UNIVERSIDADE FEDERAL DA BAHIA

INSTITUTO DE MATEMÁTICA

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

LINGUAGEM DE PROGRAMAÇÃO JAVA

Equipe:

André Monteiro

Jerônimo Teles

José Pina

Marcelo Luz

Persis Castro

1. Em que contexto a linguagem surgiu?

Foi criado um projeto de pesquisa, financiado pela Sun Microsystems, em 1991, com o nome de Green Project como promessa de geração de software embarcado.

O Projeto não estava voltado para computadores, mas para uma grande gama de equipamentos eletrônicos, como televisores e vídeos. Na tentativa de construir programas interativos para esses produtos foi criada uma linguagem baseada em C++ que seu criador, James Gosling, chamou de Oak e, posteriormente, foi batizada de Java (cidade de origem de um tipo de café importado).

Apesar da eficiência de códigos compactos, o projeto passou por algumas dificuldades e não obteve sucesso comercial neste momento inicial. O mercado de dispositivos eletrônicos inteligentes voltados para o consumo popular não estava se desenvolvendo, no início da década de 1990, tão rápido como a Sun havia antecipado.

Com a Internet se popularizando em 1993, a equipe da Sun viu o imediato potencial de utilizar o Java para adicionar conteúdo dinâmico às páginas da Web, onde seria possível a interatividade entre o usuário e a aplicação em tempo real, dinâmico e visual. A partir daí, foram criados applets com o Java e, após isso, a linguagem passou a ser mundialmente conhecida e utilizada.

A Sun anunciou formalmente o Java em uma importante conferência em maio de 1995.

2. Quais os principais nomes (pessoas ou empresas) envolvidos na criação da linguagem?

O Green Project foi elaborado em 1991 por uma equipe formada por James Gosling, Mike Sheridan e Patrik Naughton, da empresa Sun Microsystems.

Com o surgimento da Internet, a Sun criou a linguagem Java para elaboração de aplicativos Web. Em 1995, depois de lançada a versão JDK 1.0 alpha, a Netscape implementa interpretadores Java em seu navegador, criando Java applets.

3. Quantas versões anteriores possui a linguagem? Quais as principais evoluções que ocorreram? (Deve-se desenvolver um quadro comparativo entre as versões)

•[1995] Versão1.0, denominada Java Development Kit (JDK)

–212 classes em 8 pacotes

–Lento, muitos bugs, suporte a Applets

•[1997] Versão1.1, denominada JDK 1.1

–504 classes em 23 pacotes

–Melhoria na eficiência da JVM

–Principais extensões: classes aninhadas, JavaBeans, JDBC, RMI, ...

•[1998] Versão1.2, a partir daqui denominada Java 2 Platform(J2SE)

–1520 classes em 59 pacotes

–JVM da Sun com compilador JIT (just-in-time)

–Principais extensões: Swing, coleções, ...

–Codinome Playground

•[2000] Versão1.3, denominada J2SE 1.3

–1842 classes em 76 pacotes

–Melhoria na eficiência da JVM

–Codinome Kestrel

•[2002] Versão1.4, denominada J2SE 1.4

–2291 classes em 135 pacotes

–Melhoria na eficiência da JVM

–Principais extensões: asserções, transferência de exceções, segurança e criptografia, ...

–Disponibilizado em 3 plataformas:

Java 2 Micro Edition (J2ME), para dispositivos móveis

Java 2 Standard Edition (J2SE), para desktop

Java 2 Enterprise Edition (J2EE), para aplicações corporativas

-Desenvolvimento facilitado por ambientes poderosos: IDE (Integrated Development Environment)

NetBeans, da Sun

Eclipse, da IBM

–Codinome Merlin

•[2004] Versão5.0, denominada J2SE 5.0

–3000 classes em 165 pacotes

–Principais extensões: genéricos, enumerações, tipos primitivos e classes de embrulho,

número variável de argumentos

–Versão anteriormente numerada 1.5

–Codinome Tiger

•[2006] Versão6.0, denominada J2SE 6.0

–Principais extensões: XML, web services, …

–Suporte para o Red Hat Enterprise Linux 5 foi introduzido no Java SE 6, Update 4

- Java SE 6 não da suporte para Windows 98 ou Windows ME.

