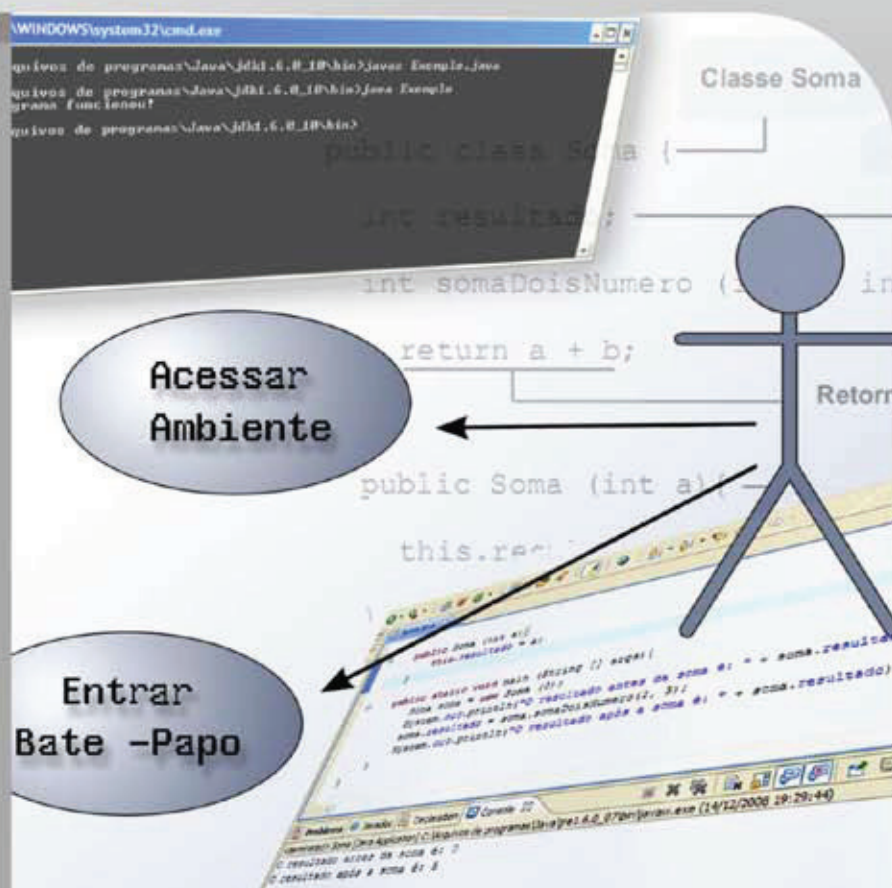


Laboratório de Programação

VERSÃO DE DEMONSTRAÇÃO

Richarlyson A. D'Emery

VERSÃO DE DEMONSTRAÇÃO



VERSÃO DE DEMONSTRAÇÃO

FASCÍCULO 1



Universidade Federal Rural de Pernambuco

Reitor: Prof. Valmar Corrêa de Andrade

Vice-Reitor: Prof. Reginaldo Barros

Pró-Reitor de Administração: Prof. Francisco Fernando Ramos Carvalho

Pró-Reitor de Extensão: Prof. Paulo Donizeti Siepierski

Pró-Reitor de Pesquisa e Pós-Graduação: Prof. Fernando José Freire

Pró-Reitor de Planejamento: Prof. Rinaldo Luiz Caraciolo Ferreira

Pró-Reitora de Ensino de Graduação: Prof^a. Maria José de Sena

Coordenação de Ensino a Distância: Prof^a Marizete Silva Santos

VERSÃO DE DEMONSTRAÇÃO

Produção Gráfica e Editorial

Capa e Editoração: Italo Amorim, Allyson Vila Nova e Rafael Lira

Revisão Ortográfica: Marcelo Melo

Ilustrações: Diego Almeida

Coordenação de Produção: Marizete Silva Santos

Capítulo 1 – O que são ambientes de programação?

