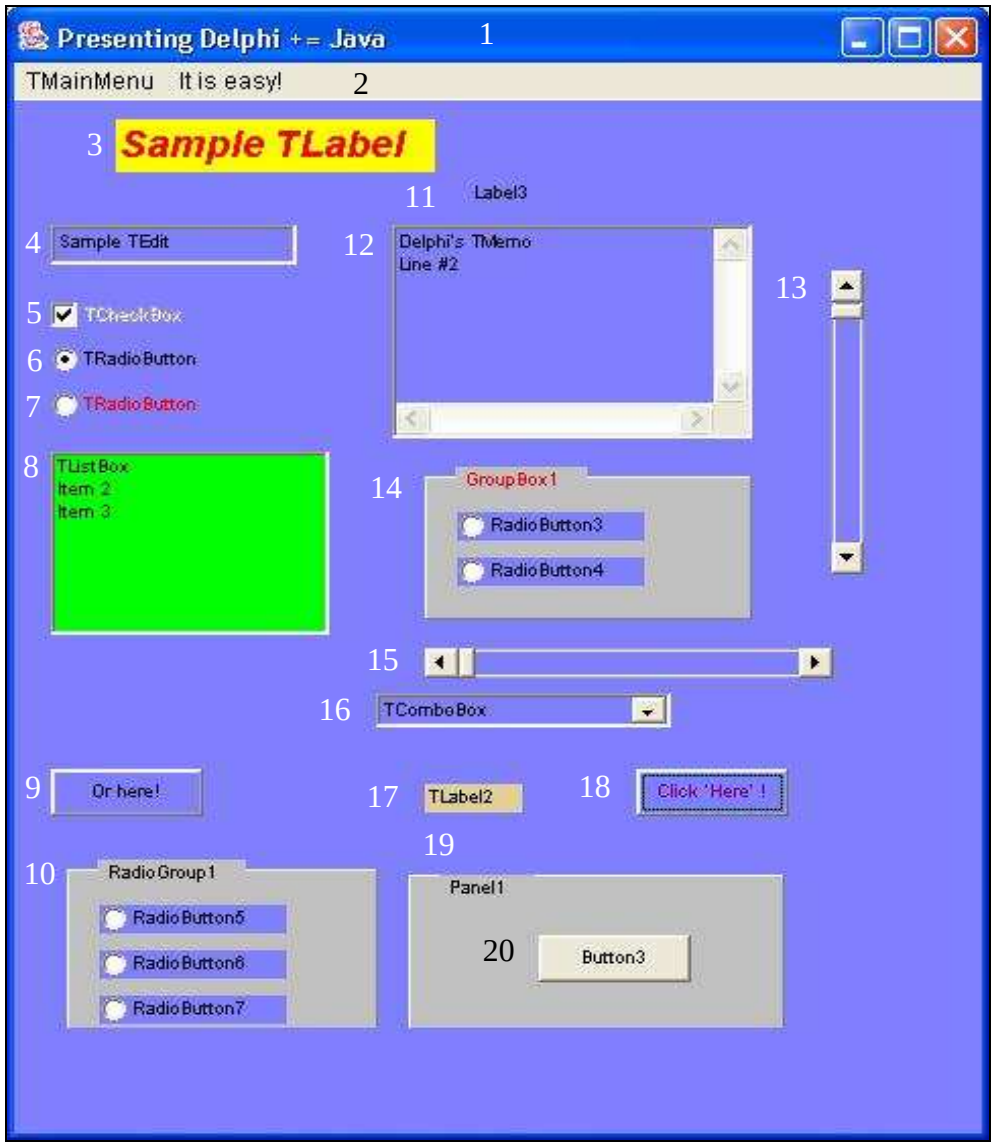

Figura 24 – Interface Java convertida pela versão *trial* do Delphi2Java

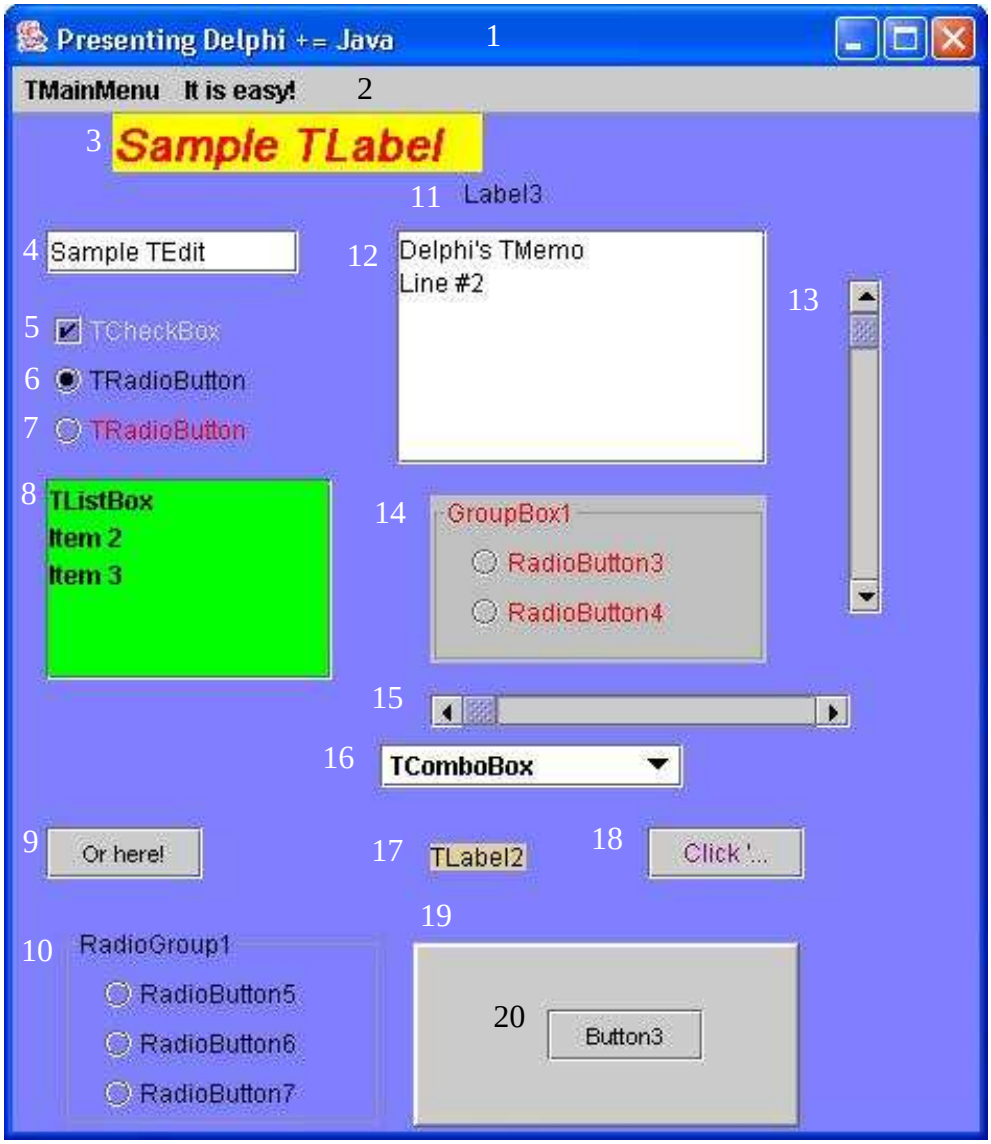

Figura 25 – Interface Java convertida pela ferramenta Delphi2Java-II

6.4.1.1 Avaliação das conversões

Comparando as figuras 24 e 25 nota-se que a figura 24 apresentou maiores disparidades em relação ao formulário original apresentado na figura 23. Os componentes 10, 14 e 19 apresentaram-se com deformidades na borda, já os componentes 4, 9, 12, 13, 15, 16 e 18 apresentaram falha na conversão das cores, assumindo a cor do formulário.

