UNIVERSIDADE FEDERAL DA BAHIA

INSTITUTO DE MATEMÁTICA

CIÊNCIA DA COMPUTAÇÃO

LINGUAGENS PARA APLICAÇÃO COMERCIAL

Trabalho Semestral

Descrição de Tecnologias Utilizadas

Salvador-BA

Abril/2009

CLEBER PINELLI

MARCUS VINÍCIUS LACERDA FAGUNDES

RAMON PEREIRA LOPES

Trabalho Semestral

Descrição de Tecnologias Utilizadas

Atividade apresentada como requisito para avaliação no curso de Ciência da Computação da Universidade Federal da Bahia na disciplina Linguagens Para Aplicação Comercial orientada pelo professor Adonai Estrela Medrado.

Salvador-BA

Abril/2009

Sumário

1. BEANS BINDING

1.1 Motivação dos desenvolvedores

O crescimento de aplicações que utilizam recuros de componentes gráficos e banco de dados trouxe consigo uma dificuldade relativa a associação dos elementos visuais aos objetos de domínio. Em meio a tal contexto, surge O Java BeansBinding, cujo objetivo é atacar o problema de sincronismo entre dois objetos, provendo recursos de validação e conversão.

1.2 Principais características, capacidades e restrições

- Recursos de validação e conversão de tipos
- Recursos para sincronismo entre JTable, List,entre outros denominado Swing Binding Extensions
- Isenta o desenvolvedor de se preocupar com centenas de listeners da linguagem Java, já que o processo de sincronismo e´automático
- Disponibiliza estratégias de sincronismo (read/write, read only, read once)
- Reuso do objeto de Binding, já que tem-se um mapeamento por string

1.3 Licença

O Java Beans Binding encontra-se sob licença GNU Lesser General Public License, escrita por Richard Stallman e Eben Moglen.[27]. O grande diferencial de tal licença para a GPL é que é permitido a associação com software proprietário, contudo devem estar em forma de biblioteca.

Permissões:

- Copiar, Distibuir, Executar o software

- Alterar o software de modo a criar um novo sobre o mesmo e fazer uso comercial

1.4 Plataforma

Como a blibioteca roda dentro da máquina virtual Java, tem-se que as plataformas e sistemas operacionais suportados são os mesmos da plataforma Java que já forma explicitados neste trabalho em outra seção.

1.5 Procedimentos para usar

O uso do BeansBinding é extremamente simples, pois não requer nenhum tipo de configuração adicional, como por exemplo configuração de arquivo XML.
O procedimento básico é adicionar a biblioteca BeansBinding ao conjunto de bibliotecas do projeto, portanto não há nada de diferente em relação a um programa básico Java.

2. SWING APPLICATIONS FRAMEWORK

2.1 Motivação dos desenvolvedores

Devido aos problemas recorrentes no processo de desenvolvimento de aplicações Java Desktop baseada no Swing, em 2006 iniciou-se o projeto Swing Application Framework JSR-296. Como uma proposta para resolução de tais problemas, o framework em questão disponibiliza uma espécie de 'kernel' para aplicações desktop, facilitando a o desenvolvimento de aplicações Java Destktop.

2.2 Principais características, capacidades e restrições

- Suporte a criação e gerenciamento de ações, em especial ações de segundo plano.
- Gerenciamento de recursos, tais como banco de dados, imagens, cores, internacionalização.
- Gerenciamento do ciclo de vida da aplicação.
- Persistência do estado da sessão, de modo que em uma próxima execução seja restaurado.

2.3 Licença

O Swing Application Framework está sob licença LGPL, assim como o Java Beans Binding, portanto ver seção 1.3 do presente documento.

2.4 Plataforma

Como a biblioteca roda dentro da máquina virtual Java, tem-se que as plataformas e sistemas operacionais suportados são os mesmos da plataforma Java que já forma explicitados neste trabalho em outra seção.

2.5 Procedimentos para usar

O uso do Swing Application Framework não requer nenhum tipo de configuração adicional, apenas deve-se colocar a biblioteca na pasta de bibliotecas do projeto.

3. VELOCITY

3.1 Objetivo e Motivação dos desenvolvedores

O Velocity é um *template engine* para a linguagem Java, isto é, a partir de um determinado modelo base podemos criar outros novos documentos a partir de parâmetros especificados. O framework em questão nasceu como um projeto individual, de modo que hoje é suportado pela fundação Apache [6], sendo migrado para outras plataformas como por exemplo o NVelocity, que atua sobre a plataforma .NET.

Por ser um *template engine*, o Velocity trabalha com processamento textual que, a partir de um *template*, processa-o de modo a invocar métodos de objetos definido no código Java, sendo que o seu resultado do processamento é textual. Um exemplo prático para tal funcionalidade seria a criação de classes Java a partir de um determinado *template*, para o qual seria passado um conjunto de parâmetros, como

por exemplo nome da classe, nomes dos atributos, entre outros; geração de arquivos XML.