VERSÃO DE DEMONSTRAÇÃO

1.1 Objetivos

Este capítulo tem como objetivo capacitar o estudante estimulando a utilização de ambientes de programação para o desenvolvimento de programas computacionais, destacando os aspectos envolvidos no desenvolvimento de aplicações e soluções algorítmicas para problemas computacionais, apresentando as características envolvidas no processo de desenvolvimento, até então abstratos e de difícil compreensão. Além dos conceitos são sugeridas atividades práticas que deverão ser realizadas para um melhor desempenho. Também são sugeridos links para práticas que estimulem a reflexão.

1.2 Introdução

Hoje é uma realidade que os novos profissionais de desenvolvimento devem estar capacitados para a utilização de tecnologias na produção de suas aplicações, seja em caráter educacional ou comercial, e tem como desafio tornar o desenvolvimento confiável, previsível e ágil, obedecendo todas as etapas no processo de desenvolvimento de software. Para realizar o desenvolvimento, o programador precisa utilizar algumas ferramentas para que o desenvolvimento das aplicações seja mais eficaz, otimizando seu trabalho e tempo de entrega.

VERSÃO DE DEMONSTRAÇÃO

Dessa forma, é muito comum o uso de um conjunto de funções já disponíveis no sistema, que tem por finalidade auxiliar a programação, reproduzindo as funções mais comuns naquele sistema.

Outro fator muito comum é uma aplicação utilizar várias funções do próprio sistema operacional a qual esteja rodando, ou mesmo outro programa ou sistema para auxiliar a execução da aplicação a ser programada. Ao conjunto de funções específicas e ao sistema operacional ou de apoio dá-se o nome de **ambiente de programação**. Na verdade o conceito é bem mais amplo, uma vez que deve englobar seu próprio sistema de apoio ao desenvolvimento.

Vale ressaltar que um programa não roda isoladamente, possuindo elementos tanto no desenvolvimento quanto na execução que caracterizam o ambiente onde o programa irá rodar. É verdade que tais ferramentas não mostram ao programador como ele deverá implementar suas soluções, mas sim organizar suas idéias, a exemplo de diagramas, e, quando possível, ajudá-lo com quais os recursos estão disponíveis, a exemplo de métodos disponíveis para um determinado contexto.

VERSÃO DE DEMONSTRAÇÃO

Dessa forma, as ferramentas que utilizaremos como ambientes de programação são mais conhecidas como **ambientes de desenvolvimento integrado**, comumente chamado de **IDE**, do inglês *Integrated Development Environment*. Um IDE é um programa de computador que reúne características e ferramentas de apoio que auxiliam ao desenvolvimento de software com o objetivo de agilizar processos. Geralmente os IDEs tendem satisfazer a técnicas de *desenvolvimento rápido de aplicativos* – RAD, do inglês *Rapid Application Development*, ou seja, que visa a maior produtividade dos desenvolvedores.

VERSÃO DE DEMONSTRAÇÃO

Você sabia?

¹ O termo **depurar** popularmente está relacionado ao ato de purificar, limpar, desembacar algo, como por exemplo, depurar a água das sujidades, depurar-se no banho retirando as impurezas do corpo, depurar o fiel livrando-o de todo o mal, entre outros. Na computação, depurar significa examinar o código de um programa, a fim de identificar, localizar e suprimir falhas ou erros. Alguns programadores também utilizam o termo “**debugar**”, que nada mais é do que testar o software desde a parte lógica, quanto ao negócio e também à sua utilização.

Podemos dizer que um IDE pode tornar um trabalho muito mais simples, sobretudo se o desenvolvimento já vai manejando um número razoável de programas, a exemplo de classes na linguagem de programação Java – uma linguagem que obedece aos conceitos de *orientação a objetos* (OO), ou a arquivos de cabeçalho na linguagem C, mas conhecidos pela extensão **.h**.

