

Java Starter

www.t2ti.com

Curso Java Starter

Apresentação

O Curso Java Starter foi projetado com o objetivo de ajudar àquelas pessoas que têm uma base de lógica de programação e desejam entrar no mercado de trabalho sabendo Java,

A estrutura do curso é formada por módulos em PDF e por mini-cursos em vídeo. O aluno deve baixar esse material e estudá-lo. Deve realizar os exercícios propostos. Todas as dúvidas devem ser enviadas para a lista de discussão que está disponível para inscrição na página do Curso Java Starter no site www.t2ti.com. As dúvidas serão respondidas pelos instrutores Albert Eije, Cláudio de Barros e Miguel Kojjio, além dos demais participantes da lista.

Nosso objetivo é que após o estudo do Curso Java Starter o aluno não tenha dificuldades para acompanhar um curso avançado onde poderá aprender a desenvolver aplicativos para Web, utilizando tecnologias como Servlets e JSP e frameworks como Struts e JSF, além do desenvolvimento para dispositivos móveis.

Albert Eije trabalha com informática desde 1993. Durante esse período já trabalhou com várias linguagens de programação: Clipper, PHP, Delphi, C, Java, etc. Atualmente mantém o site www.alberteije.com.

Cláudio de Barros é Tecnólogo em Processamento de Dados.

Miguel Kojjio é bacharel em Sistemas de Informação, profissional certificado Java (SCJP 1.5).

O curso Java Starter surgiu da idéia dos três amigos que trabalham juntos em uma instituição financeira de grande porte.

Módulo

01

Introdução ao Java

Histórico

A linguagem de programação Java foi criada em 1991 por James Gosling, ela iniciou-se como parte do projeto Green da Sun Microsystems. Inicialmente a linguagem iria chamar-se Oak (Carvalho) em referência a árvore que era visível pela janela de James Gosling.

A mudança de nome ocorreu pois já existia uma linguagem de programação com este nome, então a linguagem foi rebatizada para Java.

O termo Java é utilizado, geralmente, quando nos referimos a:

- Linguagem de programação orientada a objetos;
- Ambiente de desenvolvimento composto pelo compilador, interpretador, gerador de documentação e etc.;
- Ambiente de execução que pode ser praticamente qualquer máquina que possua **Java Runtime Environment** (JRE) instalado;

A linguagem de programação Java é uma linguagem de alto-nível com as seguintes características:

- **Simples:** O aprendizado da linguagem de programação Java pode ser feito em um curto período de tempo;
- **Orientada a objetos:** Desde o início do seu desenvolvimento esta linguagem foi projetada para ser orientada a objetos;
- **Familiar:** A linguagem Java é muito familiar para os programadores C/C++ ;
- **Robusta:** Ela foi pensada para o desenvolvimento de softwares confiáveis,

Curso Java Starter

provendo verificações tanto em tempo de execução quanto compilação, o coletor de lixo responsabiliza-se pela limpeza da memória quando houver necessidade;

- **Segura:** Aplicações Java são executadas em ambiente próprio (JRE) o que inviabiliza a intrusão de código malicioso;
- **Portável:** Programas desenvolvidos nesta linguagem podem ser executados em praticamente qualquer máquina desde que esta possua o JRE instalado;
- etc.

Máquina Virtual Java - JVM

A máquina virtual java (JVM) é uma máquina imaginária que emula uma aplicação em uma máquina real. É a JVM que permite a portabilidade do código Java, isto ocorre porque todo código Java é compilada para um formato intermediário, bytecode, este formato é então interpretado pela JVM.

Existem diversas JVMs cada uma delas destinada a um tipo de sistema operacional (Windows, Linux, Mac e etc.), desta forma sendo o código da aplicação Java, bytecode, um código interpretado pela JVM, podemos desenvolver uma aplicação sem nos preocuparmos onde ela será executada pois sabemos que existindo a JVM instalada nosso código será executável.

Coletor de Lixo – Garbage Collection

Muitas linguagens de programação nos permitem alocar espaço na memória em tempo de execução, uma vez encerrado o programa deve haver uma maneira de liberar este espaço para que outras aplicações possam utilizá-lo.

Em muitas das linguagens de programação, inclusive C e C++, a responsabilidade pela liberação do espaço que não mais será utilizado é do programador, no entanto, nem sempre é fácil gerenciar o que está e o que não está sendo utilizado, a má gerência da memória ocasiona muitas vezes o estouro de pilha (stack overflow) entre outros problemas.

Na linguagem de programação Java a responsabilidade pela gerência da

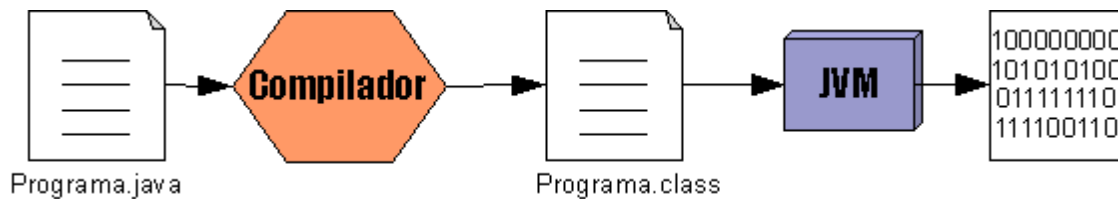
Curso Java Starter

memória é do Coletor de lixo (Garbage Collector), desta forma, programadores Java ficam livres da preocupação de alocação e desalocação da memória.

O Coletor de lixo é um processo que roda em segundo plano e é responsável pela liberação de memória alocada por variáveis que não mais serão utilizadas pela aplicação.

Fases de um programa Java

As fases pelo qual passam um programa Java relacionam-se da seguinte forma:



1. Criação do código fonte (Programa.java);
2. Compilação do código fonte e geração do bytecode (Programa.class);
3. Interpretação do bytecode pela máquina virtual;
4. Conversão do bytecode em linguagem de máquina.

Hotspot

Hotspot é a máquina virtual Java, ela provê algumas funcionalidades muito importantes. Ao contrário de outras aplicações, que são compiladas diretamente para código da máquina em que serão executadas, em Java estas somente são transformadas em código de máquina em tempo de execução quando necessário.

No princípio esta abordagem trouxe problemas de sobrecarga e lentidão dos sistemas, no entanto, a JVM vem se aprimorando e, em muitas situações, as aplicações Java tem desempenho similar as aplicações que são previamente compiladas.

Este desempenho vem melhorando muito devido a otimização que a máquina virtual consegue fazer a medida que o código é executado. Perceba que quando programamos em C, por exemplo, o código fonte é transformado em código de máquina imediatamente.

Curso Java Starter

Em princípio podemos pensar que o fato do programa não precisar passar por uma etapa a mais, interpretação, irá torná-lo mais eficiente, mas muitas vezes a compilação estática não consegue prever situações que irão ocorrer durante a execução do código: trechos da aplicação mais utilizados, carga do sistema, quantidade de usuários simultâneos, memória disponível e etc..

Estas informações, relativas ao ambiente no qual a aplicação está sendo executada, são utilizadas pela JVM para fazer otimizações em tempo de execução e havendo necessidade o código que está sendo interpretado é transformado em instruções nativas do sistema operacional (código de máquina) em um processo de compilação dinâmica.