Pode-se dizer que uma das principais motivações, se não a principal, para criação do Velocity foi o desenvolvimento de aplicações web, de modo que o código Java fica independente da camada de visualização, além de evitar que *designers* entrem em contato com código Java ou regra de negócio, propiciando separação entre a lógica e visualização, como também facilidade em termos de manutenibilidade [3].

3.2 Principais características, capacidades e restrições

- Sintaxe fácil e clara
- Adaptação a diversos problemas
- Possibilidade de acessar objetos Java dentro do *template*
- Definição de macros no *template [5]*
- Possui configurações de *caching* para deixá-lo mais rápido ainda [3]

3.3 Licença

O Apache Velocity encontra-se sob licença denominada Apache License 2.0, de modo que pode ser usado comercialmente sem custo algum, permitindo distribuição, modificação e redistribuição. A redistribuição pode ser feita sob uma outra licença, contudo deve ser informado que existe um componente sob a Apache License 2.0, como também a inclusão de dois arquivos na raiz do projeto [4]:

- LICENSE - Uma cópia da licença;
- NOTICE - Um documento que deverá listar todas as licenças utilizadas e os seus respectivos colaboradores.

3.4 Plataforma

Como a blibioteca roda dentro da máquina virtual Java, tem-se que as plataformas e sistemas operacionais suportados são os mesmos da plataforma Java que já forma explicitados neste trabalho em outra seção.

3.5 Procedimentos para usar

Como o Velocity é uma biblioteca Java, seu uso é como qualquer outra biblioteca da plataforma. O seu uso não requer nenhum arquivo de configuração ou mapeamento em XML, de modo que só se faz necessário a iniciação do componente no código Java, passagem dos parâmetros e invocação do template.

3.6 Exemplo de Uso

Segue abaixo um código de exemplo, demonstrando a utilização do Velocity:

```
// inicializando o velocity

   VelocityEngine ve = new VelocityEngine();
   ve.init();
   Date data = new Date(System.currentTimeMillis());
   DateFormat formatter = new SimpleDateFormat("dd/MM/yyyy");
   // criando o contexto que liga o java ao template
   VelocityContext context = new VelocityContext();
   String nomePrograma = new String();
    Template t = ve.getTemplate("Template1.vm");
    list1 = grupo1();
    list2 = grupo2();
    nomePrograma = "Relação de alunos aprovados";
   // aqui! damos a variavel para o contexto!
   context.put("programa", nomePrograma);
```

```
        context.put("data", formatter.format(data));
        context.put("lista1", list1);
        context.put("lista2", list2);
        StringWriter writer = new StringWriter();
        // mistura o contexto com o template
         t.merge(context, writer);
```

Neste exemplo utilizamos o velocity para popular uma página Html, utilizando como base o template abaixo:

```
<html>
<title>$programa </title>
<p align="right"> $data </p>
<head>
<p align="center">Listagem dos alunos </p>
</head>
<body>
<br><br>
<table>
<tr><th width="200" align="left">  Alunos aprovados  </th> <th width="200"
align="left">  Nota </th></tr>
#foreach ($item in $lista1)
<tr><td width="200" align="left">  $item.nome  </td> <td width="200"
align="left"> $item.nota </td>  </tr>
#end
</table>
<br><br>
<table>
<tr><th width="200" align="left">  Alunos reprovados  </th> <th width="200"
align="left">  Nota </th></tr>
#foreach ($item in $lista2)
```

```
<tr><td width="200" align="left">  $item.nome  </td> <td width="200"
align="left"> $item.nota </td>  </tr>

#end

</table>

</body>

</html>
```

Este programa portanto deve gerar uma pagina html com duas tabelas, contendo o nome e a nota dos alunos.

4. LINGUAGEM

4.1 Contexto de criação, pessoas e organizações envolvidas

Em 1991, na Sun Microsystems, foi iniciado o Green Project, os lideres deste projeto foram, Patrick Naughton, Mike Sheridan, e James Gosling, que deu origem, mais tarde, a linguagem Java, a idéia dos mentores do projeto não era apenas criar uma nova linguagem de programação, acreditavam em uma convergência dos computadores, equipamentos e eletrodomésticos, usados normalmente no dia-a-dia.

Em 1992, foi lançado então o *7 (StarSeven) , este protótipo era um controle remoto com uma interface gráfica *touchscreen*, e tinha a capacidade de controlar diversos dispositivos e aplicações. Para o *7 foi especificada uma nova linguagem de programação, batizada de Oak. Porém o *7, não encontrou o mercado que queria naquela época, que seria controlar televisões e vídeo por demanda, parecido com o que se tem hoje com a televisão digital.

No entanto, com o estouro da internet, estava surgindo uma grande rede interativa, e então o Oak foi adaptado, por responsabilidade de James Gosling, para a Internet e em janeiro de 1995, foi lançada a nova versão, que foi rebatizada de Java. E então, os Applets Java, trouxeram a dinamicidade aos estáticos HTML.

