Grupo JavaRS
JUG Rio Grande do Sul

http://www.inf.ufrgs.br/tools/java

Linguagem Java

por Leandro Soares Indrusiak
lsi@inf.ufrgs.br

1996

1 Conceito

Java é uma linguagem computacional completa, adequada para o desenvolvimento de aplicações baseadas na rede Internet, redes fechadas ou ainda programas stand-alone [CAM96].

Foi desenvolvida na 1a metade da década de 90 nos laboratórios da Sun Microsystems com o objetivo de ser mais simples e eficiente do que suas predecessoras. O alvo inicial era a produção de software para produtos eletrônicos de consumo (fornos de microondas, agendas eletrônicas, etc.). Um dos requisitos para esse tipo de software é ter código compacto e de arquitetura neutra.

A linguagem obteve sucesso em cumprir os requisitos de sua especificação, mas apesar de sua eficiência não conseguiu sucesso comercial. Com a popularização da rede Internet, os pesquisadores da Sun Microsystems perceberam que aquele seria um nicho ideal para aplicar a recém criada linguagem de programação. A partir disso, adaptaram o código Java para que pudesse ser utilizado em microcomputadores conectados a rede Internet, mais especificamente no ambiente da World Wide Web. Java permitiu a criação de programas batizados applets, que trafegam e trocam dados através da Internet e se utilizam da interface gráfica de um web browser. Implementaram também o primeiro browser compatível com a linguagem, o HotJava, que fazia a interface entre as aplicações Java e o sistema operacional dos computadores.

Com isso, a linguagem conseguiu uma popularização fora de série, passando a ser usada amplamente na construção de documentos web que permitam maior interatividade. Os principais web browsers disponíveis comercialmente passaram a dar suporte aos programas Java, e outras tecnologias em áreas como computação gráfica e banco de dados também buscaram integrar-se com o novo paradigma proposto pela linguagem: aplicações voltadas para o uso de redes de computadores.

Atualmente, a linguagem Java é a força propulsora por trás de alguns dos maiores avanços da computação mundial, como:

- Acesso remoto a bancos de dados
- Bancos de dados distribuídos
- Comércio eletrônico no WWW
- Network CAD
- Interatividade em páginas WWW
- Interatividade em ambientes de Realidade Virtual distribuídos

- Gerência de Documentos
- Integração entre dados e forma de visualização
- Network Computer
- Ensino à distância
-Jogos e entretenimento

2 Características

Java é uma linguagem poderosa em ambientes distribuídos complexos como a rede Internet. Mas sua versatilidade permite ao programador ir além, oferecendo uma poderosa linguagem de programação de uso geral, com recursos suficientes para a construção de uma variedade de aplicativos que podem ou não depender do uso de recursos de conectividade [WUT97] .

As principais características da linguagem foram divulgadas pela primeira vez em The Java Language: A White Paper em 1995 [GOS95]. A seguir, fazemos um detalhamento geral da linguagem a partir das características presentes nesse documento.

2.1 Simplicidade e eficiência de código orientado a objetos

Java é uma linguagem simples, de fácil aprendizado ou migração, pois possui um reduzido número de construções. A diminuição das construções mais suscetíveis a erros de programação, tais como ponteiros e gerenciamento de memória via código de programação também faz com que a programação em Java seja mais eficiente. Contém um conjunto de bibliotecas que fornecem grande parte da funcionalidade básica da linguagem, incluindo rotinas de acesso à rede e criação de interface gráfica.

Baseada no paradigma da Orientação a Objetos - encapsulamento em um bloco de software dos dados (variáveis) e métodos de manipulação desses dados - a linguagem permite a modularização das aplicações, reuso e manutenção simples do código já implementado.