Esta transformação em tempo de execução é realizada pelo JIT, Just-in-time compiler. O fato do código (bytecode) ser transformado, em tempo de execução, em código de máquina permite que a JVM mude a estratégia de compilação em busca de um melhor desempenho, em um ciclo de "aprendizado" contínuo.

JRE e JDK

- **JRE:** O Java Runtime Environment contém tudo aquilo que um usuário comum precisa para executar uma aplicação Java (JVM e bibliotecas), como o próprio nome diz é o "Ambiente de execução Java";
- **JDK:** O Java Development Kit é composto pelo JRE e um conjunto de ferramentas úteis ao desenvolvedor Java.

Versões do Java:

Abaixo temos uma síntese das versões do Java e as principais alterações nas nomenclaturas e no seu conteúdo.

- 1. JDK 1.0 (1996):** Primeira versão;
- 2. JDK 1.1 (1997):** Adição das bibliotecas JDBC, RMI e etc;
- 3. J2SE 1.2 (1998) – Playground:** A partir daqui todas as versões Java foram denominadas de Java 2 Standard Edition, passaram a ter apelidos (Playground) e foi adicionado o Framework Collections e etc.;
- 4. J2SE 1.3 (2000) – Kestrel:** Inclusão das bibliotecas JNDI, JavaSound e etc.;

Curso Java Starter

5. **J2SE 1.4 (2002) – Merlin:** Palavra reservada "assert", biblioteca NIO e etc.;
6. **J2SE 5.0 (2004) – Tiger:** Apesar da versão ser 1.5, agora é chamada apenas de 5. Adições importantes como: Enumeração, Autoboxing, Generics, for-each e etc;
7. **JSE 6 (2006) – Mustang:** Entre outras alterações houveram mudança na nomenclatura (remoção do 2 – J2SE) e melhora significativa na performance.

Ferramentas do JDK

A seguir temos uma breve descrição das principais ferramentas que fazem parte do JDK:

- **javac:** Compilador da linguagem Java;
- **java:** Interpretador Java;
- **jdb:** Debugador Java;
- **java -prof:** Interpretador com opção para gerar estatísticas sobre o uso dos métodos;
- **avadoc:** Gerador de documentação;
- **jar:** Ferramenta que comprime, lista e expande;
- **appletviewer:** Permite a execução e debug de applets sem browser;
- **javap:** Permite ler a interface pública das classes;
- **extcheck:** Detecta conflitos em arquivos Jar.

Primeiro contato com o Java - Definições

- **Classe:** É a estrutura que, quando construída, produzirá um objeto, dizemos "**todo objeto é instância de alguma classe**";
- **Objeto:** Em tempo de execução, quando a JVM encontra a palavra reservada **new** é criada uma instância da classe apropriada;
- **Estado:** É definido pelo conjunto de atributos de uma classe, isto é, cada instância da classe possuirá um estado independente dos demais objetos.
- **Comportamento:** São os métodos da classe, comportamento é aquilo que uma classe faz (algoritmos), muitas vezes, um determinado comportamento (método) muda o estado do objeto, isto é, após a execução do método um ou mais atributos mudaram de valor;

Primeiro contato com o Java - Nomenclatura

Existem três aspectos importantes, em relação a nomenclatura, que devemos considerar quando estamos programando Java.

- **Identificadores válidos:** Definem as regras para que o compilador identifique o nome como válido.
 - I. Devem iniciar com uma letra, cifrão (\$) ou sublinhado/underscore (_);
 - II. Após o primeiro caracter podem ter qualquer combinação de letras, caracteres e números;
 - III. Não possuem limite de tamanho;
 - IV. Não podem ser palavras reservadas;
 - V. Identificadores são case-sensitive isto é, "Nome" e "nome" são identificadores diferentes.

Exemplos:

Identificadores válidos	Identificadores inválidos
_codigo	5ident
\$turma	-idade
\$\$_5A	%valor

- **Convenção de nomenclatura da SUN:** São recomendações da SUN para nomenclatura de classes, métodos e variáveis. Seu programa irá funcionar mesmo que você não siga estas convenções.
 - I. Classes e interfaces: A primeira letra deve ser maiúscula e, caso o nome seja formado por mais de uma palavra, as demais palavras devem ter sua primeira letra maiúscula também (camelCase);
 - II. Métodos: A primeira letra deve ser minúscula e após devemos aplicar o camelCase;
 - III. Variáveis: Da mesma forma que métodos;
 - IV. Constantes: Todas as letras do nome devem ser maiúsculas e caso seja formada por mais de uma palavra separada por underscore.

Curso Java Starter

Exemplos:

Classes	Métodos	Variáveis	Constantes
Carro	desligar	motor	COMBUSTIVEL
CursoJavaIniciante	iniciarModulo	quantidadeModulos	NOME_CURSO
Hotel	reservarSuiteMaster	nomeReservaSuite	TAXA_SERVICO

- **Convenção JavaBeans:** Requisitos para que os nomes atendam a especificação para JavaBeans.
 - I. Se o atributo não for um booleano o método getter (utilizado para obter a variável) deve iniciar por "get";
 - I. Se o atributo for um booleano o método getter pode iniciar por "get" ou "is";
 - II. O método setter (atribuição) da propriedade sempre deve iniciar por "set";
 - III. O restante do nome deve ser o nome do atributo concatenado ao prefixo (is, get ou set) em letra maiúscula;
 - IV. O método setter e getter sempre devem ser públicos.

Exemplos:

Setters – atributo nome	Getters – atributo valor
public void setNome(argumento)	public boolean getValor()
	public boolean isValor()

Instalação do JDK

A seguir veremos os passos necessários para a instalação do JDK no Windows porém, antes de iniciarmos, vamos dar uma olhada nas variáveis de ambiente que devem ser atualizadas e criadas quando instalamos o JDK:

- **JAVA_HOME:** Indica o diretório onde foi instalado o JDK, muito utilizado por frameworks e por outros programas para localizar o JDK;