- Java SE 6 não da suporte para o processador Intel Itanium.

–Codinome Mustang

•A versão J2SE 7 possui codinome Dolphin, e está estimada para 2008. Semanalmente os mantenedores do projeto liberam uma versão.

O endereço do projeto é https://jdk7.dev.java.net/. A partir do mesmo, os interessados podem se cadastrar e começar a contribuir com o projeto

4. Quais aplicações comerciais já foram desenvolvidas com esta linguagem?(Incluir site para as mesmas)

Atualmente, a linguagem Java é uma das mais usadas e serve para qualquer tipo de aplicação, entre elas: web, desktop, servidores, mainframes, jogos, aplicações móveis e chips de identificação.

O eBay, atualmente o maior site do mundo para a venda e compra de bens pela internet, foi desenvolvido em Java.

Site: www.ebay.com

O JBidWatcher é uma outra aplicação baseada em Java, que tem como funcionalidade permitir o monitoramento de leilões no eBay, submeter lances, dar lancesperto do último momento do leilão.

Site:http://www.jbidwatcher.com/

O Elixir Report.Uma ferramenta para elaboração de relatórios que suporta diferentes fonte de dados como: Java/EJB, XML, JDBC, LDAP, arquivos texto.

Site: http://www.elixirtech.com/

SwisSQL. Aplicação que ajuda a converter procedures Oracle PL/SQL para Java.

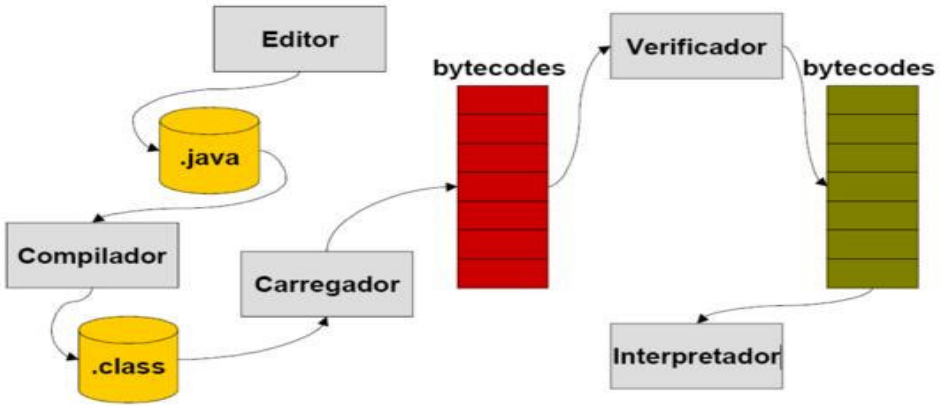
Site: http://www.swissql.com/products/oracle-to-java/oracle-to-java.html

Extracta. Oferece uma busca automática na internet de dados que podem ser integrados em outras aplicações.

Site: http://www.spark-it.com/

5. A linguagem foi desenvolvida para ser interpretada ou compilada?"Independente de máquina"? Quais os requisitos mínimos e desejáveis para se executar um programa nesta linguagem?(Exemplo: precisa-se das bibliotecas runtime X, servidor de página Y na versão X ou superior)

A linguagem é tanto compilada como interpretada. O compilador transforma o código fonte em bytecodes que são instruções compreendidas pela Máquina Virtual Java (JVM), um interpretador que transforma as instruções em linguagem de máquina. Para que a arquitetura seja compatível com a linguagem Java, basta apenas existir uma JVM para a arquitetura. Com isto Java proporciona um ambiente virtual independente de máquina, pois permite a execução de código em diversas arquiteturas.



http://my.opera.com/edsonlopes/homes/blog/Java_Funcionamento.gif/

Para executar códigos Java diretamente em um navegador Web, ou seja, para executar applets, é preciso a instalação de um plugin no mesmo. Navegadores 64-bits não suportam o plugin do Java, apenas as suas respectivas versões de 32-bits. Lembrando que sistemas operacionais 64-bits suportam a execução de programas 32-bits em muitos casos.