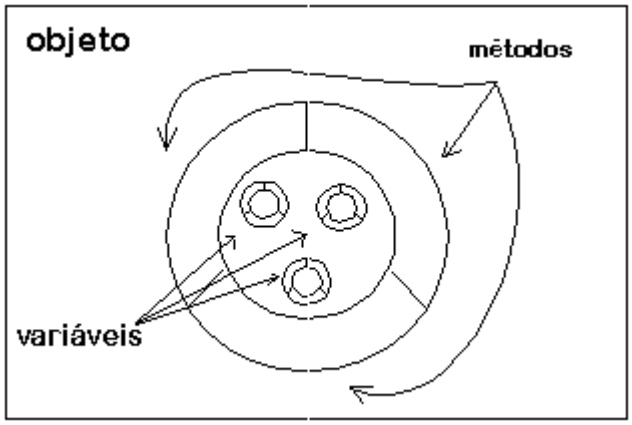

FIGURA 1 - Encapsulamento de variáveis e métodos em um objeto

2.2 Código Interpretado e Portável

As diversas linguagens de programação podem ser tanto compiladas como interpretadas. Quando se utiliza uma linguagem compilada, é necessário executar um programa para traduzir os arquivos fonte, legíveis em linguagem de alto nível, em código executável. As linguagens compiladas têm a vantagem de produzir código de alta performance, o qual está ajustado para o funcionamento em um tipo específico de processador ou arquitetura de processador. Aplicativos compilados, chamados de código binário, só podem rodar no tipo de computador para o qual foram compilados, uma vez que esses aplicativos consistem, na realidade, em instruções em linguagem de máquina, entendidas e executadas pelo microprocessador.

As linguagens interpretadas só existem em código fonte. Quando em execução, um programa chamado interpretador toma o código fonte e executa as ações indicadas pelos comandos no arquivo. O interpretador é, na realidade, o único aplicativo que está sendo executado. Entre os benefícios das linguagens interpretadas está o fato dos programas interpretados poderem rodar em uma variedade de plataformas diferentes, pois estes só existem em código fonte. Além disso são mais fáceis de depurar.

A linguagem Java é tanto compilada como interpretada. Após escrever um programa em Java, utilizando um editor de textos qualquer, salva-se o programa como código fonte. A seguir, pode-se compilar esse fonte, a fim de produzir um tipo de arquivo binário chamado de arquivo de classe. Esses arquivos não são executados diretamente pois eles não contêm instruções que são entendidas diretamente pelos processadores atualmente disponíveis no mercado. Os programas Java são compilados em um formato intermediário chamado bytecodes. Assim, esses programas podem ser executados em qualquer sistema através de um interpretador Java (runtime environment). Com isso, o código precisa ser escrito e compilado apenas uma vez, pois os bytecodes gerados serão executados da mesma forma em qualquer plataforma de hardware e software.

A Sun Microsystems atualmente está trabalhando em compiladores específicos para uma plataforma de hardware. Se o hardware onde o código Java será executado é conhecido, pode-se melhorar a performance do aplicativo. Em contrapartida, não será mais possível rodar esse código em outras plataformas.

Além disso, processadores que executem os bytecodes Java sem a necessidade de um interpretador também estão sendo colocados no mercado pela Sun.

2.3 Segurança

Por ter seu projeto voltado para a simplicidade de código, as possibilidades de erro de programação em Java são reduzidas. Apesar disso, a linguagem traz outros recursos para tornar seu código ainda mais eficiente. O processo de compilação - geração de bytecodes - é projetado para a detecção prévia dos possíveis erros, evitando que os erros se manifestem em tempo de execução. O uso de código para tratamento de exceções · exception handling - permite manter a consistência da aplicação no caso de erros.

Além de diminuir as possibilidades de erro de programação, a linguagem tem um esquema de segurança para garantir a integridade de código - principalmente no caso do código originário de rede insegura.