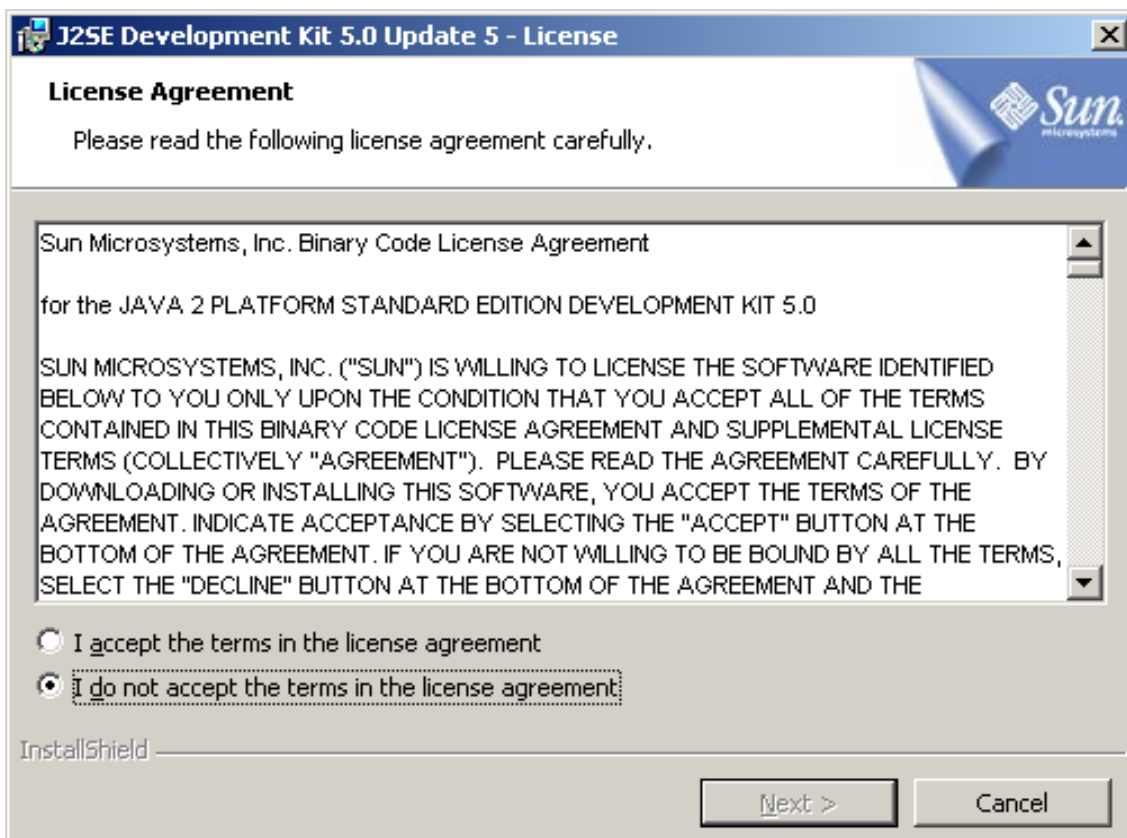
Curso Java Starter

- **PATH:** Identifica o local onde encontram-se as ferramentas de desenvolvimento (compilador, interpretador, gerador de documentação e etc.), devemos adicionar o diretório JAVA_HOME\bin;
- **CLASSPATH:** Identifica diretório onde o ClassLoader pode encontrar classes que são utilizadas pela sua aplicação.

Windows

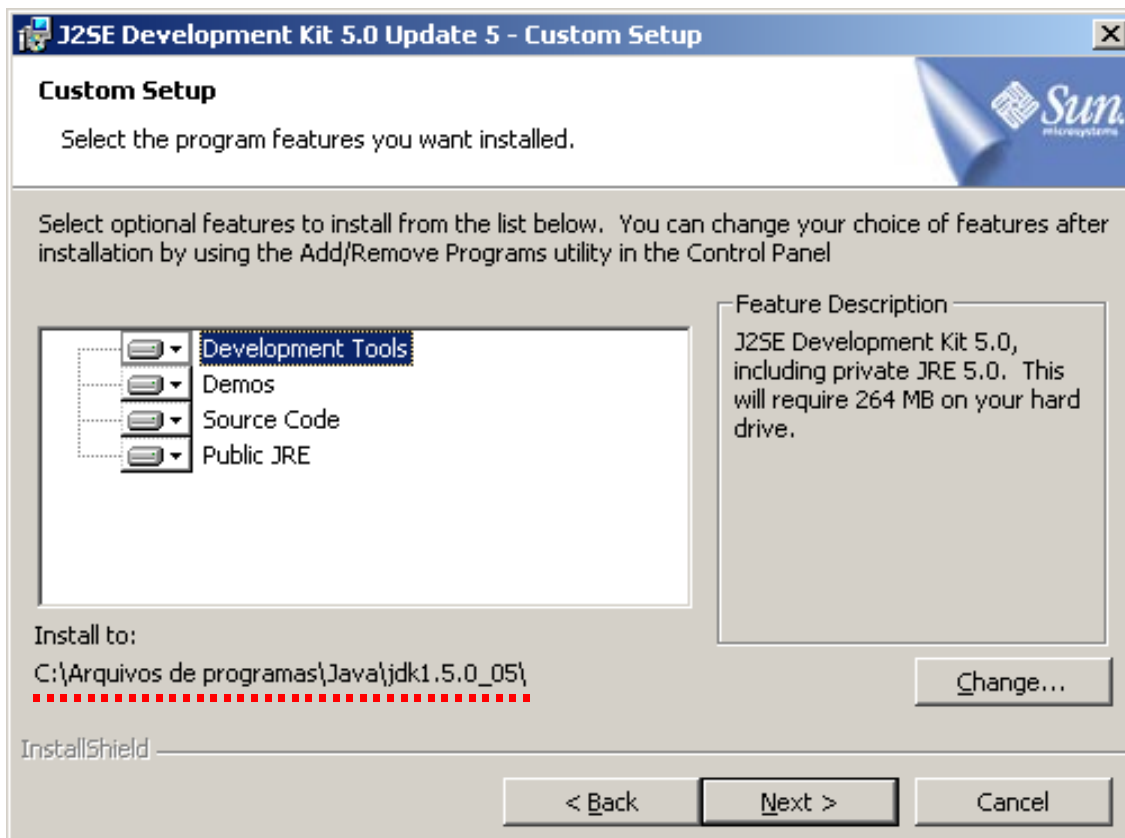
Faça o download do [JDK](#) no site da SUN e execute o instalador.

*Muitos estudantes têm tido dificuldade para escolher o instalador. O instalador correto sempre conterá o termo JDK no seu nome, se contiver o termo JRE é o incorreto.