4.2 Histórico das Versões

A linguagem java foi governada pela Java Community Process(JCP) e sofreu várias mudanças, como adições de novas classes e bibliotecas.

As mudanças ocorridas na linguagem foram muito significativas, ao longo dos anos desde o JDK 1.0 o numero de classes evolui de centenas para milhares. Surgiram

novas APIs como Swing e Java2D e muitas classes e métodos oriundos do JDK 1.0 foram depreciados.

O java evoluiu passando pelas versões:

- Linguagem Oak (1990)
    - Desenvolvimento de software embutido para eletrodomésticos
- JDK 1.0 (1995)
    - Derivada da Oak, devido ao surgimento da Web surgiu o Java
- JDK 1.1 (1997)

    As principais incorporações à linguagem foram:

    - Nova forma de tratar os eventos (listeners);
    - Componentes (JavaBeans);
    - Acesso a banco de dados (JDBC);
    - Java Remote Method Invocation (RMI).
- JDK 1.2 (1999)

    Passou a ser conhecido como Java 2 ou J2SE (Java 2 Plataform, Standart Edition), substituindo o JDK, para poder distinguir o J2EE (Java 2 Platform, Enterprise Edition) e o J2ME (Java 2 Platform, Micro Edition).

    AS principais mudanças foram:

    - A correção de bugs, e otimização;
    - Integração do Swing com as classes essenciais;
    - Integração da JIT compiler à JVM (Java Virtual Machine);
    - Java-Plugin;
    - Collections Framework.
- J2SE 1.3 (2000)

    Suas mudamças foram:

    - HotSpot incluido na JVM;

- RMI foi modificada, tornando-se compatível com a CORBA;
- JavaSound;
- Java Naming and Directory Interface (JNDI);
- Java Platform Debugger Architecture (JPDA).

- J2SE 1.4 (2002)

Foi a primeira versão desenvolvida sob a JCP (Java Community Process), mudanças:

- Internet Protocol version 6 (Ipv6);
- Leitura e escrita de imagens de formatos como JPG e PNG;
- Integradas segurança e extensões criptográficas;
- Inclusão do Java Web Start.

- J2SE 5.0 (2004)

Originalmente numerado como 1.5, que ainda é usada como numero da versão interna. Novas características importantes são adicionadas nesta versão, como:

- Generics: Tipo seguro para collections eliminando a necessidade de muitas conversões;
- Metadata: Também chamado de annotations, permite construir classes e métodos com dados adicionais;
- Autoboxing: Conversões automáticas entre tipos seguros;
- Correção nos erros no Modelo de Memória que define como as threads interagem com a memória.

- Java SE 6 (2006)

Durante o desenvolvimento, foram construídos novos acessórios e corrigidos mais bugs. Algumas mudanças incluídas nesta versão foram:

- Significativa melhoria na performance;
- Suporte ao JDBC 4.0;

- Com diversas atualizações a este versão, passa a se chamar Java SE 6 Update 10, que incorpora modificações tais como:
    - Java Kernel, uma pequena instalação inclui apenas as classes comuns usadas no JRE, os demais pacotes podem ser instalados quando necessário;
    - Melhora na performance do Java 2D, usando Direct 3D;
    - A geração de plugins java: Os applets agora rodam em processos separados dando suporte a varias aplicações do Web Start applications.
- Java Se 7 (...)

É um plano de atualização do java, novas características deverão ser incluídas, como:

- JVM suportar linguagens dinâmicas;
- Uma nova biblioteca que suporta computação paralela em processadores multi-core;
- Substituição do garbage collection pelo Garbage First (G1), que deverá assegurar consistente pausa de tempo.

4.3 Concursos Públicos

Órgão: Ministério Público da União

Ano: 2007

Nível: Superior

Cargo: Analista

(Questão 1) Quanto às variáveis Java, um inteiro de 64 bits em notação de complemento de dois que pode assumir valores entre -2^{63} e $2^{63} -1$ é:

- long
- short
- float
- byte
- double

Resposta: A

(Questão 2) Objetos que têm uma representação no banco de dados, mas não fazem mais parte de uma sessão do *Hibernate*, o que significa que o seu estado pode não estar mais sincronizado com o banco de dados, são do tipo

- transient
- detached
- attached
- persistent
- consistent

Resposta: B

Órgão: Agência Nacional de Águas – ANA

Instituição: Escola de Administração Fazendária – ESAF

Ano: 2009

Nível: Superior

Cargo: Analista Administrativo - Desenvolvimento de Sistemas e Administração e Banco de Dados

(Questão 3) Se, na superclasse, um método é declarado *protected*, o(s) modificador(es) aplicável(is) na sobrescrita do método é(são)

- private
- private e protected
- “default”
- protected, “default” e public
- protected e public

Resposta: E

(Questão 4) Os argumentos são passados para métodos, em Java, por:

- valor, sejam valores primitivos ou referências a objetos
- referência, sejam valores primitivos ou referências a objetos

- valor, apenas valores primitivos
- referência, apenas para referências a objetos
- valor ou referência, de acordo com o sistema operacional