Esses recursos de segurança são notáveis principalmente dentro do ambiente do interpretador [GON97]. Após baixar um applet da rede, o interpretador faz uma verificação do código, buscando alterações intencionais ou não. A seguir, o interpretador determina o layout de memória para execução. Em outras palavras, não é possível acessar informações diretamente da memória ou inserir código estranho ao código original. Além disso, um programa em Java não pode acessar o sistema de arquivos, salvo nos casos previstos pelo cliente: diferente conjunto de permissões de acordo com a origem das aplicações [MCG97].

2.4 Aplicações distribuídas e processamento paralelo

Com a larga utilização dos recursos da rede Internet, a linguagem teve seu projeto voltado para as aplicações em rede. Assim, a linguagem traz classes para o suporte a vários níveis de conectividade: acesso a URLs (padrão Internet), uso de conexões em sockets, criação de protocolos, criação de clientes e servidores. A introdução desses conceitos se deu de forma simples para o programador, pois permite a ele o acesso às informações da rede com a mesma facilidade do acesso aos arquivos locais.

Para permitir uma melhor performance de execução, mesmo em tarefas de maior complexidade, a linguagem permite a programação de threads - processos de execução de tarefas que ocorrem simultaneamente. A linguagem traz também mecanismos para sincronização, ativação e desativação parametrizada desses processos.

3 Recursos para desenvolvimento em Java

Ao lançar a primeira versão pública da linguagem, a Sun Microsystems preocupou-se em fornecer aos futuros desenvolvedores de aplicações Java um pacote de ferramentas e bibliotecas básicas. Esse pacote, chamado Java Development Kit [KRA97], é indispensável para o desenvolvedor iniciante. Posteriormente, quando estiver mais familiarizado com a linguagem, o desenvolvedor pode migrar para algum outro ambiente de desenvolvimento mais sofisticado (existem vários disponíveis comercialmente), já que o JDK não tem uma interface muito amigável.

3.1 JDK Tools

O JDK oferece ao desenvolvedor uma série de ferramentas, como o interpretador, o compilador, o debugger, o gerador de documentação, o visualizador de applets, o visualizador de bytecodes, o compactador de classes, o gerador de assinaturas digitais, entre outros. As ferramentas mais freqüentemente utilizadas são:

3.1.1 javac

javac é o compilador da linguagem Java. Ele lê arquivos fonte .java e gera arquivos de classe .class no formato de bytecodes. Para cada classe especificada (pode-se especificar mais de uma classe em cada arquivo fonte) é gerado um arquivo de classe chamado NomedaClasse.class. Importante: o arquivo fonte deve ter o nome da classe que ele define seguido da extensão .java, caso contrário o compilador acusa erro.

3.1.2 java

O interpretador java é utilizado para executar aplicações em Java. Ele interpreta os bytecodes gerados pelo javac. O interpretador é executado a partir de linha de comando da forma:

java NomedaClasse arg1 arg2 arg3...

Ao ser executado, o interpretador busca o arquivo NomedaClasse.class, nele procura um método public static void main() e a ele passa os argumentos que recebeu (ou não, pois os argumentos são opcionais e sua existência depende da funcionalidade da classe que está sendo interpretada).

3.1.3 appletviewer

O appletviewer é uma ferramenta para visualização de applets fora do ambiente web browser. Executado a partir de linha de comando, recebe como argumento o arquivo HTML ao qual está anexo o applet Java. Esta ferramenta é bastante útil no processo de desenvolvimento e teste de applets.

3.1.4 javadoc

O javadoc é um gerador automático de documentação. Ele lê declarações de classes - código fonte - e busca por comentários especiais (iniciam com /** e terminam com */), gerando páginas HTML com as informações contidas nesses comentários, bem como informações sobre os métodos, variáveis e herança da classe declarada.

3.2 Java API

O conjunto de bibliotecas inclusas no JDK é conhecido como Java API (Interface de Programação de Aplicações). Nessas bibliotecas estão um conjunto grande de classes, organizadas em pacotes. Cada um desses pacotes traz classes com funcionalidade básica e vital para um determinado ramo de programação Java. A seguir, uma lista dos pacotes disponíveis na versão 1.1 do JDK. Os pacotes não marcados com + também estão disponíveis na versão 1.0.