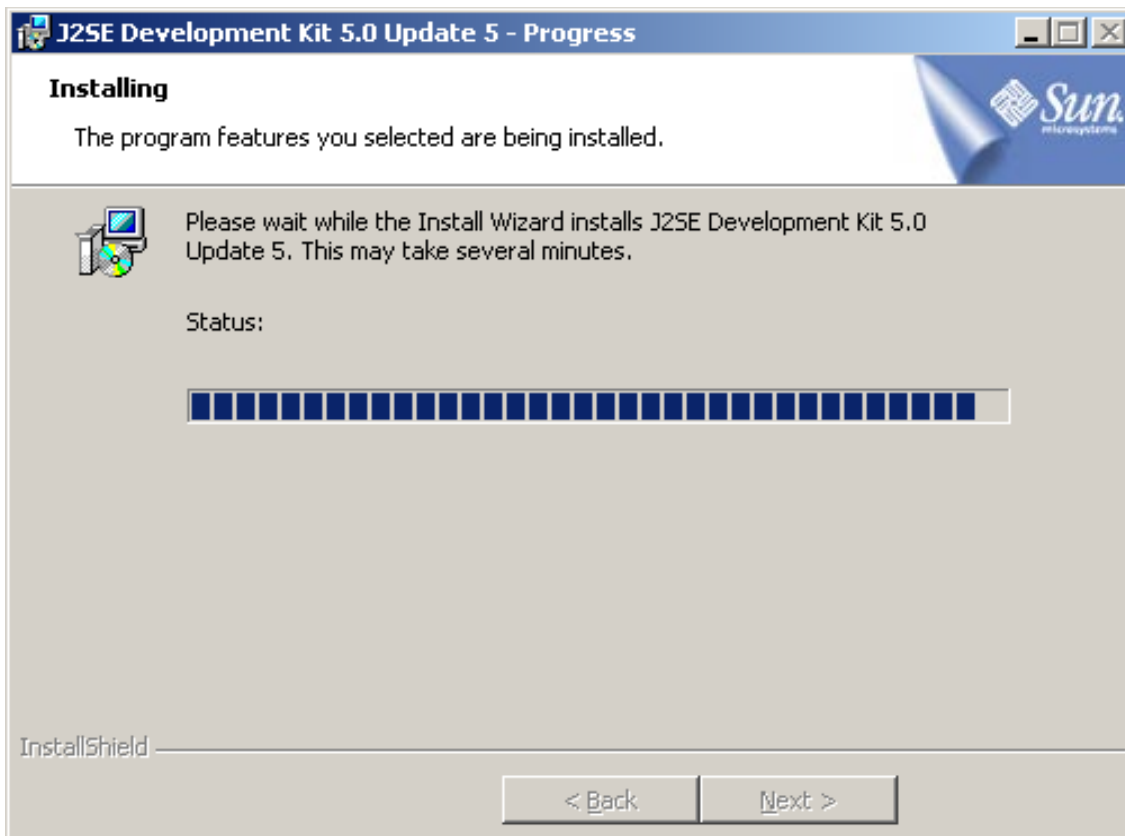


Curso Java Starter

Este diretório será utilizado na configuração do ambiente.

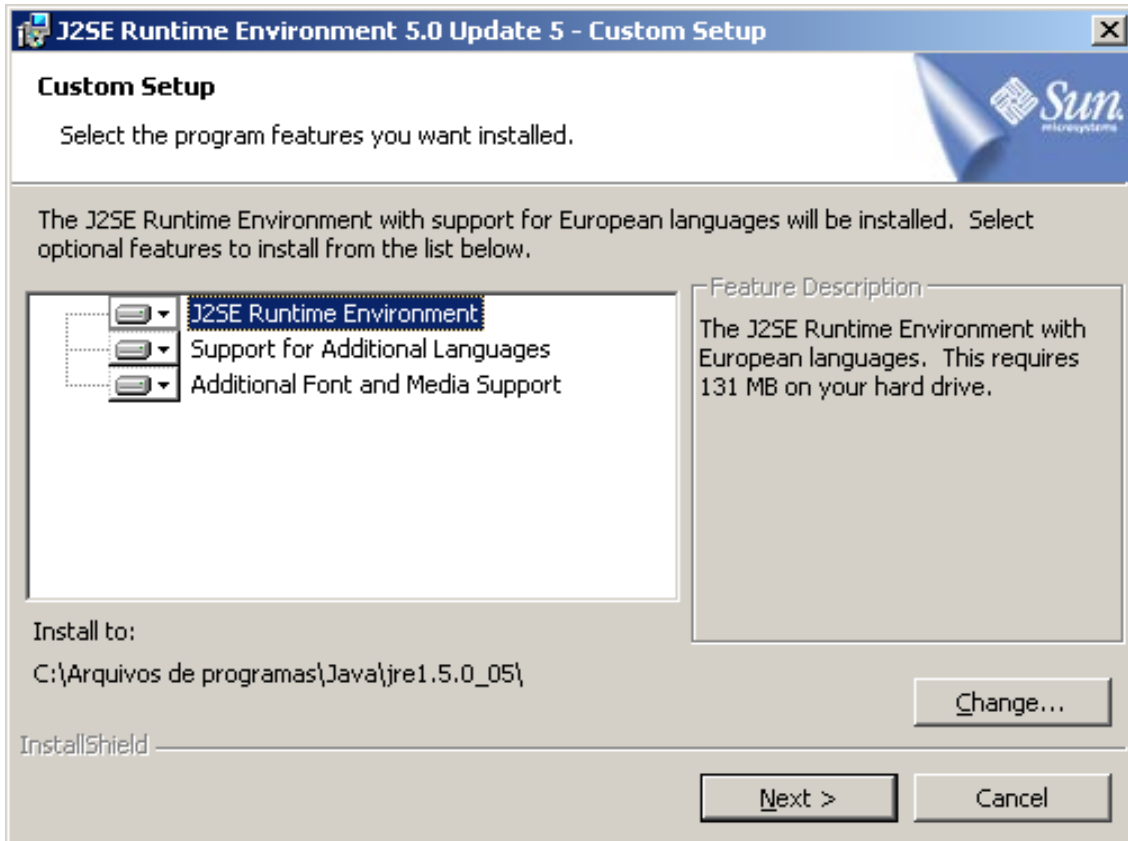


A seguir clique em **Next>** até que a instalação seja iniciada.

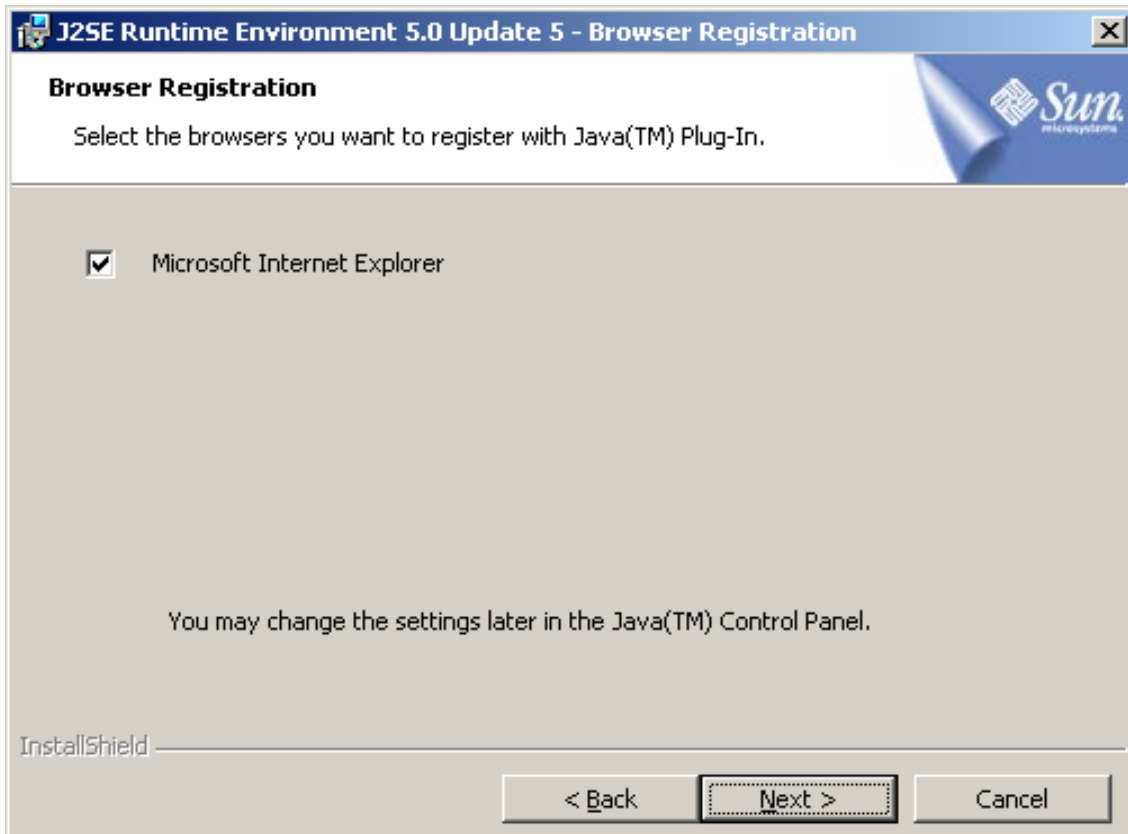


Curso Java Starter

Neste momento inicia-se a instalação do JRE, continue clicando em **Next>**.