Resposta: A

(Questão 5) Em Java, para que um método de uma superclasse não seja sobrescrito em suas subclasses, aplica-se o modificador:

- static
- public
- final
- protected
- abstract

Resposta: C

(Questão 6) Em uma aplicação Java, se o carregador de classes não conseguir localizar a classe do driver do banco de dados para uma conexão JDBC, é lançada uma exceção:

- Java.lang.ClassNotFoundException
- Java.io.FileNotFoundException
- Java.io.SecurityException
- Java.io.IOException
- Java.util.InputMismatchException

Resposta: A

4.4 Requisitos mínimos e desejáveis para se execução de um programa

Para executar um programa Java genérico faz-se necessário a instalação do *Java Runtine Environment* – JRE, sendo responsável por executar as aplicações Java. O JRE é composto pela máquina virtual Java e por um conjunto de API padrão.

O JRE está disponível para download para os seguintes sistemas operacionais Windows, Solaris, Linux, sendo que o requisito mínimo de memória varia de 64mb a 128mb[10].

4.5 Independente de Máquina

Java é tida como independente de máquina, de modo que sua linguagem é tanto compilada quanto interpretada. Inicialmente o compilador transforma o código fonte em linguagem Java em um programa fonte denominado *bytecodes*, que nada mais são que instruções para a máquina virtual. A existência do *bytecode* dá o caráter de independência ao Java, de modo que torna-se dependente exclusivamente da máquina virtual Java, sendo que esta sim é dependente da plataforma utilizada. O papel da máquina virtual Java é transformar as instruções em *bytecode* em código de máquina.

Pode-se concluir que a linguagem Java é tanto compilada quanto interpretada, independente de máquina, tendo um slogan criado pela empresa Sun Microsystems para expressar tal portabilidade: "*Write Once, run anywhere"*.

Sintaxe e modo de uso dos principais elementos e estruturas

- Tipos básicos existentes e forma de declaração.

Tipo	**Descrição**
boolean	Assume valores true ou false.
byte	Inteiro de 8 bits, pode assumir valores entre -27 e 27-1
char	Caractere em notação Unicode de 16bit, armazena valores alfanuméricos. Também pode ser como inteiro com valores entre 0 e 2^{16} – 1.
short	Inteiro de 16 bits, os valores estão compreendidos entre -215 e 215-1
int	Inteiro de 32 bits, pode assumir os valores entre -231 e 231-1
long	Inteiro de 64 bits, pode assumir os valores entre -263 e 263-1
float	Representa números em notação ponto flutuante normalizada em precisão simples de 32 bits em conformidade com a norma IEEE 754-1985. Possui 32 bits de tamanho e 23 dígitos binários de precisão.
double	Representa números em notação ponto flutuante normalizada em precisão dupla de 64 bits em conformidade com a norma IEEE 754-1985. Possui 64 bits de tamanho e 52 dígitos binários de precisão.

A declaração desses tipos primitivos é bem similar a sintaxe do C/C++, e é da forma básica:

```
TIPO IDENTIFICADOR [= VALOR_INICIAL];
```

Ou uma lista de variáveis:

```
TIPO IDENT_1 [= VALOR_INICIAL1], IDENT_2 [= VALOR_INICIAL_2], IDENT_3 [= VALOR_INICIAL_3];
```

Exemplos:

```
int numero;

char primLetra = ‘a’, ultLetra = ‘z’, algumaLetra;

double pi = 3.1415;
```

- Formas de declaração e uso de estruturas, vetores e matrizes.

As estruturas de vetores e matrizes podem ser declaradas de diversas formas em Java. E envolvem três etapas:

Declarar o vetor ou matriz. Basta acrescentar um par de colchetes antes ou após o nome a variável, ou ainda logo após o tipo da variável:

```
int ind[];

double table[][], cube[][][];

int []nota;

boolean[] valores;
```

Reservar o espaço de memória e definir tamanho. E preciso definir o tamanho do vetor e em seguida reservar o espaço de memória para armazenar os elementos do vetor, isto é realizado pelo operador *new*.

```
ind = new int[10];

table = new double[10][5];

valores = new boolean[5];
```

Armazenar valores no vetor. Para armazenar valores em um dos elementos de um vetor ou matriz, é necessário fornecer um índice que indique a posição do elemento. Os índices iniciam de zero.

```
ind[0] = 10;

table[9][4] = 3.14;
```

```
valores[2] = true;
```

Existe ainda uma forma de realizar as três etapas de uma só vez, quando se deseja criar um vetor com valores atribuídos de modo estático. Como se pode observar abaixo:

```
long[] fibonacci = {1,1,2,3,5,8,13,34,55,89,144};

char[] vogais = {'a','e','i','o','u'};
```

- Instruções condicionais.