Tabela 1 - Lista de pacotes da linguagem Java

java.applet	+ java.lang.reflect	+ java.security
+ java.awt.datatransfer	+ java.math	+ java.security.interfaces
+ java.awt.event	java.net	+ java.sql
java.awt	+ java.rmi.dgc	+ java.text
java.awt.image	+ java.rmi	java.util
+ java.beans	+ java.rmi.registry	+ java.util.zip
java.io	+ java.rmi.server	
java.lang	+ java.security.acl	

Os pacotes de uso mais comum em aplicações básicas são:

3.2.1 java.applet

Contém as classes necessárias para o desenvolvimento de applets. Basicamente, todos os métodos para a criação de um aplicativo a ser rodado dentro de um web browser são encontrados dentro de uma única classe do package: a classe Applet.

3.2.2 java.awt

Contém classes relacionadas à interface gráfica [ZUC97], como botões, caixas de texto, menus, etc. Contém ainda classes para processamento de imagens e as classes que são empregadas no tratamento dos eventos gerados pela interface gráfica (clique de mouse sobre um botão, seleção de um item de um menu, etc.), no pacote java.awt.event.

3.2.3 java.io

Classes para entrada e saída de dados das mais variadas formas. Torna transparente ao usuário as características do sistema de arquivos da plataforma na qual o programa está rodando.

3.2.4 java.lang

São as classes que dão suporte ao modelo computacional da linguagem. São indispensáveis para o funcionamento de qualquer programa Java.

3.2.5 java.net

Contém classes aptas a estabelecer conexões de rede. Possuem métodos específicos para a rede Internet, usando referências a URLs e conexões usando TCP/IP.

3.2.6 java.util

Contém uma série diversa de classes de apoio ao programador, como estruturas de dados básicas (hash tables, pilhas, etc.), referência à data do sistema, gerador de números randômicos.

4 Programas Java - Applets e Applications

Os programas escritos em Java podem assumir duas formas básicas: applets e applications. A programação desses dois tipos é baseada nos mesmos conceitos da linguagem, mas têm características bem distintas. Segurança, interface gráfica, acesso à unidades de disco e acesso à rede são pontos de divergência entre applets e applications.

Entretanto, a diferença básica está no fato de que applets precisam de um web browser para existir. Isso quer dizer que applets são aplicações voltadas para o ambiente Internet/Intranet e que são transportadas pela rede junto com hiperdocumentos HTML. Já as applications são programas escritos para operar sem a necessidade de um web browser - aplicações convencionais para computadores, como editores de texto, gerenciadores de arquivos, etc.

A seguir, temos exemplos de applets e applications, bem como as características de cada um.

4.1 Applications

Applications são aplicativos stand-alone escritos em Java. São, na realidade, classes independentes que o interpretador reconhece e executa. O método principal é nomeado como main(). Ao reconhecer uma application, o interpretador chama o método main(), que deve iniciar o funcionamento de toda a aplicação.

Principais características das applications:

- não necessitam de um web browser para montar sua interface gráfica, ou seja, precisam criar janelas para montar a interface gráfica, ou ainda operar em linha de comando
- não têm restrições de acesso a unidades de disco
- não têm restrição de acesso à rede

A seguir na Figura 2, temos um exemplo simples do código de um application. Esse código deve ser armazenado em um arquivo de mesmo nome da classe que ele define, seguido da extensão **.java**.