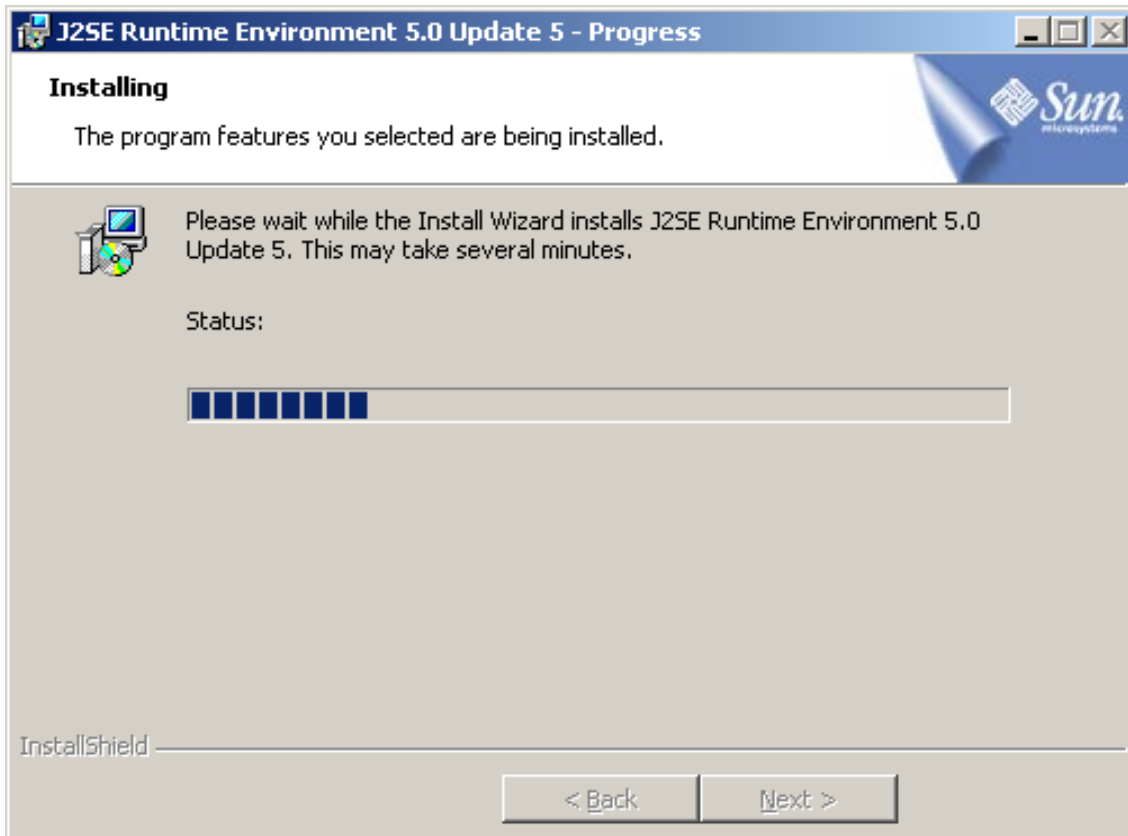


Nesta tela selecione todos os navegadores disponíveis.

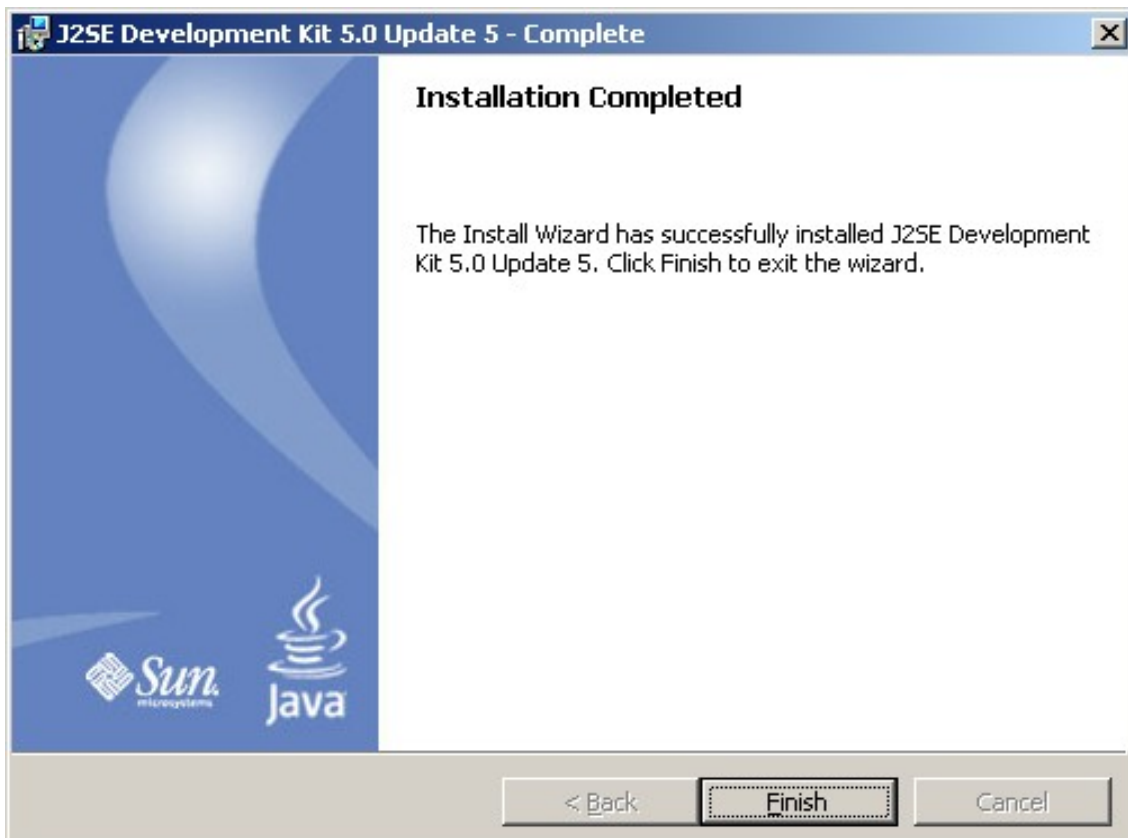


Curso Java Starter

Aguarde a conclusão da instalação.



Clique em **Finish** para terminar o processo.

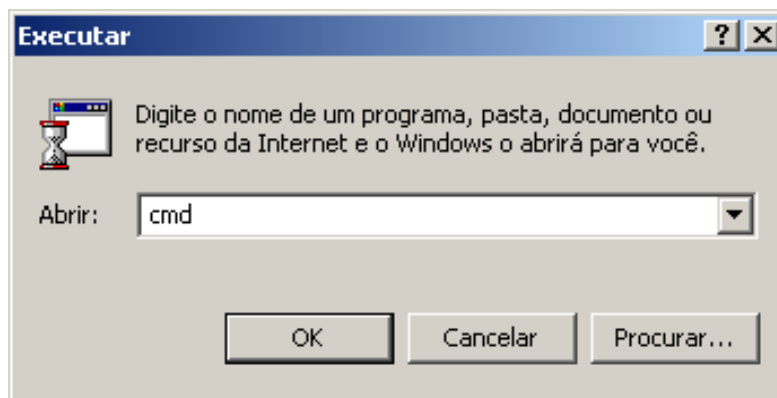


Curso Java Starter

Após a instalação do Java nós devemos iniciar a configuração das variáveis de ambiente.

