
VERTICAL Training
Treinando para o Sucesso Profissional
Java
Introdução a
Linguagem Java

Alexandre Faria

Desenvolvedor e Instrutor JAVA e Web da Vertical Training.
Trabalhando em projetos envolvendo tecnologias Java / J2 EE.
Experiência em análise, arquitetura e desenvolvimento de sistemas utilizando conceitos de Web 2.0 com integração em banco de dados dentro de grandes empresas.

Agenda

- História do Java
- Evolução
- Paradigmas da Programação

História do Java

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;

História do Java

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, 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;

Evolução

À medida que o mercado exige informatização torna-se cada vez mais necessário o uso de programas mais complexos e pesados, além de aumentar a velocidade de processamento e capacidade de armazenamento do hardware.

Podemos medir a evolução do hardware quantitativamente através do seu poder de processamento (clock do processador, medido em Hertz ou mega Hertz) e da sua capacidade de armazenamento (disco rígido - medido em mega bytes ou mais comumente em giga bytes).

Evolução

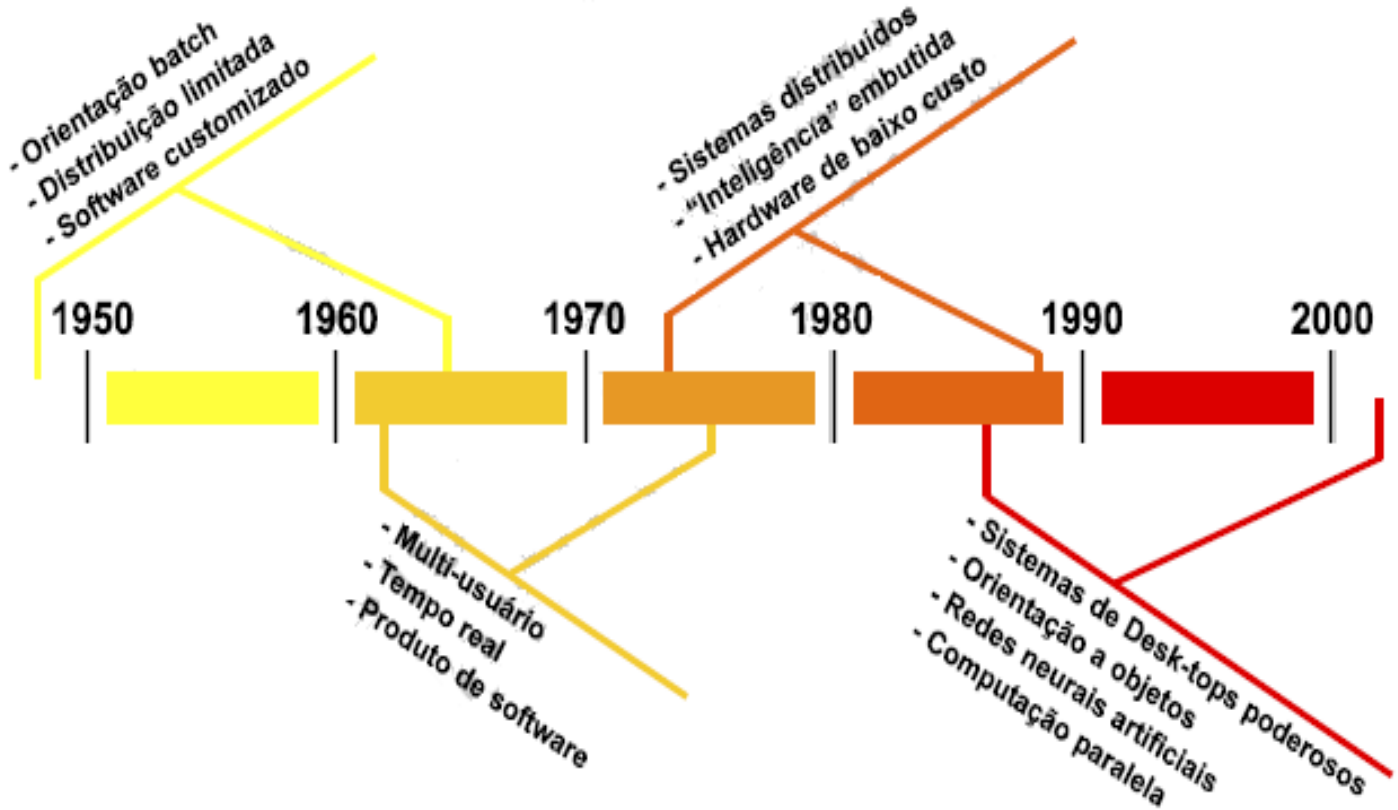
Qualidade do software: confiabilidade, operabilidade, manutenibilidade, extensibilidade, escalabilidade, entre outras métricas.