Alguns navegadores oficialmente não suportam Java, como o Netscape 7 e 7.1, embora possa funcionar.

Pode ocorrer que, na aplicação, o código Java seja executado apenas no servidor, onde o navegador do cliente recebe apenas código html. Assim apenas o servidor precisa das ferramentas Java. **Neste caso, deve haver no servidor uma aplicação.**

Para executar um programa ou applet Java é necessário ter uma JRE (Java Runtime Environment), que consiste na máquina virtual para interpretar os bytecodes, instalada na máquina. Os pré-requisitos para instalar a versão estável da JRE 6.0 num computador com processador baseado em x86, executando programas em 32 bits e rodando Windows, são 64MB de memória RAM e 98MB de espaço em disco (Windows XP HE, Windows XP Pro, Windows 2000 SP3+).

Para as versões Windows Server 2003 (WE, SE, EE, DCE), são necessários ao menos 128MB de RAM e o mesmo espaço em disco. Num computador com processador baseado em AMD64 ou EM64T, executando programas em 32 bits e rodando o Windows Server 2003, são necessários 128MB de RAM e 98MB de espaço em disco. Para um computador com processador baseado em AMD64 ou EM64T, executando programas em 64 bits e rodando Windows Server 2003, são requeridos 128MB de memória RAM e 110MB de espaço em disco. São necessários também navegadores compatíveis (Mozilla 1.4+ ou seus derivados, Internet Explorer 5.5+).

Para computadores x86 em 32 bits rodando Linux, são necessários 64MB de RAM e 58MB de espaço em disco. O mesmo vale para máquinas AMD64 ou EM64T em 32 bits no Linux. Para computadores AMD64 ou EM64T executando código de 64 bits no Linux, é necessária a mesma memória RAM e espaço em disco de 56MB. É também necessária a compatibilidade com o navegador web (Mozilla 1.4+ ou seus derivados) para executar applets.

6. Sintaxe/modo de uso dos principais elementos e estruturas da linguagem, incluindo:

Um bloco de comandos compreende uma seqüência de comandos contidos entre {}. No caso de haver apenas um comando, o uso das chaves é opcional

6.1. Instruções condicionais: o Java contém três tipos de instruções condicionais.

- IF

Executa o bloco de comandos subseqüente se uma condição for verdadeira ou então segue a execução a partir do fim do bloco de instruções definidos pelo IF se a condição for falsa.

Sintaxe:

if (<condição>) {

<bloco de comandos>
}

•IF...ELSE

Executa o bloco de comandos subseqüente ao if se uma condição for verdadeira ou executa o bloco de comandos subseqüente ao else se a condição for falsa.

Sintaxe:

```
if ( <condição> ) {
    <bloco de comandos>
} else {
    <bloco de comandos>
}
```

• SWITCH

Realiza uma de muitas ações diferentes, dependendo do valor de uma expressão, que pode ser um número do tipo inteiro (int, byte e outros) ou um caractere (char).

Sintaxe:

```
switch ( <expressão> ) {
    case <valor1>:<bloco de comandos 1>
        [break;]
    case <valor2>:<bloco de comandos 2>
        [break;]
    case <valor3>:<bloco de comandos 3>
        [break;]
    case <valor4>:<bloco de comandos 4>
        [break;] ...
    default: <bloco de comandos default>
}
```

Note que o uso do break é opcional, e como em C e C++, a ausência do mesmo ao fim de um bloco de comandos definidos em um determinado case faz com que o case subseqüente (e/ou também o default) sejam executados.

• Operador Ternário

Caso a condição seja verdadeira, fará com que seja executada a expressão1, no caso contrário, a expressão2 é executada. Equivalente ao IF...ELSE.