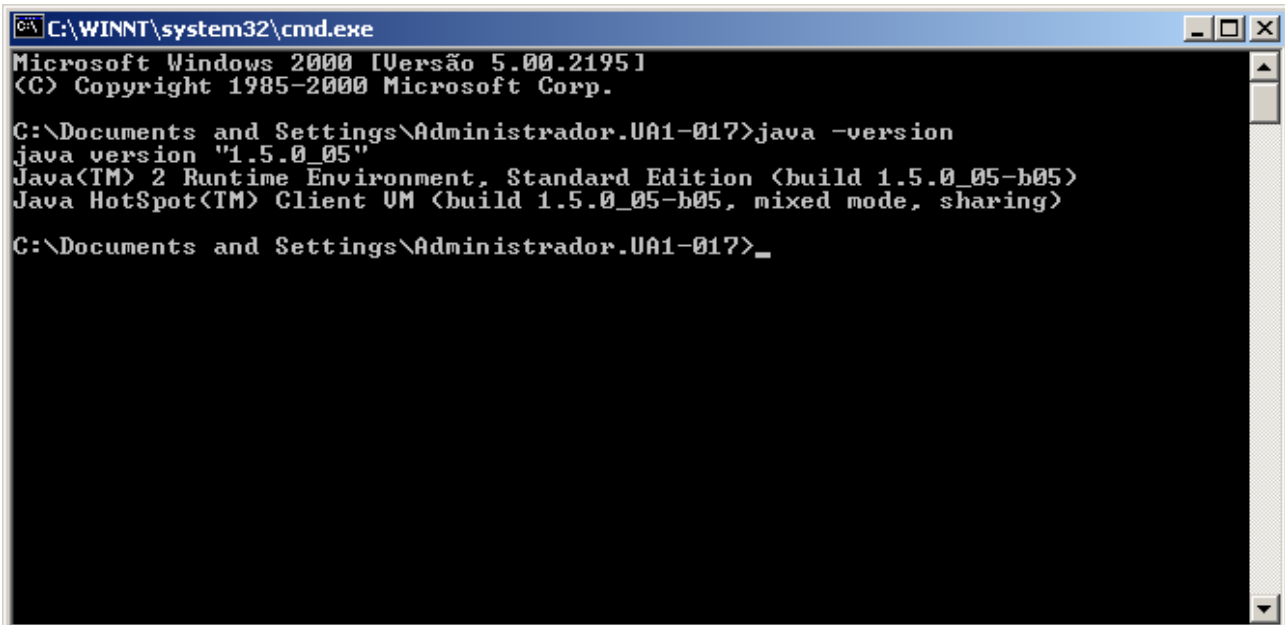
1. Clique com o botão direito em cima do ícone "Meu Computador";
2. Vá em "Propriedades";
3. Selecione a aba "Avançado";
4. Clique no botão "Variáveis de ambiente";
5. Clique no botão "Nova" em "Variáveis do sistema";
 - 5.1. Nome da variável: JAVA_HOME
 - 5.2. Valor da variável: Coloque aqui o endereço de instalação neste caso =
`C:\Arquivos de programas\Java\jdk1.5.0_05`
 - 5.3. Clique em **OK**
6. Clique novamente no botão "Nova" em "Variáveis do sistema";
 - 6.1. Nome da variável: CLASSPATH
 - 6.2. Valor da variável:
`.;%JAVA_HOME%\lib;%JAVA_HOME%\lib\tools.jar;%JAVA_HOME%\lib\dt.jar;%JAVA_HOME%\lib\htmlconverter.jar;%JAVA_HOME%\jre\lib;%JAVA_HOME%\jre\lib\rt.jar`
 - 6.3. Clique em **OK**
7. Selecione a variável PATH em "Variáveis do sistema";
 - 7.1. Adicione o seguinte endereço ao campo Valor da variável:
 - 7.2. `%JAVA_HOME%\bin`
 - 7.3. Clique em **OK**;
8. Clique em **OK**;
9. Clique em **OK**.

Agora vamos testar a instalação. Clique no botão **Iniciar**, vá em **Executar** e digite cmd.



Curso Java Starter

No prompt do MS-DOS vamos testar o interpretador, digite **java -version**, deverá aparecer algo parecido com isto:

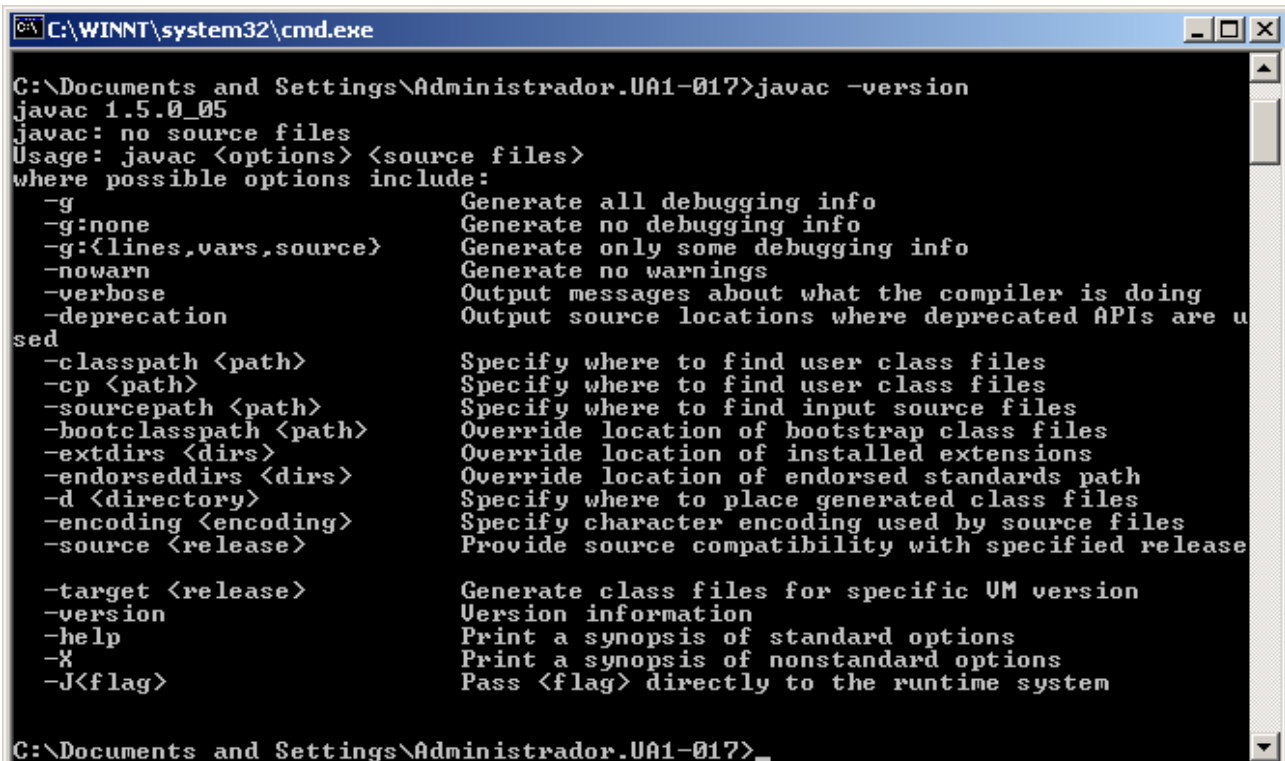


```
C:\WINNT\system32\cmd.exe
Microsoft Windows 2000 [Versão 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\Documents and Settings\Administrador.UA1-017>java -version
java version "1.5.0_05"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_05-b05)
Java HotSpot(TM) Client VM (build 1.5.0_05-b05, mixed mode, sharing)

C:\Documents and Settings\Administrador.UA1-017>_
```