Já a figura 25 apresentou diferenças nos componentes 2 e 16, os quais aparecem com fonte em negrito, mas isto é em decorrência das configurações padrões dos componente

JMenuBar e *JComboBox*. Também se nota diferença no tamanho da letra do componente 8, sendo que isto ocorre porque no ambiente Java utiliza-se um tamanho de fonte maior que a utilizada no Delphi, logo a ferramenta tenta corrigir esta diferença no momento da conversão, sendo que para alguns componentes esta correção torna o tamanho da fonte maior quando comparado com o formulário original.

Um dos requisitos da ferramenta apresentados na seção 6.1 é a geração de dois arquivos com extensão java para cada formulário, sendo um com a declaração e criação dos objetos e outro arquivo com o tratamento dos eventos. As figuras 26 e 27 apresentam parte do código Java gerado pela ferramenta Delphi2Java-II.

```
1  public class tform1FRM extends javax.swing.JFrame {
2
3      public tform1FRM() {
4          initComponents();
5          setBounds(341, 218, 471, 548 ); //define o tamanho do formulário
6  }
7
8      private void initComponents() {
9
10         /////
11         // setar o título da janela
12         setTitle("Presenting Delphi += Java");
13         });
14         getAccessibleContext().setAccessibleName("tform1FRM Frame");
15
16         Form1                    = new javax.swing.JDesktopPane();
17         Label1                   = new javax.swing.JLabel();
18         Label2                   = new javax.swing.JLabel();
19         Label3                   = new javax.swing.JLabel();
20         Button1                  = new javax.swing.JButton();
21         Edit1                    = new javax.swing.JFormattedTextField();
22         Memo1                    = new javax.swing.JTextArea();
23         ScrPnMemo1               = new javax.swing.JScrollPane();
24         CheckBox1                = new javax.swing.JCheckBox();
25         GRPForm1                 = new javax.swing.ButtonGroup();
26
```

Figura 26 – Código Java com a declaração e criação dos objetos

```
public class tform1 extends tform1FRM {

    public  tform1() {
        CriarEventos();

    }
    private void CriarEventos() {
    } //fim do método CriarEventos


    ////////////////////////////////////////
    ///// Área para inserção de código /////
    ////////////////////////////////////////



    //método principal (cria e abre formulário)
    public static void main(String args[]) {
        new tform1().show();
    }


}

```

Figura 27 – Código Java para tratamento dos eventos

Em nível de comparação, é apresentado na figura 28 o código Java gerado pela ferramenta Delphi2Java, onde é utilizado um pacote denominado d2j, o qual é distribuído com a ferramenta Delphi2Java, deixando assim a aplicação dependente de um pacote não liberado juntamente com o Java.

```

import java.awt.*;
import java.io.*;
import java.lang.*;
import java.util.*;
import d2j.dlabel;
import d2j.dprop;
import d2j.showmsg;
import d2j.showmsg2;
import d2j.dutil;
import d2j.bitbtn;
import d2j.dstr;
import d2j.dstrlist;
import d2j.dcanvas;
import d2j.dvars;

class tform1 extends Frame implements Runnable
{
  Insets finsets;
  int finsets2_top = 0;
  int finsets2_left = 0;
  boolean FLayout = false;
  boolean ConsDone = false;
  boolean InPaint = false;
  dcanvas canvas = new dcanvas(this);
  dutil DUTIL = new dutil();
  public Label  label1;
  public Button  button1;
  public TextField  edit1;
  public TextArea  memo1;
  public Checkbox  checkbox1;
```

Figura 28 – Código Java gerado pela ferramenta Delphi2Java

6.4.2 TESTE COM TODOS COMPONENTES PASSÍVEIS DE CONVERSÃO

A figura 29 mostra uma interface desenvolvida em Delphi contendo todos os componentes possíveis de serem convertidos pela ferramenta Delphi2Java-II. A figura 30 caracteriza um erro na tentativa de conversão desta interface pela cópia *trial* da ferramenta Delphi2Java. E a figura 31 demonstra a conversão desta mesma interface pela ferramenta Delphi2Java-II. A fim de facilitar a comparação entre as figuras foi adicionada uma numeração ao lado dos componentes.