Arquivo MeuApplication.java

```
public class MeuApplication {
        public static void main (String[] args) {
                System.out.println( “Este é meu application!” );
        }
}
```

FIGURA .2 - classe MeuApplication

4.2 Applets

Um applet é um tipo específico de aplicativo que é dependente de um navegador web. Em vez de ter um método main(), um applet implementa um conjunto de métodos que lidam com situações tais como inicialização, quando e como desenhar a tela, o que fazer quando ocorre um clique de mouse e assim por diante. Os navegadores habilitados para Java se beneficiam do fato dessa linguagem ser dinâmica, colocando applets ligados a páginas, carregando-os automaticamente quando essas páginas forem carregadas. O applet passa a fazer parte do navegador quando ocorre a sua execução.

Principais características dos applets:

- necessitam de um web browser para montar sua interface gráfica
- tem restrições de acesso a unidades de disco - só acessa com a permissão explícita do usuário (alguns web browsers não permitem o acesso a disco em nenhuma circunstância)
- tem restrição de acesso à rede - podem acessar apenas o site do qual vieram (alguns web browsers permitem ao usuário definir permissões para que o applet acesse outros sites)
- utilizam os métodos init( ), start( ), stop( ) e destroy( ) para definir seu ciclo de vida:

    init( ) - método chamado imediatamente após a criação do applet. É chamado pelo web browser na primeira vez que o applet é carregado.

    start( ) - método chamado pelo web browser toda a vez que o applet é materializado na tela.

    stop( ) - método chamado pelo web browser sempre que o applet deixa de ser visível

    destroy( ) - método invocado quando todos os recursos computacionais alocados pelo applet precisam ser liberados.

A seguir, temos um exemplo simples do código de um applet. Esse código deve ser armazenado em um arquivo de mesmo nome da classe que ele define, seguido da extensão **.java**.

arquivo MeuApplet.java

```
import java.applet.*;
import java.awt.*;

public class MeuApplet extends Applet {
        public void paint (Graphics g) {
                g.drawString( “Este é meu applet!” );
        }
}
```

FIGURA 3 - classe MeuApplet

Após o processo de compilação do arquivo MeuApplet.java, que pode ser efetuado pelo compilador **javac** encontrado no JDK, é necessário criar um arquivo HTML que contenha a chamada para o applet. Esse arquivo HTML é que será chamado pelo web browser. O exemplo a seguir mostra o arquivo HTML com a chamada ao applet:

Arquivo MinhaApplet.html

```
<HTML>
<applet code=“MeuApplet.class” width=“200” height=“100”>
</applet>
</HTML>
```

FIGURA 4 - Código HTML para inclusão de um applet em documento WWW

Os applets são carregados e formatados dentro de uma página web de forma semelhante a uma imagem. Na maioria dos browsers, quando o arquivo HTML indica que uma imagem deve ser colocada na página, a imagem é carregada a partir do servidor web e exibida no local apropriado: a imagem é desenhada dentro da janela do navegador, tendo o texto fluindo em torno desta, em vez de dispor de uma janela externa exclusiva.

Em um browser que suporte Java, o applet também é exibido dentro da página web. Consequentemente, nem sempre o usuário pode ter certeza se uma imagem materializada em seu browser é um arquivo de imagem ou um applet. Quando um applet é colocado em uma página web, é definida uma área específica para a sua exibição. Essa área de exibição pertence ao applet, para ser utilizada conforme a sua execução. Alguns applets utilizam essa área para apresentar animações; outros a utilizam para exibir informações a partir de um banco de dados ou para permitir que o usuário selecione itens ou digite informações. A largura e altura dessa área são definidas no arquivo HTML, dentro da chamada ao applet, nos campos width e height.

Bibliografia

[ADI97] ADIDA, B. It all starts at the server. **IEEE Internet Computing**, Los Alamitos, Feb. 1997. p. 75.

[ADI97a] ADIDA, B. Taking Web Clients to the Next Level. **IEEE Internet Computing**, Los Alamitos , p. 65-67, Mar./Apr. 1997.

[ALB93] ALBERTI, B. et al. **The Internet Gopher Protocol**. University of Minnesota, IETF RFC 1436, 1993. Disponível por WWW em http://reference.nrcs.usda.gov/ietf/rfc/1500/rfc1436.txt.