Sintaxe:

condição ? expressão1 : expressão2

6.2. Instruções de repetição: o java fornece três tipos de estrutura de repetição.

•FOR

Executa o bloco de comandos no seu corpo zero ou mais vezes. Se a condição de continuação do loop for inicialmente falsa, o bloco de comandos não será executado.

Sintaxe:

```
for (<inicialização>; <condição>; <incremento>) {
  <bloco de comandos>
}
```

O funcionamento se dá da seguinte forma:

i. É executada a inicialização e algumas outras instruções que podem estar na 1ª parte da declaração For.

ii. É executada a verificação da condição. Caso seja falsa, há a saída do laço For.

iii. O bloco de comandos é executado até o fim (exceto se houver uma instrução break - que sai do laço do for -, ou uma instrução de continue - que retorna a execução para a 3ª parte da instrução For).

iv.O incremento e outras operações que podem estar na 3ª parte da declaração For são executadas. Ao fim disto, retorna-se ao passo b.

•WHILE

Executa o bloco de comandos no seu corpo zero ou mais vezes. Se a condição de continuação do loop for inicialmente falsa, o bloco de comandos não será executado.

Sintaxe:

```
while (<condição>) {
  <bloco de comandos>
 }
```

•DO WHILE

Executa o bloco de comandos no seu corpo uma ou mais vezes.

Sintaxe:

```
do {
  <bloco de comandos>
 } while (<condição>);
```

6.3. Definição de funções/objetos.

A linguagem Java é orientada a objetos baseada em classes. Uma classe de objeto define os dados que um objeto deverá armazenar e as funções, chamadas de métodos, que manipulam os dados da classe.

Um objeto em Java é definido da seguinte forma:

Sintaxe:

NomeDaClasse nomeDoObjeto = new NomeDaClasse([Parâmetros, se declarados]);

Quando um código em Java possui tarefas complexas, é comum criar um grupo de instruções com tarefas específicas denominadas método.

Sintaxe:

```
modificadores_de_acesso tipo_de_retorno nome_do_método(parâmetros){
   <Bloco de intruções >
}
```

Os modificadores de acesso podem ser private, public, protected e default.

Para cada método é dado um nome. Depois de criado o método este é chamado especificando seu nome e passando valores pelos parâmetros.

6.4.Definição de comentários.

Comentários são importantes para o melhor entendimento do código, tanto pra outro desenvolvedor que venha utilizar o código quanto pra o próprio desenvolvedor. Java dispõe de três formas para adicionar comentários ao código, como:

i. Comentário ocupando várias linhas utiliza-se um par de caracteres: /* (no início do comentário) e */ (ao final do comentário).

ii. Comentário do que houver na mesma linha e à direita de duas barras: //.

iii.Comentário especial que começa com /** e termina com */ para gerar documentação automática com o utilitário javadoc.

6.5.Mecanismo de controle de erros (exceções)

O tratamento de exceções em Java é o mecanismo responsável pelo tratamento da ocorrência de condições que alteram o fluxo normal da execução de programas. Ela fornece ao programador um mecanismo simples e poderoso para tratar eventos excepcionais durante a execução de um programa.

Sintase:

```
try{
   instruções
}catch( Tipo de exceção1){
   instrução de tratamento de exceções
} catch( Tipo de exceção2){
   instrução de tratamento de exceções
}
...
finally{
   instruções
   instruções de liberação de recursos
}
```

O bloco de comandos dentro da cláusula finally será sempre executado.

Palavras-chave:

Throws repassa uma exceção adiante na pilha de chamadas de método. É utilizada na assinatura do método.

```
public double dividir (double a, double b) throws ArithymeticException{

return a/b;

}
```

Throw lança uma exceção de forma explícita no código.

```
public double dividir (double a, double b) {

if (b == 0)
```

```
        throw  new ArithymeticException("Tentativa de divisão por zero.")
    return a/b;

    }
```

6.6. Mecanismo de acesso ao banco de dados e arquivos

Mecanismo de acesso a banco de dados

Os programas Java se comunicam com bancos de dados e manipulam seus dados utilizando a API do JDBC (Java Database Connectivity). Um driver JDBC permite aos aplicativos Java conectar-se a um banco de dados particular e permite aos programadores manipular esse banco de dados utilizando a API do JDBC. Geralmente este driver JDBC é fornecido pelo fabricante do SGDB ou então é desenvolvido e disponibilizado pela comunidade.