Agora vamos testar o compilador, digite **javac -version**, deverá aparecer algo parecido com isto:



```
C:\WINNT\system32\cmd.exe

C:\Documents and Settings\Administrador.UA1-017>javac -version
javac 1.5.0_05
javac: no source files
Usage: javac <options> <source files>
where possible options include:
  -g                Generate all debugging info
  -g:none           Generate no debugging info
  -g:<lines,vars,source> Generate only some debugging info
  -nowarn           Generate no warnings
  -verbose          Output messages about what the compiler is doing
  -deprecation      Output source locations where deprecated APIs are used
sed
  -classpath <path> Specify where to find user class files
  -cp <path>        Specify where to find user class files
  -sourcepath <path> Specify where to find input source files
  -bootclasspath <path> Override location of bootstrap class files
  -extdirs <dirs>   Override location of installed extensions
  -endorseddirs <dirs> Override location of endorsed standards path
  -d <directory>   Specify where to place generated class files
  -encoding <encoding> Specify character encoding used by source files
  -source <release> Provide source compatibility with specified release

  -target <release> Generate class files for specific VM version
  -version          Version information
  -help            Print a synopsis of standard options
  -X               Print a synopsis of nonstandard options
  -J<flag>         Pass <flag> directly to the runtime system

C:\Documents and Settings\Administrador.UA1-017>_
```

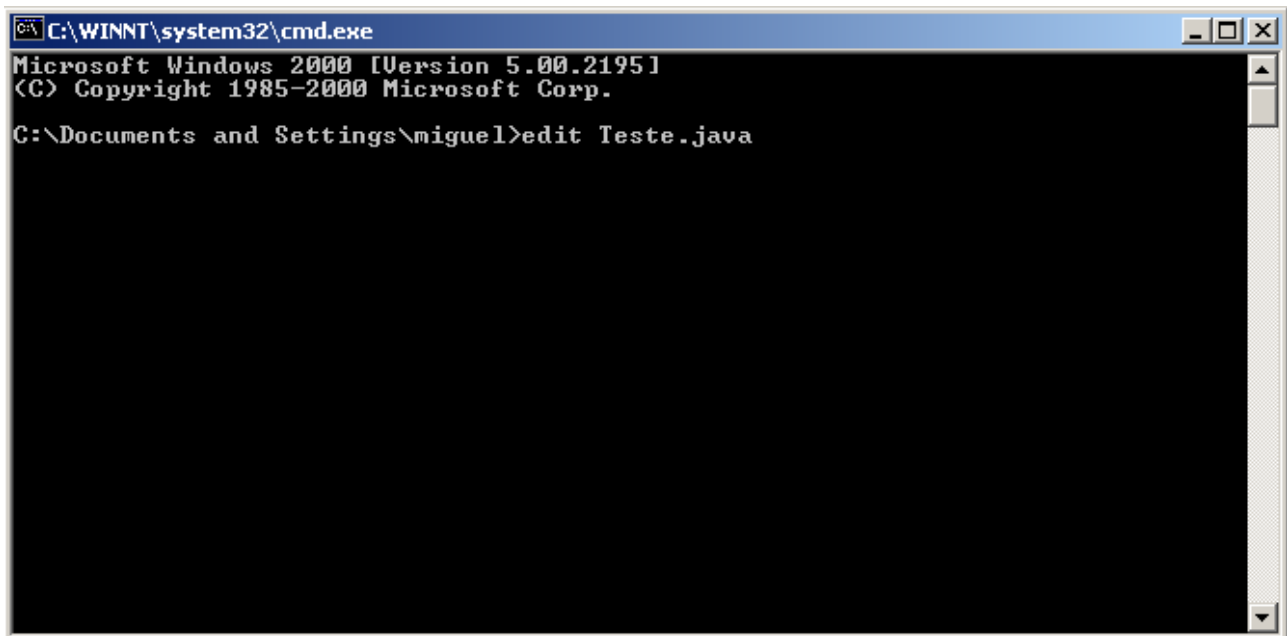
Nos dois casos se apareceram textos semelhantes aos apresentados

Curso Java Starter

significa que a instalação foi bem sucedida.

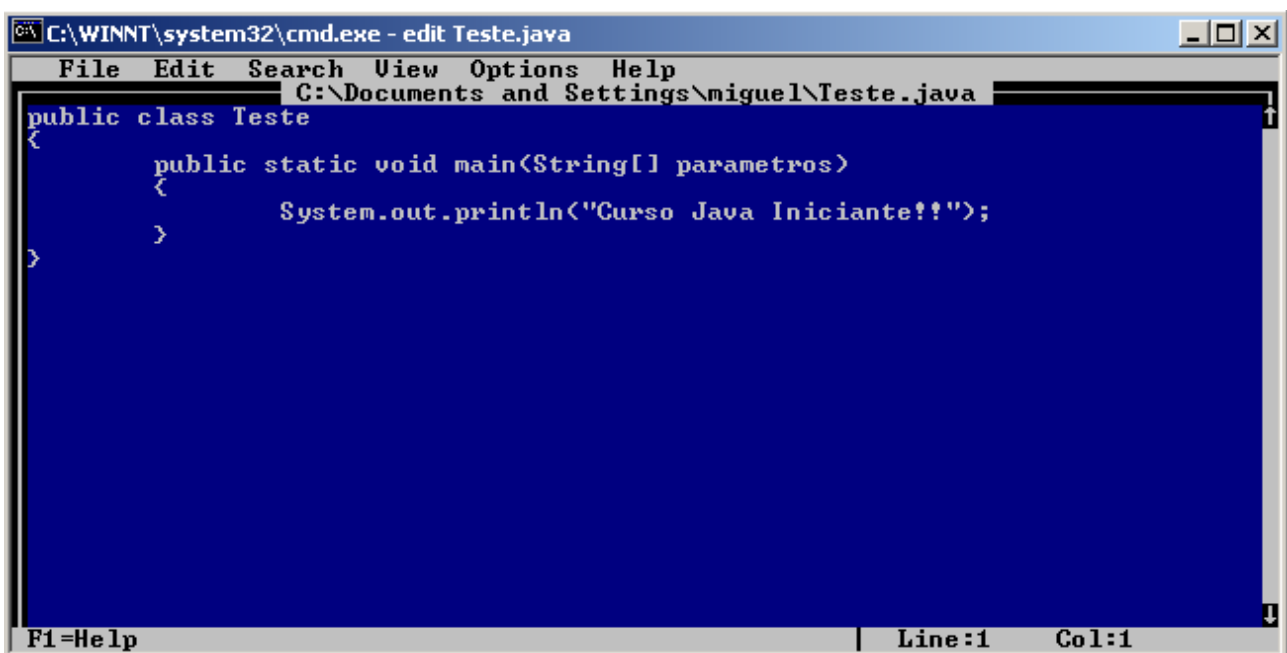
Primeira aplicação Java

Vamos criar a nossa primeira aplicação Java utilizando o editor do MS-DOS, para isto dirija-se ao prompt e digite **edit Teste.java**.



```
C:\WINNT\system32\cmd.exe
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.
C:\Documents and Settings\miguel>edit Teste.java
```