As estruturas condicionais também são similares as linguagens C/C++. Porém as expressões condicionais devem explicitamente ser do tipo boolean. Como se pode observar abaixo:

if (*expressão_booleana*) *instrução_simples;* if (*expressão_booleana*) { *instruções* }	if (*expressão_booleana*) { *instruções* } else if (*expressão_booleana*) { *instruções* } else { *instruções* }

Como pode-se notar quando existe apenas uma instrução não é preciso, abrir um bloco com as chaves. E se existir mais de uma condição usa-se o *else* como se pode notar na segunda coluna.

```
if ( (numero % 2) == 0 ) {
   par = true;
} else {
   par = false;
}
```

- Instruções de repetição.

As estruturas também seguem a mesma sintaxe do C/C++, exceto pela condição de parada/continuação. As estruturas são:

Estruturas while e do-while:

while (*expresão_booleana*) { *instruções;* }	do { *instruções;* } while (*expresão_booleana*);

No caso do while as instruções são executadas enquanto a expressão booleana for verdadeira.

E no caso do do-while as intruções são executadas ao menos uma vez, e continua a ser executadas enquanto a expressão booleana for verdadeira.Abaixo exemplos das duas estruturas:

```
int x = 0;

while (x < 10)

{

   System.out.println ("item " + x); // imprime a string “item 0” “item 1” até “item 9”.

   x++; // incrementa x em 1.

}
```

```
int x = 0;

do{

   System.out.println ("item " + x); // imprime a string “item 0” “item 1” até “item 10”.

   x++; // incrementa x em 1.

} while (x < 10);
```

Estrutura for:

for (*inicialização*;	for (*variável*: *array*)

expressões booleanas; *passo da repetição*) { *instruções*; }	{ instruções; }

Na segunda coluna tem-se uma alternativa para ‘varrer’ um array, no qual a cada iteração do loop, um dos elementos de *array* é atribuído a *variável*, que pode ser usada no escopo da estrurura. Abaixo exemplos:

```
for ( int x = 0; x < 10; x++ ) {

   System.out.println ("item " + x);

}

int[] ind = {1,2,8,16,32,64};

for(int x : ind){

    System.out.println ("item " + x); //imprime todos elementos de ‘ind’

}
```

- Definição de função/objetos.

A definição de funções, tomando a liberdade para usar tal termo já que o correto seria método, deve ser feita dentro de classes, podendo ser do tipo estático ou não estático. Um método ao ser declarado deve seguir:

```
Modificador* TipoRetorno NomeMétodo Throws*  (Parâmetros){}

Modificador – public, protected private static abstract final native
synchronized

TipoRetorno – void, tipos primitivos, tipos abstratos de dados

Throws – Explicita possíveis exceções lançadas
```

A respeito de objetos, temos que esses podem ser declarados dentro de métodos, blocos de código, assim como podem ser vistos como atributos e parâmetros dos métodos. Um objeto precisa ser uma instância de uma classe, segundo a sintaxe Java:

```
Tipo identificador = new Tipo();
```

`Tipo – Classes, Interfaces`

`new – palavra reservada`

- Definição de comentários.

Existem duas forma de fazer comentários nos código Java, são eles: “//” e o “/* */”.

O primeiro, consiste em tudo que vier após // serão desconsiderados para a compilação do código, porém é o comentário de apenas uma linha, ou seja, será desconsiderado o que estiver entre // e a próxima quebra de linha.

O segundo, consiste em tudo que vier /* e */ serão desconsiderados no momento da compilação, e neste caso o comentário pode usar quantas linha forem necessárias.

- Mecanismo de controle de erros (exceção).

O mecanismo de tratamento de erros é fundamental nas linguagens atuais. Uma exceção é lançada quando algum tipo de falha ocorre, como por exemplo tentar acessar uma posição de um array com um índice maior que o tamanho do array, ou tentar executar um método de um objeto ainda não instanciado. Quando é lançada uma exceção é possível, trata-la e neste caso, a execução continua normalmente, quando não é tratada, o programa é encerrado de forma inesperada. Eh possível então em Java, tratar exceções, como também lançar exceções quando estas forem detectadas. Eh interessante ressaltar ainda que nem sempre as exceções são lançadas por erro na lógica do programa, mas também quando o sistema viola de alguma forma a regra de negocio do sistema em questão. Abaixo alguns exemplos de tratamento e lançamento de exceções:

```
public static void main(String args[]){

    int[] array = new int[3];

    System.out.println("Tudo ok?");

    array[4] = 10;

    System.out.println("Sim");

}
```

Neste exemplo acima será lançado a exceção, `java.lang.ArrayIndexOutOfBoundsException`.

Que não está sendo tratada, a execução será terminada, sem a impressão do texto na ultima linha. A fim de construirmos programas mais robustos, poderíamos tratar situações como esta. Para isto iremos utilizar o bloco try-catch:

```
try{
      //Código perigoso
}catch(Exception e)
{
      //tratamento da exceção
}
```

O bloco try-catch define dois trechos de código, um sujeito a erros (try) e outro que responsável pelo tratamento do erro caso aconteça (catch).

```
public static void main(String args[]){
        int[] array = new int[3];
        try{
            System.out.println("Tudo ok?");
            array[4] = 10;
        }
        catch(java.lang.ArrayIndexOutOfBoundsException e){
            System.out.println("Aconteceu um problema");
        }
        System.out.println("Mas tudo ok");
    }
```

- Mecanismo de acesso ao banco de dados e arquivos.