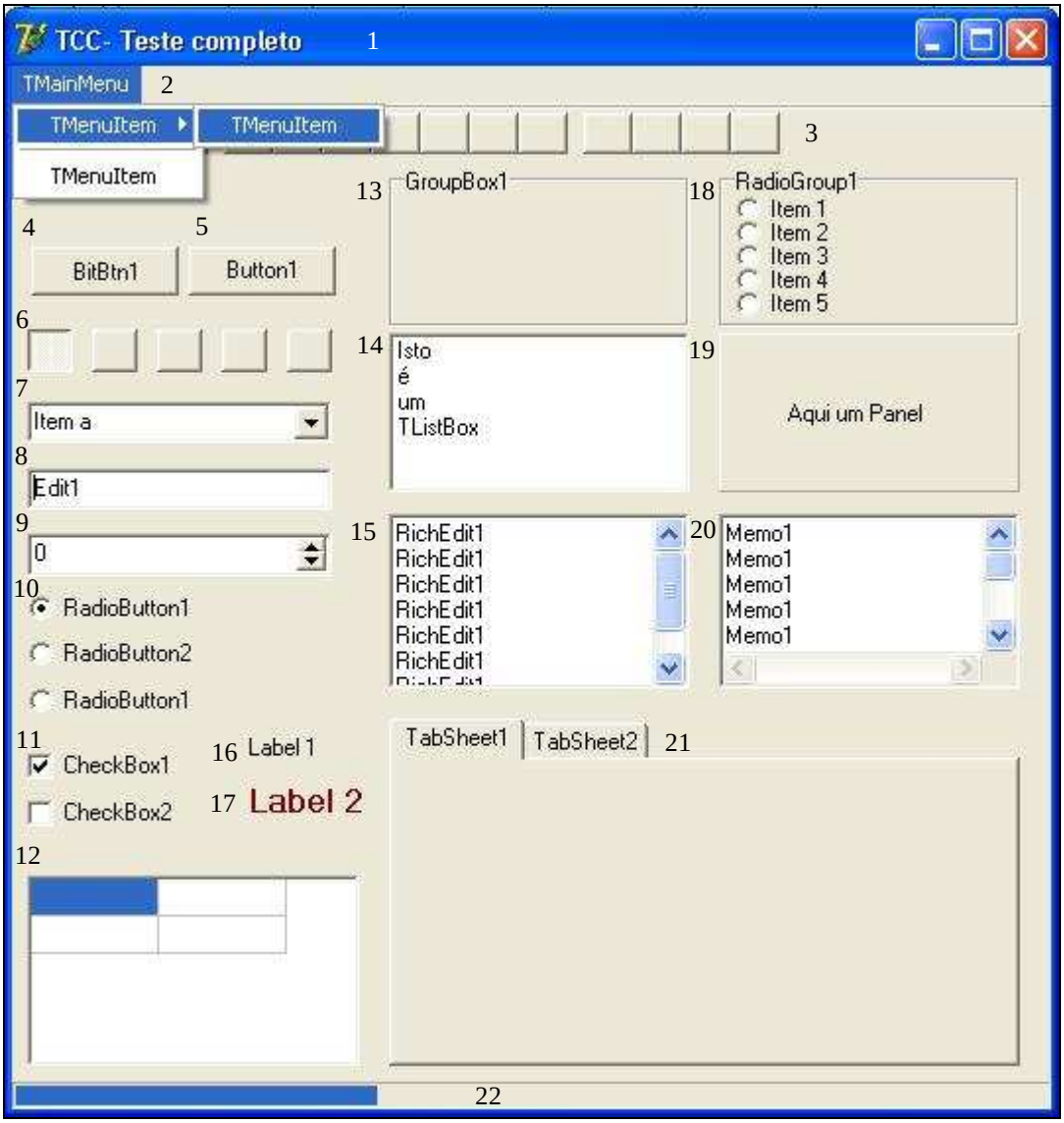

Figura 29 – Interface Delphi com os componentes contemplados pela ferramenta Delphi2Java-II