Dentro do editor digite o texto conforme abaixo:



```
C:\WINNT\system32\cmd.exe - edit Teste.java
File Edit Search View Options Help
C:\Documents and Settings\miguel\Teste.java
public class Teste
{
    public static void main(String[] parametros)
    {
        System.out.println("Curso Java Iniciante!!");
    }
}
```

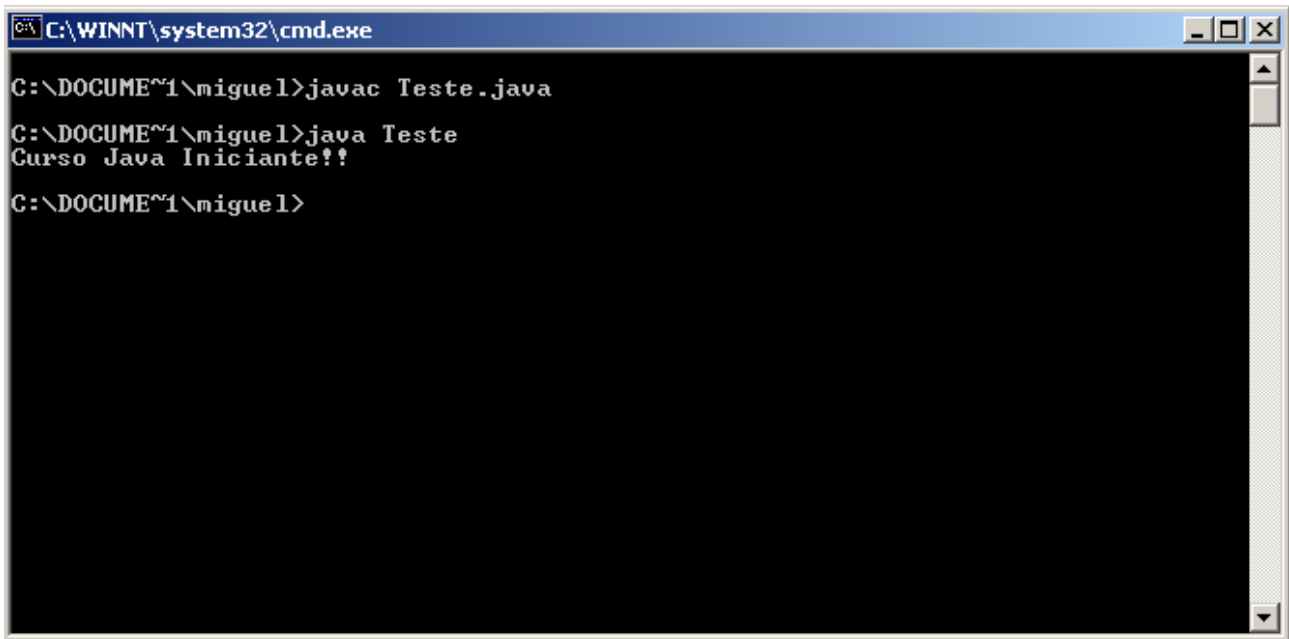
F1=Help | Line:1 Col:1

Curso Java Starter

Saia do editor pressionando **ALT + F e X**, se a versão do MS-DOS for em português a combinação de teclas será diferente (provavelmente **ALT + A e R**), ao sair, o editor perguntará se você deseja salvar. Escolha a opção afirmativa.

Retorne ao console e digite: **javac Teste.java**. Este comando irá compilar a nossa classe e gerar o bytecode (Teste.class).

Após a geração do arquivo compilado (.class) digite: **java Teste**. Este comando invoca o interpretador que irá transformar o nosso bytecode em código de máquina.



```
C:\WINNT\system32\cmd.exe
C:\DOCUME~1\miguel>javac Teste.java
C:\DOCUME~1\miguel>java Teste
Curso Java Iniciante!!
C:\DOCUME~1\miguel>
```

Pronto, temos nossa primeira aplicação Java sendo executada¹. Agora vamos entender um pouco de cada trecho do código digitado.

```
1. public class Teste
2. {
3.     public static void main(String[] parametros)
4.     {
5.         System.out.println("Curso Java Iniciante!!");
6.     }
7. }
```

Linha:

1. Declaração da classe pública de nome Teste;
2. Início do corpo da classe;
3. Declaração do método main (público, estático, sem retorno e parametrizado);

¹ A instalação do JDK e a criação da aplicação são demonstradas no mini-curso “Instalação do JDK”

4. Início do corpo do método;
5. Comando para impressão na tela;
6. Fechamento do corpo do método main;
7. Fechamento da classe.

Método main

Antes de iniciar o método main, vamos abrir um breve parênteses para introduzir a forma como comentamos o código em Java. Basicamente existem duas formas:

1. `// texto`: Esta é a forma de comentar apenas uma linha de código
2. `/* texto */`: Esta declaração é utilizada quando desejamos comentar mais de uma linha de código

Vejam os a classe anterior agora com comentários:

```
//Este é o comentário de uma linha
public class Teste {

    /*
    Este é o comentário
    de mais de
    uma linha
    */

    public static void main(String[] parametros) {
        System.out.println("Curso Java Iniciante!!");
    }
}
```

Retornando ao main, temos o seguinte:

```
public static void main(String[] parametros)
```

O main é o método que inicia as aplicações Java, quando solicitamos ao interpretador que execute uma determinada classe compilada ele procura o método main, se este método não existir irá ser gerada uma exceção informando que o método não foi localizado.

A JVM só irá reconhecer o método main se ele possuir as seguintes características:

1. Ser público (public);
2. Estático (static);
3. Não retornar nenhum valor (void);
4. O nome deve ser "main";
5. Receber como parâmetro um array de String.

Exercícios

Aprenda com quem também está aprendendo, veja e compartilhe as suas respostas no nosso [Fórum](#):

[Exercícios – Módulo 01 – Introdução ao Java](#)

1. Qual a diferença entre JRE e JDK?
2. Quais são os componentes da JDK?
3. Instale o JDK na sua máquina.
4. Crie uma classe que imprima o seguinte texto "Terminei o módulo 3 com um programa Java!".
5. Compile e execute a classe desenvolvida no exercício anterior.
6. Comente o trecho de código responsável pela execução da impressão, compile e execute.
7. Mude o nome do método "main" para "start", compile e execute. O que aconteceu?
8. Mude novamente o nome do método "main", agora para "#main", compile e execute. O que aconteceu?
9. Qual dos exercícios, 7 ou 8, gerou uma exceção durante a compilação? E durante a execução?