Ademais estes ambientes nos permitem muito mais versatilidade para **depurar**¹ nossos programas visto que têm *debuggers* muito mais avançados, coisa que nos favorece para ir começando neste mundinho.

Podemos dizer que as principais características e ferramentas encontradas nos IDEs são:

- **Editor** – uma área de edição que substitui um editor de texto qualquer, ou seja, é uma área que permite editar o código fonte do programa escrito na(s) linguagem(ns) suportada(s) pelo IDE;
- **Compilador** – compila o código fonte do programa, editado em uma linguagem específica e a transforma em linguagem de máquina;

VERSÃO DE DEMONSTRAÇÃO

- **Linker** – utiliza o conceito de *link*, ou seja, liga (linka) os vários “pedaços” de código fonte, compilados em linguagem de máquina, em um programa executável que pode ser executado em um computador ou outro dispositivo computacional.
- **Depurador** (*debugger*) – auxilia no processo de encontrar e corrigir erros (*bugs*) no código fonte do programa, na tentativa de aprimorar a qualidade de software;
- **Modelagem** – criação do modelo. Se o IDE suporta a utilização de linguagens orientada a objetos, permite a criação de classes, objetos, interfaces, associações e interações dos artefatos envolvidos no software com o objetivo de solucionar as necessidades-alvo do software final.
- **Geração de código** – os IDEs permitem gerar códigos, sendo essa a característica mais explorada em ferramentas CASE (*Computer-Aided Software Engineering*), contudo com um escopo mais direcionado a *templates* de código comumente utilizados para solucionar problemas rotineiros. Todavia, em conjunto com ferramentas de modelagem, a geração pode originar todo ou praticamente todo o código fonte do programa com base no modelo proposto, tornando muito mais rápido o processo de desenvolvimento e distribuição do software;
- **Distribuição** – auxilia no processo de criação do instalador do software, ou outra forma de distribuição do mesmo, seja em discos ou via internet. **VERSÃO DE DEMONSTRAÇÃO**
- **Testes automatizados** – realiza testes no software de forma automatizada, com base em scripts ou programas de testes previamente especificados, gerando um relatório dos mesmos, assim auxiliando na análise do impacto das alterações no código fonte. **VERSÃO DE DEMONSTRAÇÃO**
- **Refatoração** – o termo relaciona-se quanto na melhoria constante do código fonte do software, seja na construção de código mais otimizado, mais limpo e/ou com melhor entendimento pelos envolvidos no desenvolvimento do software. A refatoração, em conjunto com os testes automatizados, é uma poderosa ferramenta no processo de erradicação de *bugs*, tendo em vista que os testes garantem o mesmo comportamento externo do software ou da característica sendo reconstruída.

No próximo capítulo será apresentado o IDE Eclipse que será utilizado no desenvolvimento de programas e um breve passeio sobre os conceitos envolvidos na programação orientada a objetos.

VERSÃO DE DEMONSTRAÇÃO

1.3 Apresentando o IDE Eclipse e o J2SE

O Eclipse é um IDE de código aberto para a construção de programas de computador que podem ser desenvolvidos nas linguagens como Python, C, C++, além de Java, claro. Atualmente, pode-se dizer que Eclipse é uma comunidade, cujos projetos estão focados sob uma plataforma de desenvolvimento composta de frameworks, ferramentas e um compilador para construção, implantação e gestão de um software em todo seu ciclo de vida. O projeto Eclipse foi iniciado na empresa IBM (*International Business Machines Corporation*) em parceria com consórcios de desenvolvedores que em novembro de 2001 desenvolveu a primeira versão do produto e presenteou o mundo com um software livre. Em janeiro de 2004, passou a ser gerenciado pela Eclipse Foundation, uma entidade sem fins lucrativos mantida por colaborações de membros associados, fornece quatro serviços a comunidade: *IT Infrastructure, Intellectual Property Management, Development Process, and Ecosystem Development*. Hoje podemos dizer que o Eclipse é o IDE Java mais utilizada no mundo. Possui como características marcantes o uso da SWT (*Standard Widget Toolkit*) como biblioteca gráfica, a forte orientação ao desenvolvimento baseado em *plug-ins* e o amplo suporte ao desenvolvedor com centenas de *plug-ins* que procuram atender a diferentes programadores as mais diferentes necessidades [Eclipse].