Depende do processo de desenvolvimento do software. Por isto, para que haja maior produtividade e qualidade do software existem cada vez mais softwares de apoio, tais como:

- **Compiladores;**
- **Ambientes de desenvolvimento;**
- **Servidores de aplicação;**
- **Banco de dados;**
- **APIs e frameworks.**

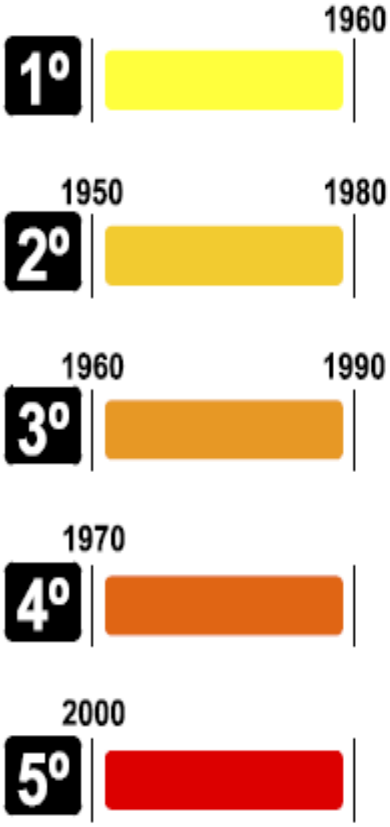
Evolução

Evolução tecnológica



Evolução

ão das linguagens de programação



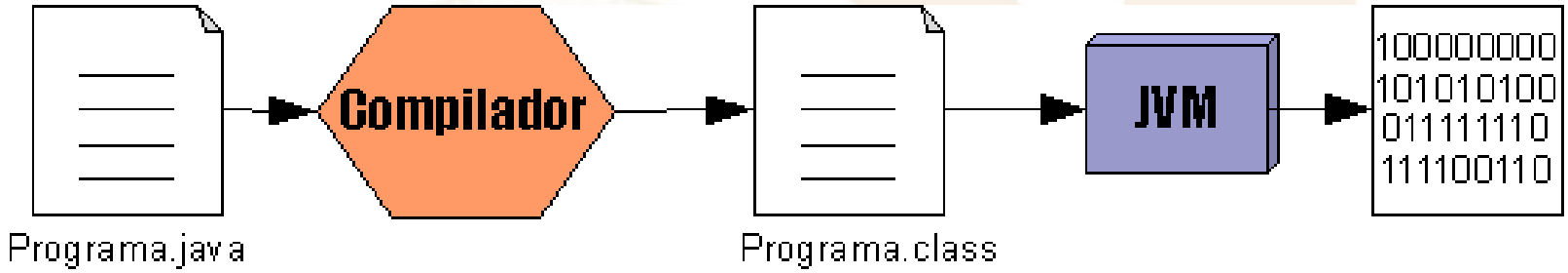
1º	- Código de Máquinas - Representa o mais baixo nível de abstração com o qual um programa pode ser representado	Assembler
2º	- Base para todas as linguagens modernas - Amplo uso - Enormes bibliotecas de software	Fortran, Cobol, Algol, Basic
3º	- Linguagem de programação estruturada - Linguagem Orientada a Objetos	C, Pascal, C++, Java, Smalltalk, Lisp, Prolog
4º	- Linguagem de consulta - Linguagem Orientada a Objetos - Geradores de código	SQL, Java
5º	Próximas linguagens reconhecerão vozes e gestos	

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.

Fases 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.

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 alteraçõesnas 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.;
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 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;
- **javadoc**: 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.

Ambiente

Alguns ambientes de desenvolvimento Java são:

- **JBuilder (www.borland.com)**
- **NetBeans (http://www.netbeans.org)**
- **Java Studio Creator (www.sun.com)**
- **JEdit (www.jedit.org)**
- **IBM Websphere Studio Application Developer (WSAD) (www.ibm.com)**
- **Eclipse (www.eclipse.org)**
- **JDeveloper (www.oracle.com)**

Nota: Todos os ambientes de desenvolvimento dependem do J2DKSE instalado.

Mitos da Linguagem

Java é da SUN?

A especificação Java foi criada pela SUN, no entanto, a linguagem é mantida pelo Java Community Process (JCP) que reúne Java experts, empresas e universidades que através de processos democráticos definem a evolução da linguagem.

Java é uma linguagem direcionada para Web?

Java não é apenas uma linguagem direcionada para Web, apesar de ser atualmente bastante conhecida e divulgada por seus "dotes" para desenvolvimento de aplicações Web, Java é uma linguagem completa como: C++, Pascal e Basic.

Mitos da Linguagem

Java é igual a JavaScript?

Não. Java é compilada e JavaScript é interpretada pelo interpretador contido no browser.