Uma das formas de acessar um banco de dados e utilizando Java DataBase Connectivity – JDBC, cuja teoria é explanada na seção 4.8. Abaixo um exemplo:

```
public class ConnectionFactory {
    public static Connection getConnection(){
        try{
            Class.forName("com.mysql.jdbc.Driver");
```

```
            return
DriverManager.getConnection("jdbc:mysql://localhost:3306/bd_usuario","root"
,"lacerda");

        }

        catch (ClassNotFoundException ex) {


Logger.getLogger(ConnectionFactory.class.getName()).log(Level.SEVERE, null,
ex);

        }catch(SQLException sqle){


Logger.getLogger(ConnectionFactory.class.getName()).log(Level.SEVERE, null,
sqle);

        }

        return null;

    }

    public static void main(String[] args) throws SQLException {

         Connection con = ConnectionFactory.getConnection();

         PreparedStatement st = con.prepareStatement("Select pr.nome from
bd_usuario.produto pr Where pr.preco > 1 ");

         ResultSet res = st.executeQuery();



         while(res.next()){

             System.out.println(res.getString("nome"));

         }



         con.close();

    }

}
```

Existem várias maneiras de se ler e escrever arquivos em Java. É possível, por exemplo escrever e ler, um dos tipos primitivos, uma linha inteira do arquivo ou até mesmo um objeto 'serializado'. Abaixo alguns simples exemplos de leitura e escrita em arquivo.

```
try {
   FileWriter fw = new FileWriter(new File("teste.txt"));
   fw.write("escrevendo uma string");
   fw.close();
   FileReader fr = new FileReader(new File("teste.txt"));
   BufferedReader br = new BufferedReader(fr);
   String linha = br.readLine();
   System.out.println(linha);
} catch (IOException ex) {
   ex.printStackTrace();
}
```

- Mecanismo de acesso aos dispositivos externos

O Java possui uma API denominada Java Communications 3.0, cujo foco é prover a comunicação de aplicações Java com recursos como SmartCards, sistemas embarcados, Point-of-Sale POS, fax, modems, entre outros.

Em termos de comunicação não nativa, isto é, utilizando bibliotecas de terceiros, encontra-se a existência do jUSB, que é uma API para comunicação com dispositivos USB.

O tipo de comunicação *bluetooth*, que hoje é bastante utilizada no mundo dos dispositivos móveis, surge como uma forma de integração de aplicações Java com dispositivos externos. Existe um Java Specification Request de número 82, cujo foco é especificar uma API para comunicação via *blutetooh* na plataforma Java. Vale ressaltar que tal API é difícil uso, sendo comumente utilizando frameworks para facilitar o desenvolvimento, dentre os quais destaca-se o projeto brasileiro Marge.

4.6 Como realizar conexão com um banco de dados

A conexão com banco de dados é realizada de forma bastante elegante, pois o Java fornece o JDBC API do pacote *java.sql* que fornece uma série de classes e interfaces para conectividade, isto é, conexão com o banco de dados, execução de *queries*. Vale ressaltar que o uso de tal API é independente do Sistema Gerenciador de Banco de Dados utilizado, logo faz-se necessário um componente denominado geralmente denominado por driver JDBC, geralmente fornecido pelo fabricante do

banco de dados, que é responsável por fazer o intermédio em entre o sistema gerenciador e o Java.

O uso de tal API atualmente pode ser enxergado como um gargalo no desenvolvimento de aplicações que utilizam persistência em banco de dados. Como tentativa de solucionar tal problema, em meio a especificação do EJB 3.0 surgiu o Java Persistence API, que nada mais é do que uma especificação, um novo padrão para persistência na plataforma Java. De um modo grosseiro, o JPA pode ser entendido como uma evolução sobre o JDBC API, pois fornece recursos de mapeamento objeto relacional, disponibilizando uma série de funcionalidades.

4.7 Interação com bibliotecas desenvolvidas em outras linguagens

A Java Native Interface – JNI define um padrão que permite uma aplicação Java, isto é, rodando sobre a máquina virtual Java, invocar aplicações nativas escritas em linguagens como C, C++. Apesar do recurso de invocação de código nativo por uma aplicação Java, o JNI impossibilita o famoso jargão da plataforma Java: “Escreva uma vez, rode em qualquer lugar”, pois uma vez que necessitamos de código nativo, a aplicação torna-se dependente de tal plataforma.

Contornando o problema o problema de dependência de plataforma do JNI, a JSR 223 define um padrão para uso de linguagens de scripts, tais como Python, Ruby, Tcl, JavaScript, dentro da plataforma Java, isto é, de uma aplicação Java podemos invocar funções, instanciar objetos de uma biblioteca escrita em uma linguagem de script que seja suportada pela plataforma Java, tirando proveito do melhor dos dois mundos.