O JDBC é quase sempre utilizado com um banco de dados relacional. Entretanto, ele pode ser utilizado com qualquer origem de dados baseada em tabelas.

Arquitetura do JDBC

A biblioteca do JDBC é composta por um conjunto de interfaces de acesso ao banco de dados. Uma implementação em particular dessas interfaces é chamada de Driver. Cada banco de dados possui um Driver específico, usado de forma padrão.

Principais classes pertencentes ao JDBC (pacote java.sql)

DriverManager - gerencia o driver e cria uma conexão com o banco.

Connection - é a classe que representa a conexão com o bando de dados.

Statement - controla e executa uma instrução SQL .

PreparedStatement - controla e executa uma instrução SQL. É melhor que Statement.

CallableStatement – permite a execução de Stored Procedures.

ResultSet - contém o conjunto de dados retornado por uma consulta SQL.

ResultsetMetaData - é a classe que trata dos metadados do banco.

Acessando os dados com o JDBC

1 - Carregamento do driver.

Class.forName(“nome do driver”);

2 - Criação da conexão com o banco de dados

Connection conn =
DriverManager.getConnection(“url”,“usuario”,“senha”);

3 – Interagindo com o banco de dados

```
Statement st = conn.createStatement();
String sql = ”INSERT INTO PESSOA VALUES (2,’Maria’)";
st.executeUpdate( sql );
sql = ”INSERT INTO PESSOA VALUES (3,’Jurema’)";
st.executeUpdate( sql );
st.close();
conn.close();
```

O método createStatement() de Connection retorna um objeto Statement, que é usado para executar um comando SQL no banco de dados.

O método executeUpdate() de Statement recebe o SQL a executado. Este método deve ser usado para comandos SQL de insert, update, delete, etc.

Os métodos close() de Statement e Connection são invocados para liberar os recursos.

4 – Recuperando os dados

```
String sql = "SELECT id, nome FROM pessoa";
ResultSet rs = st.executeQuery( sql );
while( rs.next() ) {
System.out.println( rs.getString("id") );
System.out.println( rs.getString("nome") );
}
rs.close();
st.close();
conn.close();
```

O método executeQuery() de Statment executa uma consulta (query) SQL e retorna um objeto ResultSet, que contém os dados recuperados.

O método next() de ResultSet, faz com que o cursor interno é movido para o próximo registro. O método retorna false caso não existam mais registros ou true caso contrário.

5 – Executando Stored Procedures

```
String sql = "{? = call SP_Gera_Relatorio_Pessoa(?)}";
CallableStatement cstmt = conn.prepareCall(sql);

cstmt.registerOutParameter(1, Types.INTEGER);
cstmt.setString(2, cpf);
cstmt.executeQuery();

ResultSet rs = cstmt.getResultSet();
```

Como O CallableStatement é herdeiro da classe PreparedStatement, o mesmo também retorna um ResultSet. Note que na String sql podemos ter o saída da Stored Prcedure e a entrada, como uma lista de parâmetros. O método **registerOutParameter** ajusta o tipo que deve ser recebido na saída da Stored Procedure e a parte do CallableStatement que deve receber esta definição. Já o métodos **setString, setInt**, por exemplo tratam apenas do ajuste dos parâmetros que serão recebidos pela Stored Procedure.

6 – Controlando transações

A interface Connection oferece os métodos commit() e rollback para o controle transacional. Se quisermos confirmar uma transação, o método commit() deve ser invocado. Já o método rollback desfaz a transação.

Além de todas as funcionalidades citadas anteriormente, a API do JDBC também oferece a possibilidade de tratar erros, acessar procedures, dentre outras.