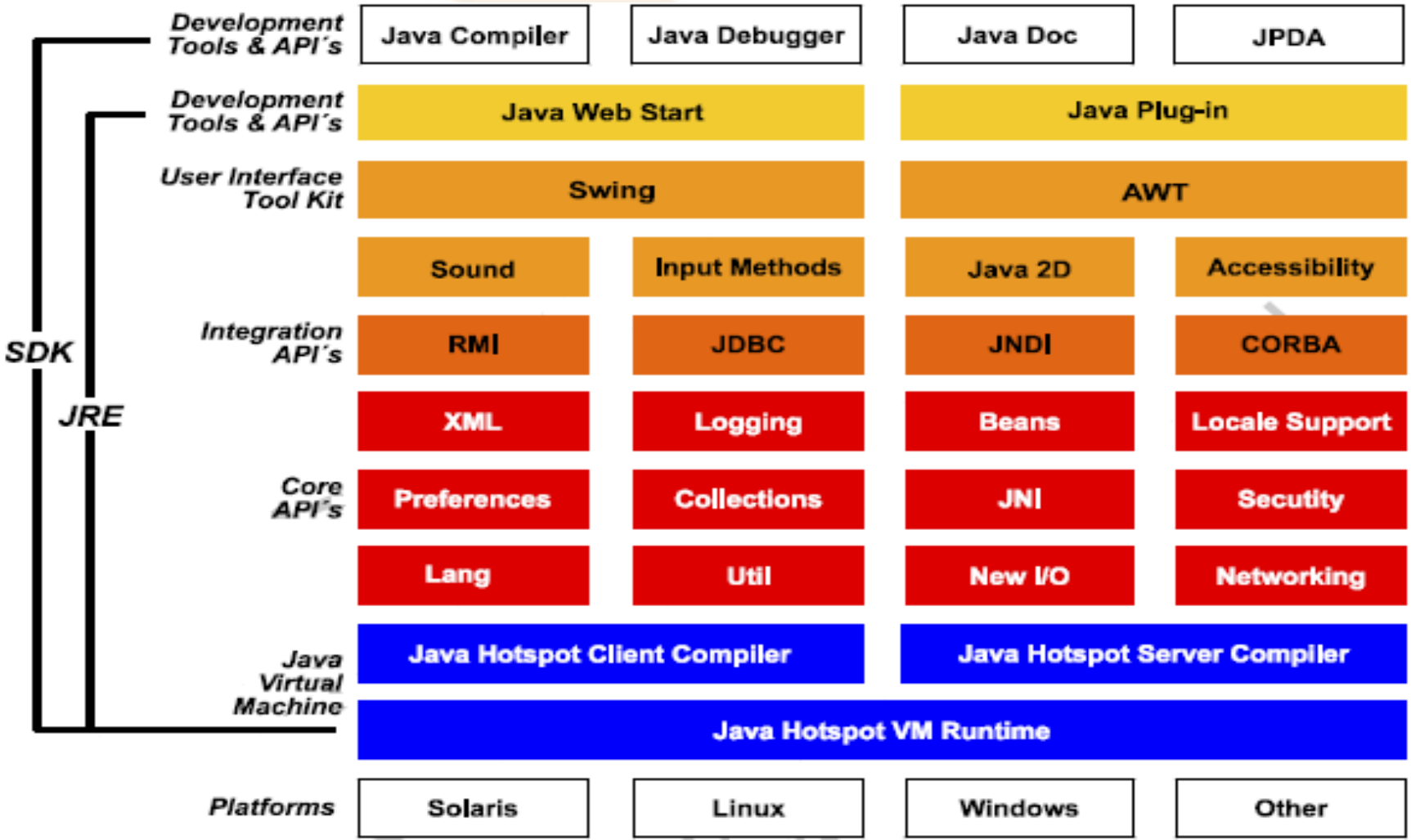
Java é uma criação da SUN e JavaScript é uma criação da Netscape. A linguagem JavaScriptoriginalmente chamava-se LiveScript, mas um acordo entre a Sun e a Netscape acabou fazendo com que LiveScript viesse a se chamar JavaScript.

Java é lento?

Java, como aplicação stand alone, é mais lento que uma linguagem compilada com código nativo (por exemplo, linguagem C), pois para ser portável não interage diretamente com o servidor gráfico do sistema operacional. No entanto, a afirmação de que "Java é lento" é completamente falsa para softwares distribuídos (em servidores), onde bibliotecas gráficas não são necessárias para gerar respostas aos usuários. (Servlet, JSP, RMI).

Java 2 Standard Edition(J2SE)

É a especificação do Java que contém APIs com as funções básicas do Java como I/O, multithread, network, conectividade com bancos de dados entre outras mais.