Figura 30 – Tentativa de conversão pela versão *trial* do Delphi2Java

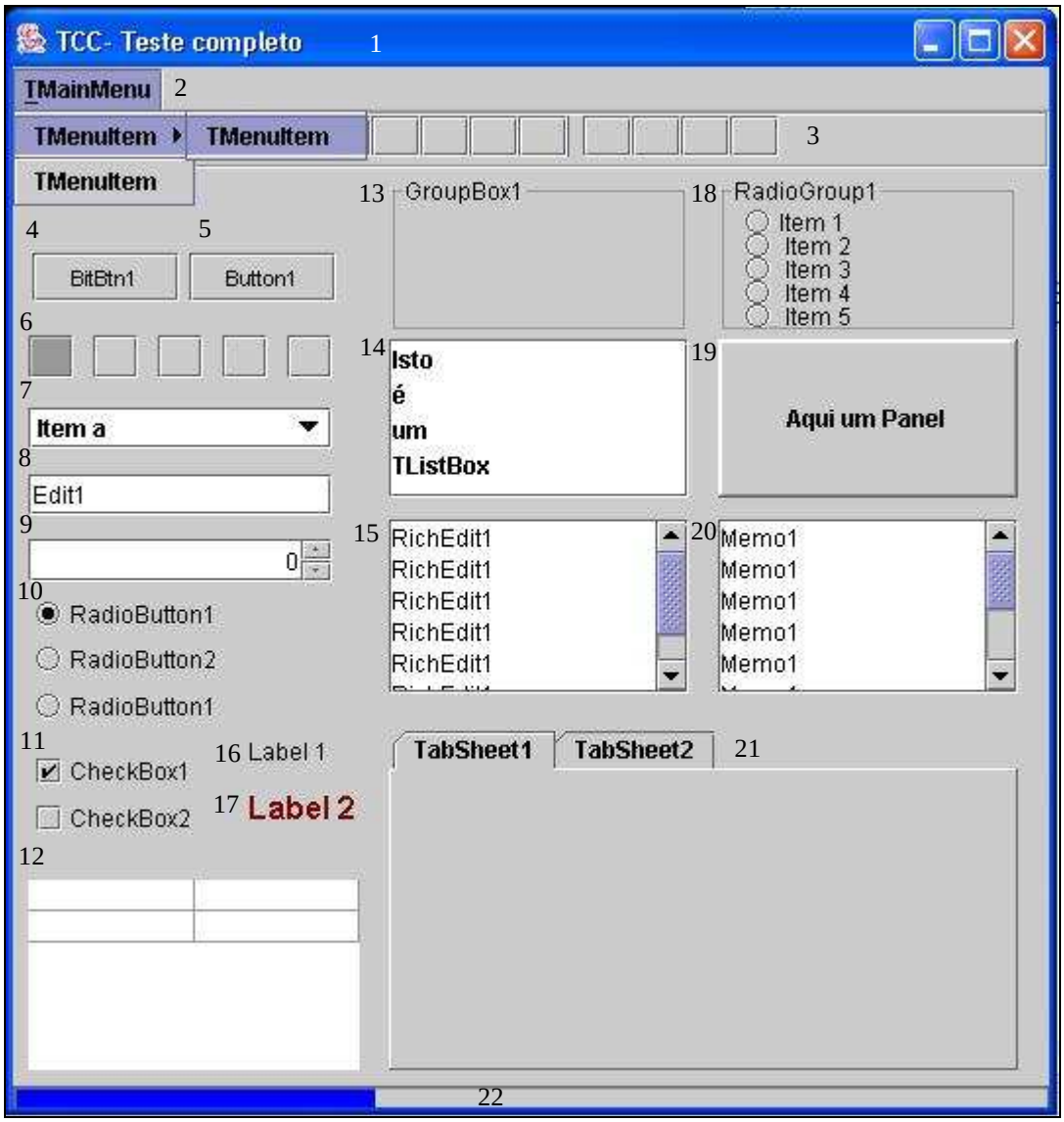

Figura 31 – Interface Java convertida pela ferramenta Delphi2Java-II

Como citado na seção 6.4.1.1, alguns componentes do Java apresentam fonte negrito como padrão, o que pode ser observado nos componentes 2, 7, 14, 19 e 21 da figura 31.

6.5 RESULTADOS E DISCUSSÃO

Alguns obstáculos iniciais, gerados principalmente pela inexperiência no ambiente Java por parte do acadêmico, não foram suficientes para a desistência deste projeto, muito pelo contrário, serviram de motivação para a busca do conhecimento necessário para a

continuidade e conclusão deste.

Os testes realizados demonstraram que a ferramenta desenvolvida atende os requisitos dispostos na seção 6.1. Sendo de simples manuseio, os formulários convertidos apresentaram-se da forma mais próxima a original além de manter os nomes dos objetos conforme o formulário desenvolvido em Delphi. Por gerar um arquivo separado para o tratamento dos eventos, tornou-se fácil a inserção de código pelo usuário final.

A ferramenta Delphi2Java, apresentada no capítulo 5, também demonstrou ser uma boa ferramenta, contudo foi descontinuada e tornou-se obsoleta para os parâmetros atuais.

A tabela 1 apresenta uma comparação entre as duas ferramentas.

Tabela 1 – Comparação entre as ferramentas Delphi2Java e Delphi2Java-II

DELPHI2JAVA X DELPHI2JAVA-II

	Conversão de interfaces gráficas	Conversão de código e tratadores de eventos	Utilização de pacote proprietário	Utilização AWT	Utilização *Swing*
Delphi2Java	Sim	Sim	Sim	Sim	Não
Delphi2Java-II	Sim	Não	Não	Sim	Sim