Antes de iniciarmos a instalação do IDE Eclipse, precisamos verificar se o J2SE (*Java 2 Standard Edition*) está instalado. Mas o que é o J2SE?

VERSÃO DE DEMONSTRAÇÃO

O **J2SE** é uma ferramenta de desenvolvimento Java que contém todo o ambiente necessário para a criação e execução de aplicações Java, incluindo principalmente a máquina virtual Java (JVM), o compilador Java e as APIs (*Application Programming Interface*) do Java.

A JVM é como se chama o “*emulador*” do Java para os principais sistemas (Windows, UNIX/Linux, Mac OS X) e, provavelmente, você já deve ter usado o Java mesmo sem perceber, pois existem inúmeros sites na Internet que utilizam *Applets* desenvolvidos em

Java. E estes nada mais são do que programas Java rodando dentro de um browser (Internet Explorer, Netscape, Firefox, etc.). Ou seja, a JVM é um programa que carrega e executa os aplicativos Java, convertendo os **bytecodes**² em código executável de máquina além de ser responsável pelo gerenciamento dos aplicativos, à medida que são executados.

VERSÃO DE DEMONSTRAÇÃO

O J2SE é dividido em duas partes, sendo a primeira o **JRE** (do inglês *Java Runtime Edition*, ou simplesmente **Ambiente de Tempo de Execução Java**) necessária para rodar programas implementados em Java, ou seja, é indicado apenas para executar as aplicações Java, sendo que o pacote contém somente as bibliotecas (APIs) necessárias para tal e a JVM. Dessa forma, não se consegue compilar um código Java apenas com um bloco de notas por exemplo, sendo necessário a utilização de um IDE, a exemplo do Eclipse que já possui um compilador. A segunda parte é constituída do **JDK** (do inglês *Java Developer Kit*, ou seja, **Kit de Desenvolvimento Java**) que serve para criar os programas, e é composto pelo compilador, *appletviewer* para executar *applets* e bibliotecas.

Geralmente, o JRE e JDK são instalados no diretório padrão **C:\Arquivos de programas\Java**. Se não encontrar e ainda assim desejar ter certeza se está instalado, basta acessar uma página web que contenha *applets* Java, a exemplo do site do Banco do Brasil, onde o teclado virtual para a inserção da senha necessita do JRE.

Os pacotes do J2SE (JDK e JRE) podem ser baixados no site da Sun Microsystems: <http://java.sun.com/javase/downloads/index.jsp>.

VERSÃO DE DEMONSTRAÇÃO



jdk-6u10-windows-i586-p.exe



jre-6u11-windows-i586-p.exe

A instalação dos pacotes é bem simples, uma vez baixados, basta executar os instaladores e seguir suas instruções.

Como o JRE será fundamental para o desenvolvimento dos programas em Java, acesse a página do Java.com (http://java.com/pt_BR/download/help/testvm.xml) e verifique se a imagem com animação do logotipo do Duke dançando aparecerá na tela, isso significa que o JRE foi instalado e configurado corretamente.

Saiba Mais!