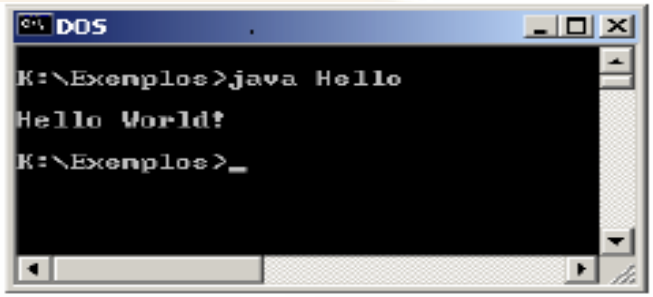
Java 2 Standard Edition(J2SE)

J2SE é composto por classes para atender as seguintes necessidades:

- **Classes essenciais**
- **Applets**
- **Networking**
- **Internacionalização**
- **Segurança**
- **Serialização de objetos**
- **Java Database Connectivity (JDBC)**
- **Utilitários**

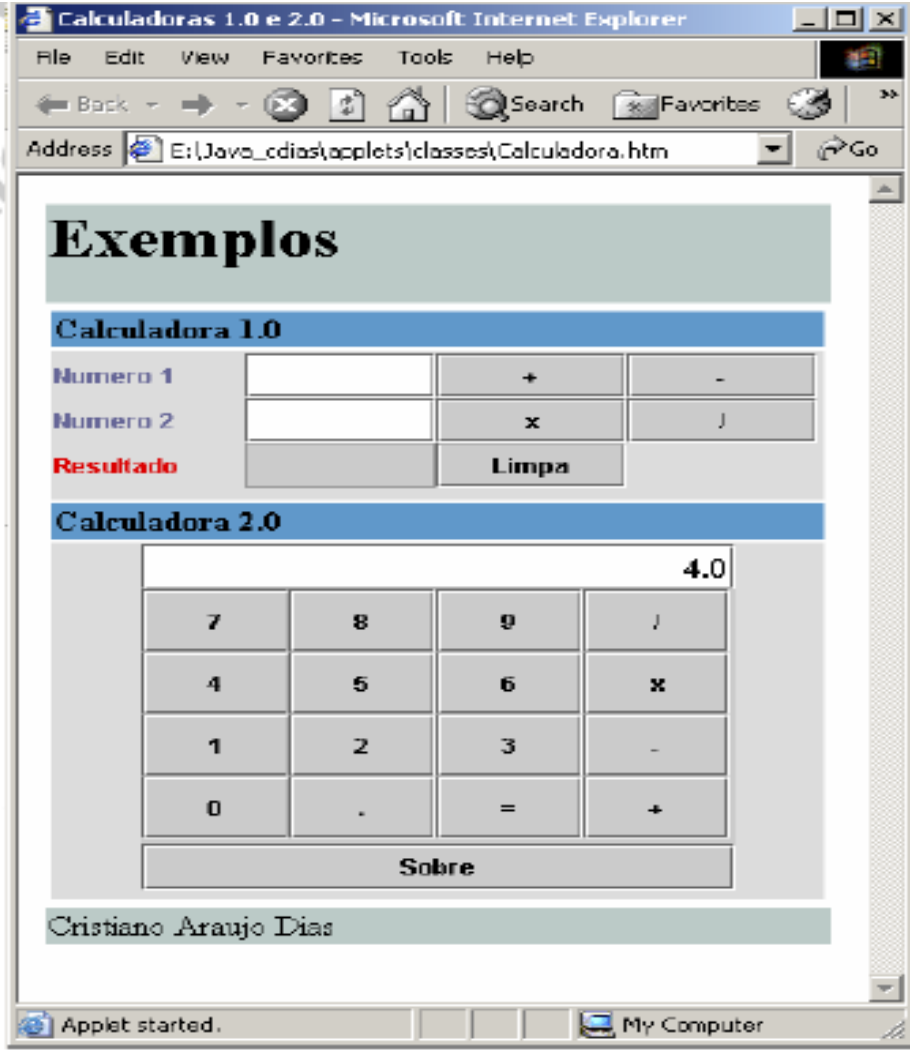
Java 2 Standard Edition(J2SE)

Veja alguns exemplos de aplicações implementadas utilizando a plataforma J2SE:



Console

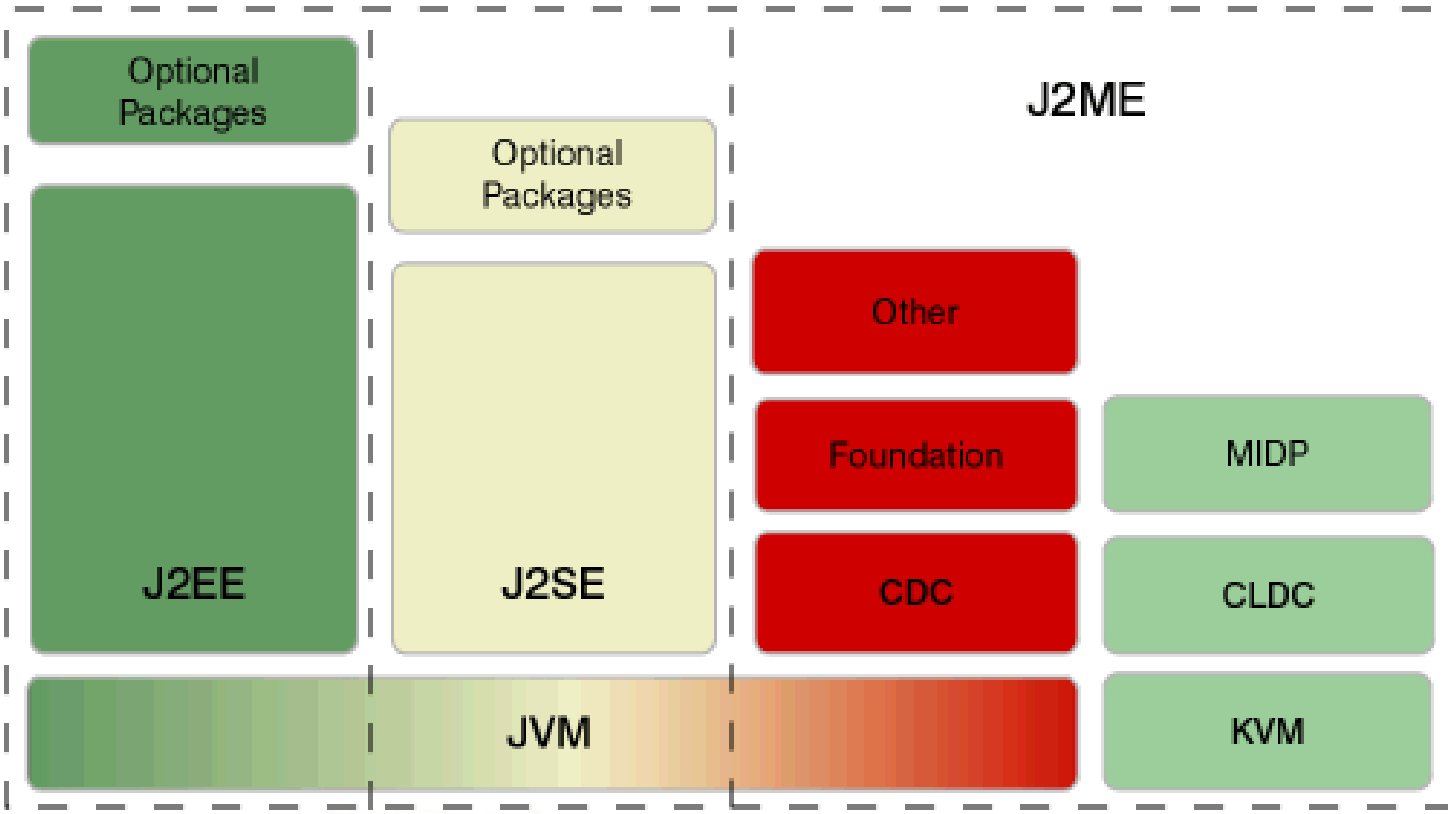
AWT / Swing



Applet

Java 2 Micro Edition (J2ME)