Mecanismo de acesso a arquivos

O pacote **java.**io oferece um conjunto de classes para se trabalhar com arquivos, diretórios e seus dados, independente de plataforma. Ela também oferece meios para manipulação de dados durante o processo de leitura e gravação.

Dentro do pacote **java.io** contém a classe **File**, que é usada para representar o sistema de arquivos. A existência de um objeto dessa classe não garante a existência de um arquivo ou diretório. A classe contém vários métodos para testar a existência de arquivos, apagar arquivos, criar diretórios, listar o conteúdo de diretórios, etc.

Alguns métodos da classe File:

String getAbsolutePath()
String getParent()
boolean exists()
boolean isFile()
boolean isDirectory()
boolean delete()
boolean mkdir()
String[] list()

Classes e interfaces para entrada e saída de dados
Entrada e saída de bytes : InputStream e OutputStream
Entrada e saída de chars:
Reader (lida com fluxos de caracteres de entrada)
Writer (lida com fluxo de caracteres de saída).

Principais Saídas
FileWriter
BufferedWriter (gravação de texto com buffer)
PrintWriter

Principais Entradas
FileReader
BufferedReader (leitura de texto com buffer)

Exemplos: ler e gravar em arquivos

Gravar no arquivo:

```
File file  = new File(caminho do arquivo);
```

```
FileWriter escrita = new FileWriter(file);

char[] chars = new char[file.length()];
for(int i = chars.length -1 ; i >= 0; i--)
escrita.write(chars[i]);
```

Ler do arquivo:

```
File file  = new File(caminho do arquivo);
FileReader leitura = new FileReader(file);
leitura.read();
```

6.7 Mecanismo de acesso a dispositivos externos (dispositivos biométricos, impressoras, scanners etc.)

Java não tem uma biblioteca específica para acessar dispositivos USB. Para utilizar estes tipos de dispositivos, é necessário utilizar uma API de terceiros, como a jUSB, da IBM, ou até mesmo construir sua própria API. Existe uma API que é forte candidata a se tornar oficial no Java: a JSR-80, construída para sistemas Linux.

Exemplos de código:

```
//Executando E/S com a API JSR-80
import javax.usb.*;
import java.util.List;
        public class TraverseUSB
{
   public static void main(String argv[])
   {
      try
      {
         // Acessa o sistema de serviços USB, e acessa o hub raiz.
         // Então percorre através do hub raiz.
         UsbServices services = UsbHostManager.getUsbServices();
         UsbHub rootHub = services.getRootUsbHub();
         traverse(rootHub);
```

```
    } catch (Exception e) {}
}

public static void traverse(UsbDevice device)
{
    if (device.isUsbHub())
    {
        // Se este é um hub USB, percorre através do hub.
        List attachedDevices =
        ((UsbHub) device).getAttachedUsbDevices();
        for (int i=0; i<attachedDevices.size(); i++)
        {
            traverse((UsbDevice) attachedDevices.get(i));
        }
}
    else
    {
        //Esta é uma função USB, não um hub.
        //Faz alguma coisa...
    }
}
```

Em relação a dispositivos biométricos, Java possui um pacote chamado Java Authentication and Autorization Service(JAAS) que aprimora a plataforma Java com ferramentas para autenticação e controle de acesso para usuários.Dentre as classes e interfaces principais do java encontra-se a LoginModule que descreve a interface implementada pelos provedores de tecnologias de autenticação. São encaixados como base para as aplicações, provendo serviços de um tipo de autenticação específica. Por exemplo, um tipo de LoginModule pode fornecer autenticação baseada no login/senha, enquanto que outro LoginModule pode interagir com hardwares especiais e fornecer autenticação baseada em smartcards ou dispositivos biométricos. Os principais métodos dessa classe são as seguintes: login(), commit(), abort() e logout().