A fim de utilizar tal recurso, apenas faz-se necessário a obtenção do *engine* apropriado para a linguagem de script. A versão do Java 6 fornece como *engine* padrão o Mozilla Rhino, referente ao JavaScript.

Segue abaixo um pequeno trecho de código que demonstra a utilização de tal recurso com a linguagem Ruby:

```
//Trecho do código Java invocando a biblioteca em Ruby

ScriptEngineManager manager = new ScriptEngineManager();

ScriptEngine engine = manager.getEngineByName("jruby");

engine.eval(new BufferedReader(new FileReader("script.rb")));

Invocable invoke = (Invocable) engine;

invoke.invokeFunction("soma", 1, 2);
```

```
//Biblioteca em Ruby

def soma(x,y)

  puts x+y

end
```

4.8 Utilitário de geração de documentação

O Javadoc é uma ferramenta desenvolvida pela Sun Microsystems para geração de documentação no formato HTML a partir do código fonte, permitindo documentação de pacotes, classes, atributos e métodos.

A geração automática é possível desde que o desenvolvedor utilize as marcações definidas no padrão Javadoc, sendo que os comentários devem ser estar englobados por /** */ e ser iniciados pelo caractere @.

Segue abaixo uma lista das principais *tags* do padrão:

`@author` – Informação sobre o autor da documentação e/ou código.

`@deprecated` – Marca o método como *deprecated*.

`@exception` – Explicita uma exceção lançada por um método.

`@param` – Define um parâmetro do método.

`@return` – Documenta o valor de retorno de um método.

4.9 Tipo de aplicações que pode desenvolver

A linguagem pode ser utilizada não somente para escrever aplicações desktop, como também web e mobile. A plataforma Java pode ser dividida segundo os seguintes padrões:

Java Standard Edition – Plataforma cujo principal foco é o desenvolvimento de aplicações desktop, fornecendo API para conectividade com banco de dados, redes, Remote Method Invocation, operações de I/O.

Java Enterprise Edition – Plataforma cujo foco é o desenvolvimento de aplicações corporativas, baseada em componentes que rodam dentro de um servidor de aplicações. Além de ser uma plataforma, pode ser vista como uma especificação, padrão de desenvolvimento para tais aplicações. A plataforma JEE contém uma série de componentes como Enterprise Java Beans, Java Server Pages, Servlets.

Java Micro Edition – Plataforma cujo foco é o desenvolvimento de aplicações para dispositivos móveis e embarcados como celular, PDA, refrigeradores, entre outros. Um dos aspectos mais relevantes desta plataforma é a limitação computacional dos dispositivos para os quais as aplicações são desenvolvidas, resultando em uma série de limitações e modificações feitas na máquina virtual Java. A plataforma J2ME define dois principais tipos de configurações Connected Device Configuration – CDC e Connected Limited Device Configuration – CLDC, como também perfis, sendo que o mais conhecido é o Móbile Information Device Profile – MIDP.

4.10 Ambientes de desenvolvimentos integrados (IDE)

Os dois principais IDEs disponíveis do mercado são:

1- Netbeans

Fabricante: Projeto Open-Source, contudo patrocinado pela Sun Microsystems

Site: http://www.netbeans.org/

Licença: Como o Netbeans é um IDE que reúne uma série de funcionalidades, encontra-se os seguintes tipos de licença: GNU Lesser General Public License version 2.1, GNU General Public License version 2.

Valor: Gratuito

Características: Suporte ao JSE, JME, JEE, C/C++, Ruby, PHP, JavaScript, Ruby, Python, Groovy, SOA, pode ser utilizado como plataforma para construção de aplicações, além de recursos como *debuger*, *profiler*, geração de código automático.

2- Eclipse

Fabricante: O Projeto Eclipse foi criado pela IBM em 2001

Site: http://www.eclipse.org/

Licença: Eclipse Public License (EPL)

Valor: Gratuito

Características: A forte orientação ao desenvolvimento baseado em plug-ins e o amplo suporte ao desenvolvedor com centenas de plug-ins que procuram atender as diferentes necessidades de diferentes programadores.

Plugins Básicos: XML, GUI (Swing/SWT), J2EE (Web/EJB), Testes, UML, Relatórios e Gráficos.

4.11 Padronização

Em 1997 a Sun Microsystems tentou submeter a linguagem à padronização pelos orgãos ISO/IEC, mas acabou desistindo. Java estava em bem encaminhada para padronizar de fato, a Sun tornou-se reconhecida pela Publicly Available Specifications (PAS), mas a JCTI modificou o procedimento da PAS impedindo a Sun de utilizar seu status. Em 1999 tentou submeter pelo orgão ECMA, mas desistiu novamente, após conhecer o comitê e as regras que não foram bem elaboradas, entrando em desacordo com a Sun.

Pelo falta incentivo, o Java é ainda um standard, que é controlada pela Java Community Process – JCP.