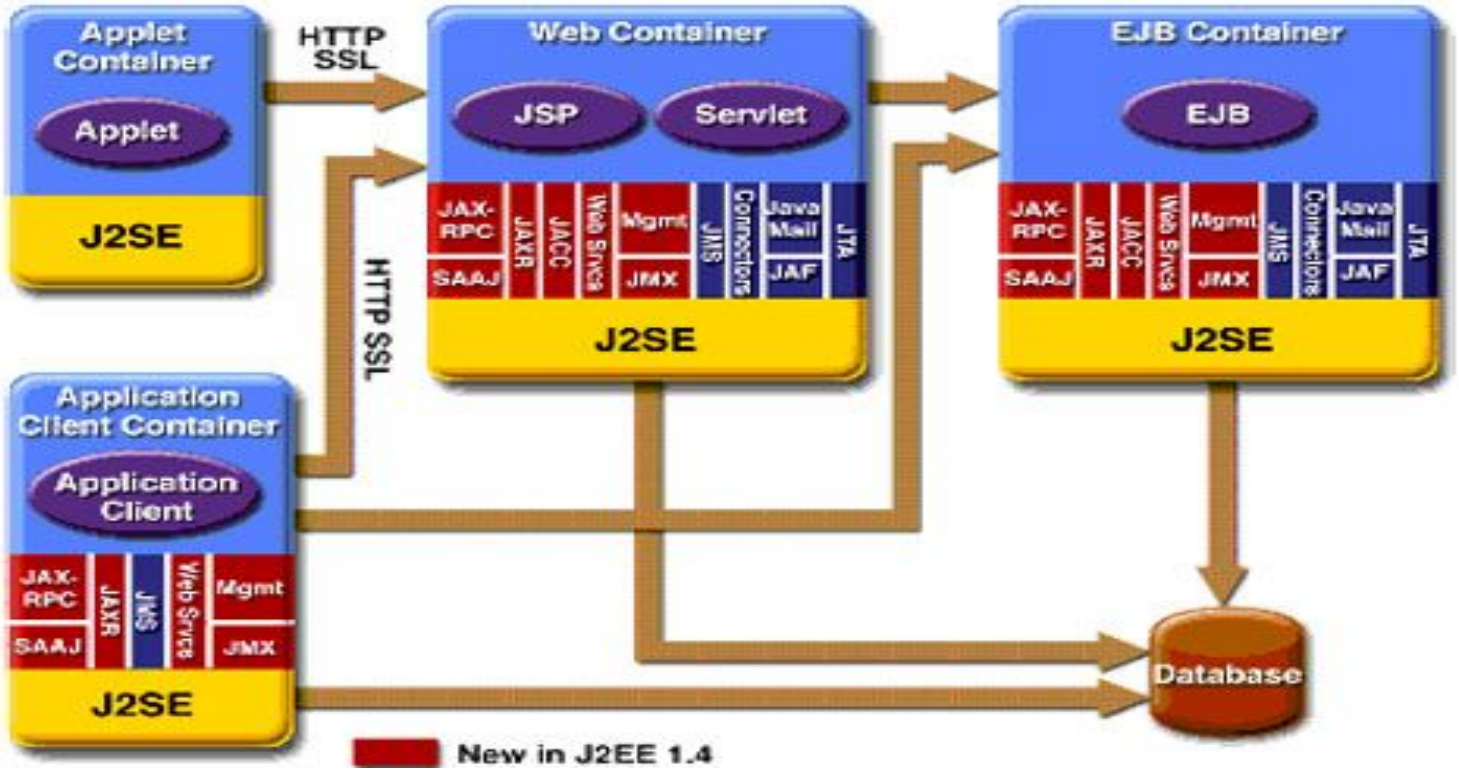
Esta plataforma é utilizada no desenvolvimento de pequenas aplicações para dispositivos móveis, como por exemplo: agendas eletrônicas, telefone celular, palmtop e aparelhos eletrônicos em geral, que possuam a KVM (Máquina Virtual para pequenos dispositivos).



Java 2 Enterprise Edition (J2EE)

É a especificação Java que contém APIs com funcionalidades específicas para o desenvolvimento de aplicações para servidores tais como, Servlets, JSP, EJB e JMS.

O J2EE é uma extensão ao J2SE e acompanha um servidor Web, um servidor de componentes transacionais de negócio, servidor de banco de dados, e um servidor de filas de mensagens.



Fundamentos da Linguagem

Palavras Chaves

byte	short	static	long	char	boolean	double
float	public	private	protected	int	abstract	final
strictfp	transient	synchronized	native	void	class	interface
implements	extends	const	else	do	default	switch
case	break	continue	assert	if	goto	throws

Declaração e Controle de Acessos

A tabela abaixo mostra os modificadores e onde podem ser usados:

modificador	classe	método	var instância	var local
padrão	sim	sim	sim	não
public	sim	sim	sim	não
protected	não	sim	sim	não
private	não	sim	sim	não
static	não	sim	sim	não
final	sim	sim	sim	sim
abstract	sim	sim	não	não
strictfp	sim	sim	não	não
synchronized	não	sim	não	não
transient	não	não	sim	não
native	não	sim	não	não
volatile	não	não	sim	não

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;

Instalação 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;