[BAL93] BALASUBRAMANIAN, V. **State of the Art Review on Hypermedia Issues And Applications**. Newark, NJ: Rutgers University, 1993.

[BER93a] BERNERS-LEE, T.; CONNOLY, D**. Hypertext Markup Language (HTML)**. Internet Draft (work in progress), Jul.1993. Disponível por WWW em http://info.cern.ch/hypertext/WWW/MarkUp/HTML.html.

[BER93b] BERNERS-LEE, T. **Uniform Resource Locators (URL)**. Internet Draft (work in progress), Oct.1993. Disponível por FTP anônimo em ftp.w3.org, no arquivo /pub/www/doc/url-spec.txt.

[BER94] BERNERS-LEE, T. et al. The World-Wide Web. **Communications of the ACM**, New York, vol. 37, no. 8, p. 76-82, 1994.

[BER95] BERNERS-LEE, T; FIELDING, R. T.; NIELSEN, H. F. **Hypertext Transfer Protocol - HTTP/1.0**. Internet Draft (work in progress) - HTTP Working Group, Mar.1995. Disponível por FTP anônimo em ftp.w3.org, no arquivo /pub/www/doc/http-spec.txt.

[BOR97] BORLAND INTERNATIONAL INC. **JBuilder Professional**. 1997.

[BRA94] BRANDÃO, H.J.R., GONDIN, P.R.L. Terminal Virtual. In: CARVALHO, T.C.H. de B. (Org.). **Arquiteturas de Redes de Computadores OSI e TCP/IP.** São Paulo: Makron Books, 1994. p. 473-505.

[CAM96] CAMPIONE, M; WALRATH, K. **The Java Tutorial**: Object-Oriented Programming for the Internet. [S.l.]: SunSoft Press, 1996.

[CHP97] CHANG, P.I. **Java Web Server Ships**. JavaWorld, IDG Communications, June 1997. Disponível por WWW em http://www.javaworld.com/javaworld/jw-06-1997/jw-06-webserver.html.

[CON87] CONKLIN, Jeff. Hypertext: An Introduction and Survey. **IEEE Computer**, Los Alamitos, v.20, n.9. p.17-40, Sep.1987.

[DON95] DONNELLY, C.; STALLMAN, R. **The Bison Manual**. [S.l.]: Free Software Foundation, 1995, 104 p.

[FIE97] FIELDING, R.T.; KAISER, G. The Apache HTTP Server Project. **IEEE Internet Computing**, Los Alamitos, p. 88-90, Jul./Aug. 1997.

[FLA96] FLANAGAN, D. **Java in a Nutshell**. [S.l.]: O'Reilly & Associates, 1996. 460 p.

[GON97] GONG, L. Java Security: present and near future. **IEEE Micro**, Los Alamitos, p. 14-19, May/Jun. 1997.

[GOS96] GOSLING, J.; JOY, B.; STEELE,G. **The Java Language Specification**. Sun Microsystems, July 1996. Disponível por WWW em http://java.sun.com/doc/language_specification.html.

[GOS97] GOSLING, J. The Feel of Java. **IEEE Computer**, Los Alamitos, p. 53-57, June 1997.

[GRA96] GRALLA, P. **Como Funciona a Internet**. [S.l.]: Quark do Brasil, 1996.

[HOR96] HORSTMANN, C.; CORNELL, G. **Core Java**. SunSoft Press, Prentice-Hall, 1996.

[IND95] INDRUSIAK, L.S. **Relatório de Estágio**. Santa Maria: Centro de Processamento de Dados, Universidade Federal de Santa Maria, 1995.

[JAV97] JAVA.SUN.COM. **The Source for Java**. Disponível por WWW em http://java.sun.com (1997).

[KRA96] KRAMER, D. **The Java Platform:** A White Paper. [S.l.]: Sun Microsystems, 1996.