6.8. Uso de bibliotecas/controles feitos na própria linguagem.

Java possui uma rica coleção de classes existentes nas bibliotecas de classe Java, que também são conhecidas como APIs do Java ou Java APIs (application programming interfaces). O ambiente padrão de desenvolvimento do Java já contém uma série de bibliotecas já prontas para os desenvolvedores utilizarem.

Para fazer uso dessas bibliotecas no código, é necessário importá-las para o mesmo.

Exemplos:

import java.io.* (importa todo o conteúdo do pacote)

ou

java.io.BufferedInputStream (importa apenas uma classe ou interface específica)

6.9.Forma e disponibilidade para acesso a bibliotecas/controles desenvolvidos em outras linguagens (método de interação entre sistemas) – Exemplo: a linguagem de programação escolhida pode acessar bibliotecas em C++? E em Java?

Para fazer uso de outras bibliotecas ou até mesmo controles desenvolvidos em outras linguagens, o Java faz uso da API Java Native Interface (JNI). Esse recurso é utilizado em situações que o Java não tem o suporte necessário para tarefas específicas ou não apresenta um bom desempenho.

Exemplo:

```
class HelloWorld {
   public native void mostrarOlaMundo();
      static {
            System.loadLibrary("Ola");
          }

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

7. A linguagem possui utilitários ou formas de automação/auxílio na documentação?

Sim. A Sun disponibiliza o utilitário Javadoc como parte de seu JDK (Java Development Kit) desde a versão 1.2. Esta ferramenta gera automaticamente documentação em formato HTML a partir de comentários *doc* inseridos no código-fonte Java. Em conjunto com o Javadoc, é fornecido um kit de ferramentas chamado Doclet API. Este kit oferece recursos para personalizar o doclet padrão (a maneira clássica como o Javadoc gera documentação) ou até mesmo criar do zero outros doclets, ou seja, outras formas de saída para o Javadoc.

8. Pode ser utilizado para desenvolver aplicações Web? E "desktop"?

Sim. O Java oferece possibilidade tanto de desenvolvimento de aplicações Web quanto de aplicações Desktop, além de aplicações móveis. Para cada uma dessas aplicações o Java disponibiliza diferentes plataformas (citadas na questão 3). No desenvolvimento voltado para Web é utilizada a plataforma J2EE, a qual se dispõem de componentes tais como JSP, Servlets, EJB etc. Já no desenvolvimento voltado para aplicações Desktop o Java utiliza-se de bibliotecas como Swing, Abstract Windowing Toolkit (AWT), além de produtos de terceiros. A biblioteca AWT foi inicialmente o padrão para renderização de controles gráficos nas aplicações Desktop. Posteriormente o Java2 padronizou a biblioteca Swing a qual implementava os controles visuais, utilizando apenas as primitivas gráficas da plataforma.

9. Permite desenvolver controles personalizados? Como?

Sim. A biblioteca gráfica Swing, que faz parte da API padrão do Java, é altamente modularizada e extensível. Isso permite "plugar" várias implementações personalizadas de interfaces específicas do framework. O Swing permite modificações via extensão do framework existente ou mesmo pela criação de implementações alternativas dos componentes essenciais da biblioteca.

Inicialmente, para desenvolver controles personalizados utilizando-se o WebLogic Workshop é preciso criar uma pasta específica para o controle. Não se pode criar o controle no root do projeto. Não é preciso criar o arquivo Java, nem mesmo editá-lo. Na verdade, o próprio WebLogic Workshop JSC cria o arquivo Java enquanto o desenvolvedor necessita apenas se preocupar para a criação do arquivo JSC de extensão "Impl". Quando é preciso editar o arquivo, isto é feito no arquivo JSC e o

WebLogic Workshop atualiza o código do arquivo em Java, e estas mudanças se refletem no comando da interface.