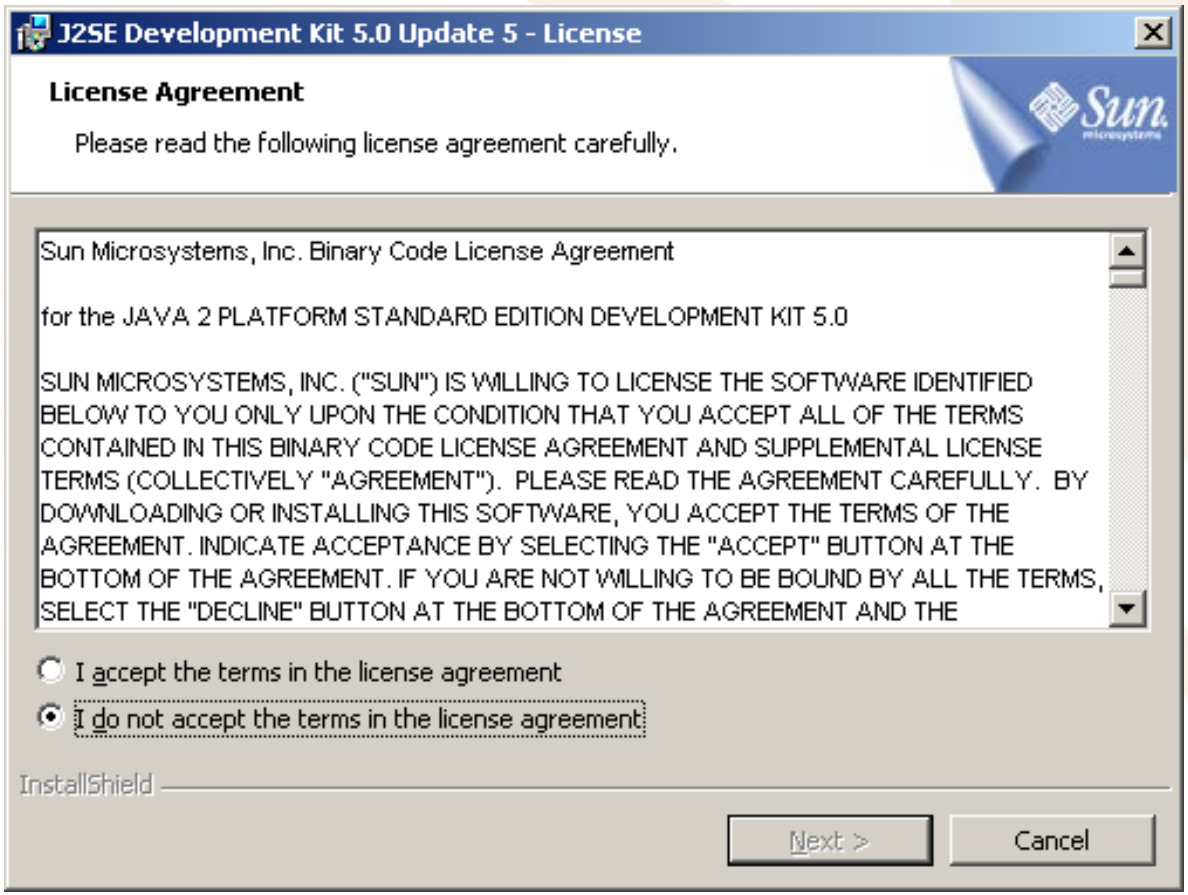
PATH: Identifica o local onde encontram-se as ferramentas De desenvolvimento (compilador, interpretador, gerador de documentação e etc.), devemos adicionar o diretório JAVA_HOME\bin;

CLASSPATH: Identifica diretório onde o ClassLoader pode encontrar classes que são utilizadas pela sua aplicação.