[KRA97] KRAMER, D. **JDK Documentation**. [S.l.]: Sun Microsystems, 1997.

[MCC93] MCCOOL, Rob. **Common Gateway Interface Overview**. Illinois University: National Center for Supercomputing Applications, 1993. Disponível por WWW em http://hoohoo.ncsa.uiuc.edu/cgi/overview.html.

[MCG97] MCGRAW, G.; FELTEN, E.W. **Java Security**. [S.l.]: Wiley Computer Publishing, 1997. 194 p.

[MIC96] MICROSOFT CORPORATION. **Microsoft Visual J++**. 1996.

[NCS9?a] NCSA - National Center for Supercomputing Applications. **A Beginner's Guide to HTML**. Illinois University. Disponível por WWW em http://www.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimer.html (1996).

[NCS9?b] NCSA - National Center for Supercomputing Applications. **NCSA Fill-out Forms**. Illinois University: NCSA HTTPD Development Team. Disponível por WWW em http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/fill_out_forms/overview.html (1996).

[NCS9?c] NCSA - National Center for Supercomputing Applications**. NCSA HTTP Server**. Illinois University: NCSA HTTPd Development Team Disponível por WWW em http://www.ncsa.uiuc.edu/General/Internet/WWW/ (1997).

[NCS95] NCSA - National Center for Supercomputing Applications. **The Common Gateway Interface**. Illinois University: NCSA HTTPd Development Team, 1995. Disponível por WWW em http://hoohoo.ncsa.uiuc.edu/cgi/.

[NIE90] NIELSEN, J. **Hypertext & Hypermedia**. Boston: Academic Press, Inc, 1990. p.268.

[RAG9?] RAGGETT, D. **HTML Specification**, Disponível por WWW em http://www-uk.hpl.hp.co.uk/people/dsr/ (1997).

[REA94] REALE, G.P.V. Transferência de Arquivos. In: CARVALHO, T.C.H. de B. (Org.). **Arquiteturas de Redes de Computadores OSI e TCP/IP**. São Paulo: Makron Books, 1994. p. 397-471.

[ROB95] ROBINSON, D. et al. **Apache: an HTTP Server - Reference Manual**. Apache Group, 1995. Disponível por WWW em http://www.apache.org.

[SAN97] SANKAR, S; VISWANADHA, S. **The Java Compiler Compiler Documentation**. Sun Microsystems, 1997. Disponível por WWW em http://www.suntest.com/JavaCC/.

[SHI97] SHIMMIN, B.F. Java jolts server performance. **LAN Times**, Aug., 1997.

[SYB97] SYBASE, INC. **PowerJ Enterprise**. Powersoft, 1997.

[TAN96] TANENBAUM, A.S. **Computer Networks**. 3. ed. [S.l.]: Prentice Hall, 1996.

[TAR95] TAROUCO, L.M.R.; OTSUKA, J.L. **Repositório de Educação à Distância**. Universidade Federal do Rio Grande do Sul, 1995. Disponível por WWW em http://www.penta.ufrgs.br/edu/home_edu.html (1996).

[VOS97] VOSS, G. **JavaServer Technologies**. JavaSoft, Java Developer Connection, 1997. Disponível por WWW em http://jdc.javasoft.com/.

[WAY96] WAYNER, P. Inside the Network Computer. **Byte**, New York, p. 105-110, Nov. 1996.

[WOD93] WODASKI, R. **Multimídia**. [S.l.]: Ciência Moderna, 1993.

[WOL97] WOLLRATH, A.; WALDO, J.; RIGGS, R**.** Java-Centric Distributed Computing. **IEEE Micro**, Los Alamitos, p. 44-53, May/Jun. 1997.

[WUT97] WUTKA, M. **Java:** Técnicas Profissionais. [S.l.]: Berkeley, 1997.

[ZUK97] ZUKOWSKI, J. **Java AWT Reference**. [S.l.]: O'Reilly & Associates, 1997.