Com base nesta tabela, é possível visualizar os prós e contras de cada ferramenta. Enquanto a ferramenta Delphi2Java converte o código fonte e os métodos tratadores de eventos, esta se encontra desatualizada por utilizar apenas a biblioteca visual AWT, além de deixar o código fonte dependente de classe proprietária.

Por sua vez a ferramenta Delphi2Java-II não converte o código fonte, mas gera o código Java baseado no *framework Swing*, o qual é uma evolução da AWT possuindo maiores recursos visuais. Assim, uma extensão para esta ferramenta seria a implementação da conversão de código fonte dos tratadores de eventos e demais métodos presentes na aplicação.

7 CONCLUSÕES

A ferramenta desenvolvida atingiu os objetivos propostos, visto que os testes realizados apresentaram resultados satisfatórios quando comparados com os formulários originais desenvolvidos no ambiente Delphi.

O *framework Swing*, utilizado na construção do código Java, demonstrou ser uma boa opção para o desenvolvimento de interfaces gráficas em Java. Por se tratar de uma biblioteca que emula os componentes visuais, *Swing* obteve uma boa performance, não sendo observável a diferença no tempo de navegação entre os formulários no ambiente Java e no ambiente Delphi.

Uma limitação da ferramenta está no grupo de componentes selecionados para conversão, o qual não implementa todo o conjunto de componentes visuais existentes no ambiente Delphi. Contudo, por ser formado pelos componentes mais utilizados nas aplicações em geral, torna-se uma ferramenta eficaz em seu propósito, surgindo como uma opção para empresas que necessitem migrar seus softwares da plataforma Windows para a plataforma Java.