² O termo **bytecode** na computação diz respeito a um conjunto de bytes que contém instruções, sendo o resultado de um processo semelhante ao dos compiladores de código fonte, mas que não é executável diretamente, sendo necessária uma máquina virtual para interpretá-lo e realizar sua execução. Logo suas informações servem para qualquer arquitetura, por isso podemos dizer que é *portável*. Em Java, o bytecode será executado pela JVM. Saiba mais sobre bytecode Java acessando o link http://pt.wikipedia.org/wiki/Bytecode_Java



Duke

1.3.1 Rodando uma aplicação Java

Antes de utilizarmos um ambiente de desenvolvimento (IDE) como o Eclipse, iremos escrever e compilar um programa Java fazendo uso do JDK e JRE com o auxílio de um editor de texto qualquer, como por exemplo o Bloco de notas.

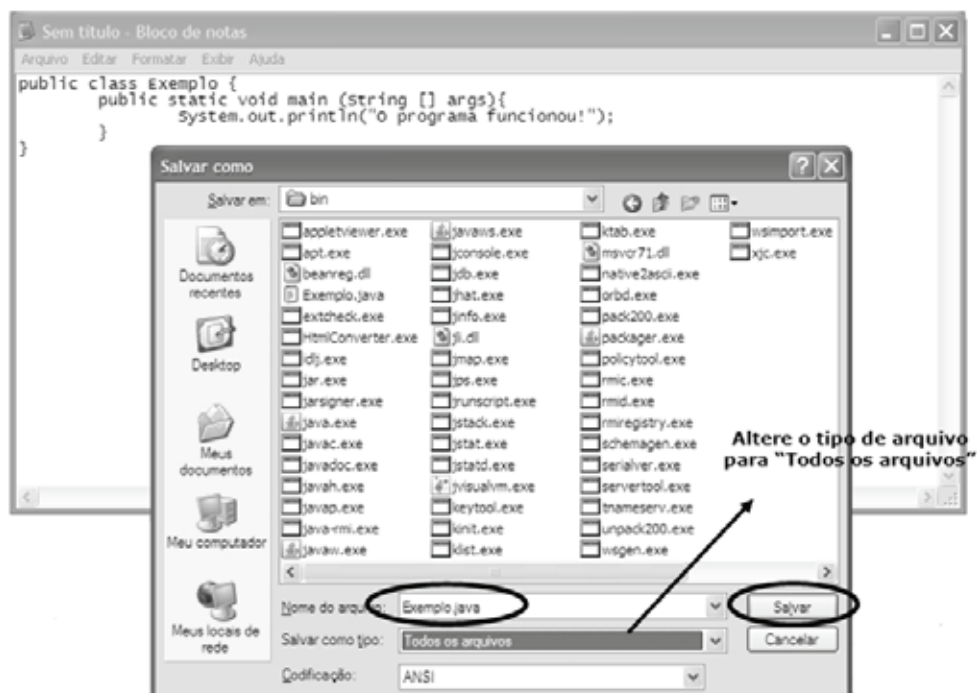
Vamos utilizar o seguinte programa (**código fonte**) em Java:

```
public class Exemplo {
    public static void main (String [] args){
        System.out.println("O programa funcionou!");
    }
}
```

VERSÃO DE DEMONSTRAÇÃO


Após digitado o código fonte acima no Bloco de notas, salvaremos o arquivo com a extensão **.java**, que indica que o arquivo contém código fonte Java. Para o nosso exemplo, salvaremos **Exemplo.java** no diretório **C:\Arquivos de programas\Java\jdk1.6.0_10\bin**, sendo este o local onde o JDK foi instalado. Vale ressaltar que devemos diferenciando as letras maiúsculas das minúsculas, pois a linguagem Java é **Case Sensitive**, ou seja, as letras maiúsculas diferem das minúsculas no programa. Outra característica é que o nome da classe é exatamente o mesmo nome do arquivo.

VERSÃO DE DEMONSTRAÇÃO



Após salvo o programa, fechamos o editor de texto e abrimos o *prompt* de comandos DOS. No diretório **C:\Arquivos de programas\Java\jdk1.6.0_10\bin** digitamos

o comando `javac Exemplo.java` e confirmamos (Enter).