10. Quais os ambientes de desenvolvimento integrados (IDEs) disponíveis ? – Informar fabricante, site, tipo de licença, características e valor(se for o caso) de cada uma delas.

Abaixo estão listadas as principais IDEs utilizadas:

→ NetBeans

Fabricante : SUN

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

Tipo de licença: Common Development and Distribution License (CDDL)

Características: IDE gratuito, open-source para programadores de software. Possui todas as ferramentas para criação de aplicações desktop, empresariais, web e móveis, em Java, C/C++ e até Ruby. Suporta muitas plataformas como Windows, Linux, Mac OS X e Solaris.

Valor: Gratuito

→ Eclipse

Fabricante : IBM

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

Tipo de licença: Eclipse Public License (EPL)

Características:IDE gratruito. Possuí características interessantes como: refactoring, building, running e debugging de aplicações, fácil integração com os controladores de versão mais conhecidos do mercado, e um grande número de plugins que atendem diferentes necessidades dos desenvolvedores.

Valor: Gratuito

→JBuilder

Fabricante : Borland CodeGear

Site: http://www.codegear.com/br/products/jbuilder

Tipo de licença: Proprietária

Características: É uma IDE que tem como principal utilização, a construção de aplicações gráficas a partir de JFrames. É a única com suporte a virtual peer programming, ou seja, é possível que desenvolvedores locais e remotos em conjunto conceba, edite e depure aplicações em tempo real.

Valor: Versão de teste gratuito.

Profissional – R$ 1.197,60

→JDeveloper

Fabricante : Oracle

Site: http://www.oracle.com/technology/products/jdev/index.html

Tipo de licença: Oracle Technology Network Early Adopter License

Características: Oracle JDeveloper é uma IDE grátis que simplifica o desenvolvimento de aplicações baseadas em Java que fazem uso de SOA.

Valor:Gratuito

→JCreator

Fabricante :Xinox

Site: http://www.jcreator.com/

Tipo de licença:Proprietário

Características: Escrito inteiramente em C++, o que o torna mais rápido e eficiente comparado a outras IDEs escritas em Java.

Valor: 1 usuário U$ 89, 5 usuários U$ 399, 10 usuários U$ 600, 20 usuários U$ 1099, 30 usuários U$ 1600, sob demanda U$ 1099 - U$ 5000.

11. A linguagem é padronizada (ANSI/ISO)? Qual a referência (documento, data, ano)?

A Sun tentou padronizar a linguagem Java pelos órgãos ISO/IEC no ano de 1997, porém não obteve êxito. Em 2006 foi lançada pela Sun, de acordo com a GNU, a versão do Java em software livre e em 2007 foi concluído a disponibilização de praticamente todo código Java como software de código aberto. A parte do código que a Sun não possui copyright foi a única a não ser disponibilizada.

No site da Sun (http://java.sun.com/docs/codeconv/) é possível encontrar um documento com a convenção de codificação Java oficial da Sun, revisado e atualizado em 20 de abril de 1999. Este documento contém os padrões a serem seguidos para melhor legibilidade do código a fim de se obter mais facilidade na manipulação do mesmo.

Fontes de Pesquisa

1. Livro: Java Como Programar, 6ª edição - Deitel.
2. http://www.infotem.hpg.ig.com.br/lin_progr_java.htm#serajava
3. Java 1001 dicas de programação, Mark C. Chan, Steven W. Griffith e Anthony F. Iasi .Amkron Books
4. Desenvolvimento em Java para Desktops Livres - (http://www.lozano.eti.br/palestras/java-desktop-livre.pdf)
5. http://java.sun.com/j2se/javadoc/
6. http://www.eclipse.org/
7. Introdução ao Eclipse RCP - (http://www.devmedia.com.br/articles/viewcomp.asp?comp=4352)
8. http://www.javafree.org/content/view.jf?idContent=85
9. http://www.di.ufpe.br/~java/jai/aula7/index.html
10. http://www.codegear.com/br/products/jbuilder
11. http://www.oracle.com/technology/software/products/jdev/htdocs/soft11tp.html
12. http://www.ibm.com/developerworks/linux/library/j-usb.html
13. http://www.dimap.ufrn.br/~flavia.delicato/JDBC.pdf
14. http://edocs.bea.com/workshop/docs81/doc/en/core/index.html