Por fim, além de agregar conhecimento em relação à programação Java, o desenvolvimento deste projeto serviu para conhecer melhor uma área do mercado pouco explorada no meio acadêmico, a migração de softwares, que surge como uma solução paliativa no momento de decidir se deve reescrever toda a aplicação em uma nova plataforma ou manter o que se tem e torná-la obsoleta.

7.1 EXTENSÕES

Tendo em vista um uso mais eficiente da ferramenta, sugerem-se as seguintes extensões:

a) converter o código fonte dos tratadores de eventos e demais métodos existentes na aplicação;
b) implementar a conversão de componentes do Delphi com acesso a banco de dados utilizando a ferramenta JDBC;
c) tornar a ferramenta mais flexível, permitindo ao usuário adicionar novos componentes para conversão.

REFERÊNCIAS BIBLIOGRÁFICAS

BINSTOCK, Andrew. **Porting Java code to .NET using J# .NET**. [S.l.], 2002. Disponível em: <http://www.devx.com/SummitDays/Article/6918>. Acesso em: 25 nov. 2004.

BORBA, Paulo. **Java com qualidade**. Recife, 2002. Disponível em: <http://www.cin.ufpe.br/~phmb/papers/JavaComQualidadeParaAJavaMagazine.htm>. Acesso em: 02 out. 2004.

BORLAND SOFTWARE CORPORATION. **Delphi™ 8 for the Microsoft® .NET framework**. [S.l.], [2003?]. Disponível em: <http://www.borland.com.br/delphi_net/>. Acesso em: 19 set. 2004.

CESAR, Ricardo. **Java x .Net: disputa acirrada no mercado nacional**. São Paulo, 2003. Disponível em: <http://computerworld.uol.com.br/AdPortalv5/adCmsDocumentShow.aspx?DocumentID=75187 >. Acesso em: 01 maio 2005.

EIWA SYSTEM MANAGEMENT. **Jude download**. [S.l.], 2005. Disponível em: <http://www.esm.jp/jude-web/en/download-e.html>. Acesso em: 13 maio 2005.

FILHO, Alcides Soares. **Migrar software reduz custos e economiza tempo**. [S.l.], 2003. Disponível em: <http://webinsider.uol.com.br/vernoticia.php/id/1844 >. Acesso em: 29 maio 2005.

MACÊDO, José Antônio F. **Aplicações na web**. Rio de Janeiro, 2003. Disponível em: <http://www.inf.puc-rio.br/~jmacedo/ppt/INF1345_2.ppt>. Acesso em: 19 set. 2004.

MICROSOFT CORPORATION. **Microsoft libera novo conversor Java para .NET**. São Paulo, 2002. Disponível em: <http://www.microsoft.com/brasil/pr/2002/jlca.htm>. Acesso em: 02 out. 2004.

PEIL, Norberto de Castro. **Borland Delphi 5**. Pelotas, 2000?. Disponível em: <http://www.cefetrs.tche.br/~npeil/>. Acesso em: 11 jul. 2005.

SIQUEIRA, Frank. **O modelo de componentes do Java**. Florianópolis, 2004. Disponível em: <http://www.inf.ufsc.br/~frank/INE5612/3.%20Componentes%20Java%20(6x1).pdf>. Acesso em: 24 set. 2004.

SPRITEMASTER DEVELOPMENT. **Delphi2Java supported procedures and functions**. [S.l.], 1997?. Disponível em: <http://www.spritemaster.com/delphi2java_supported_procedures_and_functions.html >. Acesso em: 08 jun. 2005.

WINSITE. **Winsite featured software**. [S.l.], 1997. Disponível em: <http://www.winsite.com/bin/Info?3253>. Acesso em: 01 abr. 2005.