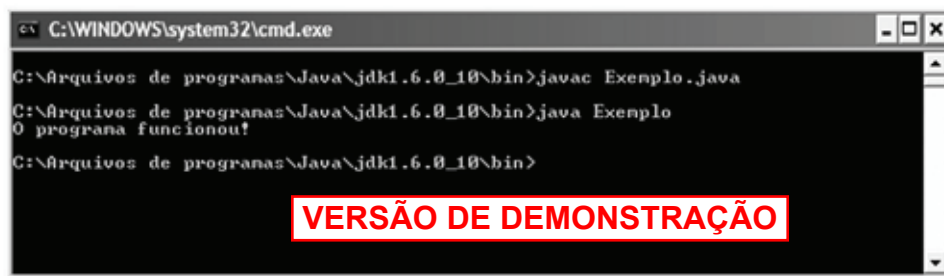
```
C:\WINDOWS\system32\cmd.exe
C:\Arquivos de programas\Java\jdk1.6.0_10\bin>javac Exemplo.java_
```

Saiba Mais!

Saiba mais sobre os vários pacotes que a tecnologia Java possui, é uma verdadeira "sopa de letrinha com sabor Java". Acesse o link <http://www.guj.com.br/java.artigo.105.1.guj> e conheça um pouco mais.

Depois de compilado o arquivo `Exemplo.class` é gerado no diretório. Observe que o arquivo só foi compilado pelo fato do JDK ter sido instalado. O comando `javac` (compilador Java) foi utilizado para compilar o programa, onde o compilador Java converte o código fonte do programa em bytecodes que representam as tarefas a serem realizadas durante a fase de execução. Os bytecodes são executados pela JVM.

Para executar o arquivo `Exemplo.class` utilizaremos o comando `java Exemplo` (não sendo necessário informar a extensão `.class`), vejamos:



```
C:\WINDOWS\system32\cmd.exe
C:\Arquivos de programas\Java\jdk1.6.0_10\bin>javac Exemplo.java
C:\Arquivos de programas\Java\jdk1.6.0_10\bin>java Exemplo
O programa funcionou!
C:\Arquivos de programas\Java\jdk1.6.0_10\bin>
```

O comando `java` invoca a JVM que inicia os passos necessários para executar o aplicativo. Após a execução do comando podemos observar que a frase "O programa funcionou!" foi exibida no prompt isso porque foi executada a instrução `System.out.println()` que tem a mesma função do `println()` da linguagem de programação C.

VERSÃO DE DEMONSTRAÇÃO

Dessa forma podemos concluir que uma aplicação Java pode ser simplesmente construída a partir da ferramenta de desenvolvimento Java **J2SE**, destacando as principais funções do JDK e do JRE.



Atividades de Estudo

VERSÃO DE DEMONSTRAÇÃO

Após apresentados os primeiros passos para iniciar a programação Java e algumas principais características envolvidas, exercite respondendo as seguintes questões:

1. O que são ambientes de programação?
2. Quais as principais características encontradas em um IDE?
3. O que significa a expressão “*depurar um programa*”?
4. De que é constituído o J2SE?
5. Qual a diferença entre JRE e JDK?
6. Qual a função da JVM?
7. Implemente o seguinte código fonte e informe qual o seu resultado:

```
public class Questao7 {
    public static void main (String args[]) {
        int i;
        for (i=0; i<=10; i++) {
            System.out.print (i + "\n");
            i++;
        }
    }
}
```

VERSÃO DE DEMONSTRAÇÃO

8. Implemente o seguinte código fonte e informe qual o seu resultado:

```
public class Questao8 {
    public static void main (String args[]) {
        int linha = 10, coluna;
        while (linha >= 1) {
            coluna = 1;
            while (coluna <= 10) {
                System.out.print (linha % 2 == 1 ? "<" : ">");
                ++coluna;
            }
            --linha;
            System.out.println();
        }
    }
}
```

VERSÃO DE DEMONSTRAÇÃO