5. REFERÊNCIAS

[1] “Swing Application Framework”. Disponível em: <https://appframework.dev.java.net/>. Acessado em: 25 de Abril de 2009.

[2] “Using the Swing Application Framework (JSR 296)” . Disponível em: <http://java.sun.com/developer/technicalArticles/javase/swingappfr/>. Acessado em: 25 de Abril de 2009.

[3] “Introdução ao Velocity” . Disponível em: <http://www.guj.com.br/article.show.logic?id=18>. Acessado em: 25 de Abril de 2009.

[4] Licença Apache . Disponível em: <http://en.wikipedia.org/wiki/Apache_License>. Acessado em: 25 de Abril de 2009.

[5] Apache Velocity . Disponível em: <http://www.roseindia.net/apachevelocity/>. Acessado em: 25 de Abril de 2009.

[6] Velocity . Disponível em: <http://velocity.apache.org/engine/devel/index.html>. Acessado em: 25 de Abril de 2009.

[7] Java Linguagem de Progrmacação - Wikipedia . Disponível em: <http://pt.wikipedia.org/wiki/Java_(linguagem_de_programação)>. Acessado em: 25 de Abril de 2009.

[8] "Fundamentos Java" . Disponível em: <http://www.inf.pucrs.br/~flash/lapro2/lapro2_2.pdf>. Acessado em: 25 de Abril de 2009.

[9] Java Version History - Wikipedia . Disponível em: <http://en.wikipedia.org/wiki/Java_version_history>. Acessado em: 25 de Abril de 2009.

[10] "What are the system requeriment for Java 6?". Disponível em: <http://www.java.com/en/download/help/6000011000.xml>. Acessado em: 25 de Abril de 2009.

[11] Java Programming Language - Wikipedia. Disponível em: <http://en.wikipedia.org/wiki/Java_(programming_language)>. Acessado em: 25 de Abril de 2009.

[12] "*Java Language Specification*" . Disponível em: <http://java.sun.com/docs/books/jls/first_edition/html/19.doc.html>. Acessado em: 25 de Abril de 2009.

[13] "Java Communications 3.0 API" . Disponível em: <http://java.sun.com/products/javacomm/ >. Acessado em: 25 de Abril de 2009.

[14] "Java USB" . Disponível em: <http://jusb.sourceforge.net/>. Acessado em: 25 de Abril de 2009.

[15] "JSR 82: Java APIs for Bluetooth" . Disponível em: <http://jcp.org/en/jsr/detail?id=82>. Acessado em: 25 de Abril de 2009.

[16] Marge. Disponível em: <https://marge.dev.java.net/>. Acessado em: 25 de Abril de 2009.

[17] JDBC - Tutorial. Disponível em: <http://java.sun.com/docs/books/tutorial/jdbc/index.html >. Acessado em: 25 de Abril de 2009.

[18] Marge . Disponível em: <http://www.devmedia.com.br/articles/viewcomp.asp?comp=4590 >. Acessado em: 25 de Abril de 2009.

[19] Java Persistence API. Disponível em: <http://en.wikipedia.org/wiki/Java_Persistence_API>. Acessado em: 25 de Abril de 2009.

[20] “Java invocando Ruby - JSR 223” . Disponível em: <http://blogs.sun.com/ramonlopes/>. Acessado em: 25 de Abril de 2009.

[21] “JSR 223: Scripting for the Java Platform” . Disponível em: <http://jcp.org/en/jsr/detail?id=223>. Acessado em: 25 de Abril de 2009.

[22] “Scripting” . Disponível em: <https://scripting.dev.java.net/>. Acessado em: 25 de Abril de 2009.

[23] Java Native Interface - Wikipedia. Disponível em: <http://en.wikipedia.org/wiki/Java_Native_Interface>. Acessado em: 25 de Abril de 2009.

[24] Java Native Interface – Tutorial. Disponível em: <http://eupodiatamatando.com/estudos/tutorial-jni/>. Acessado em: 25 de Abril de 2009.

[25] “How to Write Doc Comments for the Javadoc Tool” . Disponível em: <http://java.sun.com/j2se/javadoc/writingdoccomments/index.html>. Acessado em: 25 de Abril de 2009.

[26] Javadoc. Disponível em: <http://pt.wikipedia.org/wiki/Javadoc>. Acessado em: 25 de Abril de 2009.

[27] Java Micro Edition - Wikipedia. Disponível em: <http://pt.wikipedia.org/wiki/Java_ME>. Acessado em: 25 de Abril de 2009.

[28] Java Enterprise Edition - Wikipedia . Disponível em: <http://pt.wikipedia.org/wiki/Java_EE>. Acessado em: 25 de Abril de 2009.

[26] “Why Java was not standardized twice” . Disponível em: <http://csdl2.computer.org/comp/proceedings/hicss/2001/0981/05/09815015.pdf>. Acessado em: 25 de Abril de 2009.

[27] LGPL, Disponível em: <http://pt.wikipedia.org/wiki/LGPL>. Acessado em: 26 de Abril de 2009.