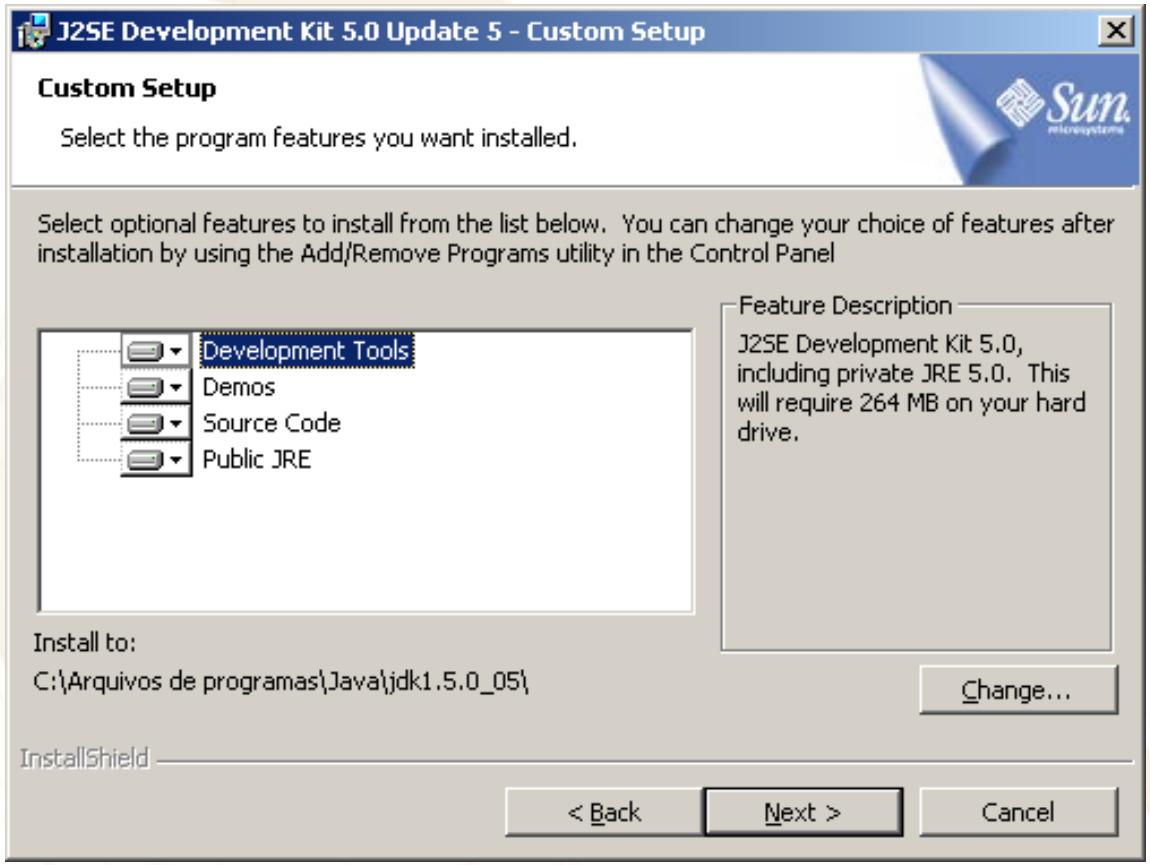
Instalação

Faça o download do JDK no site da SUN e execute o instalador

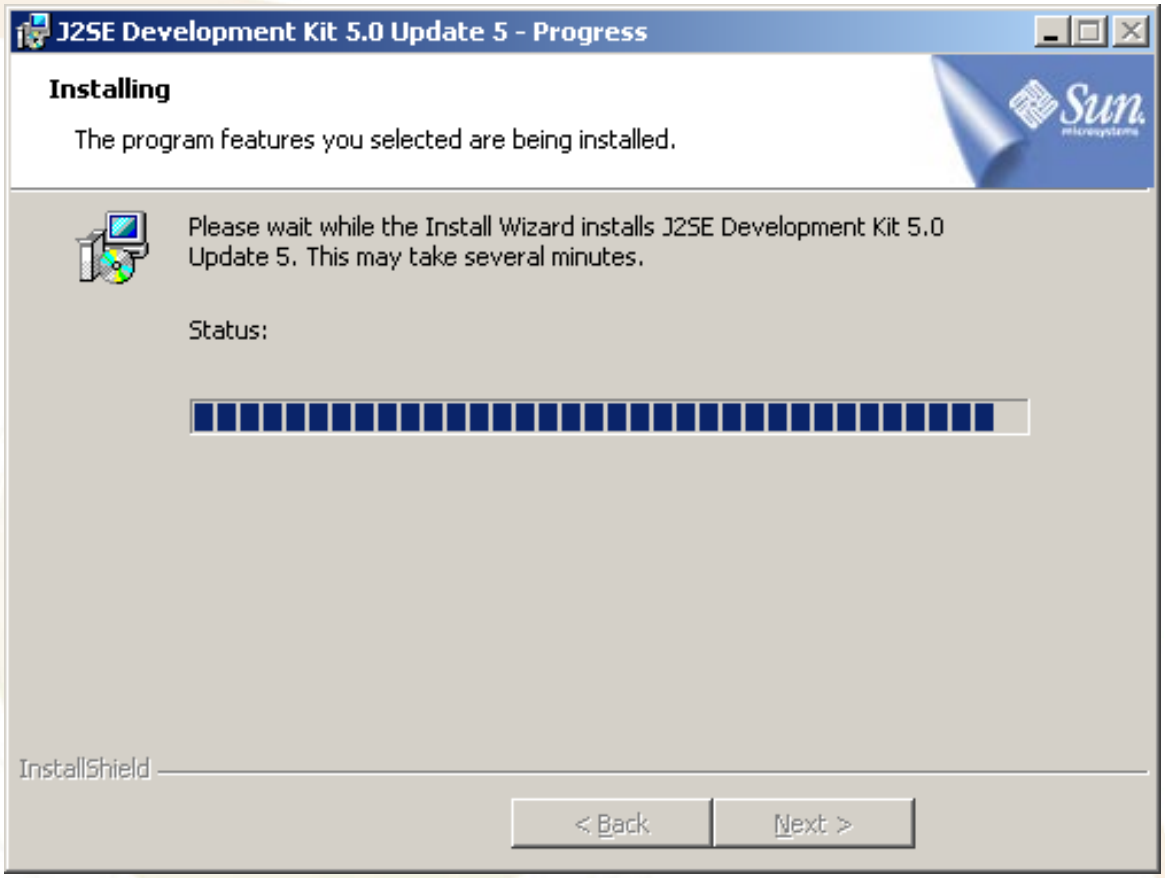
http://java.sun.com/javase/downloads/index.jsp



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

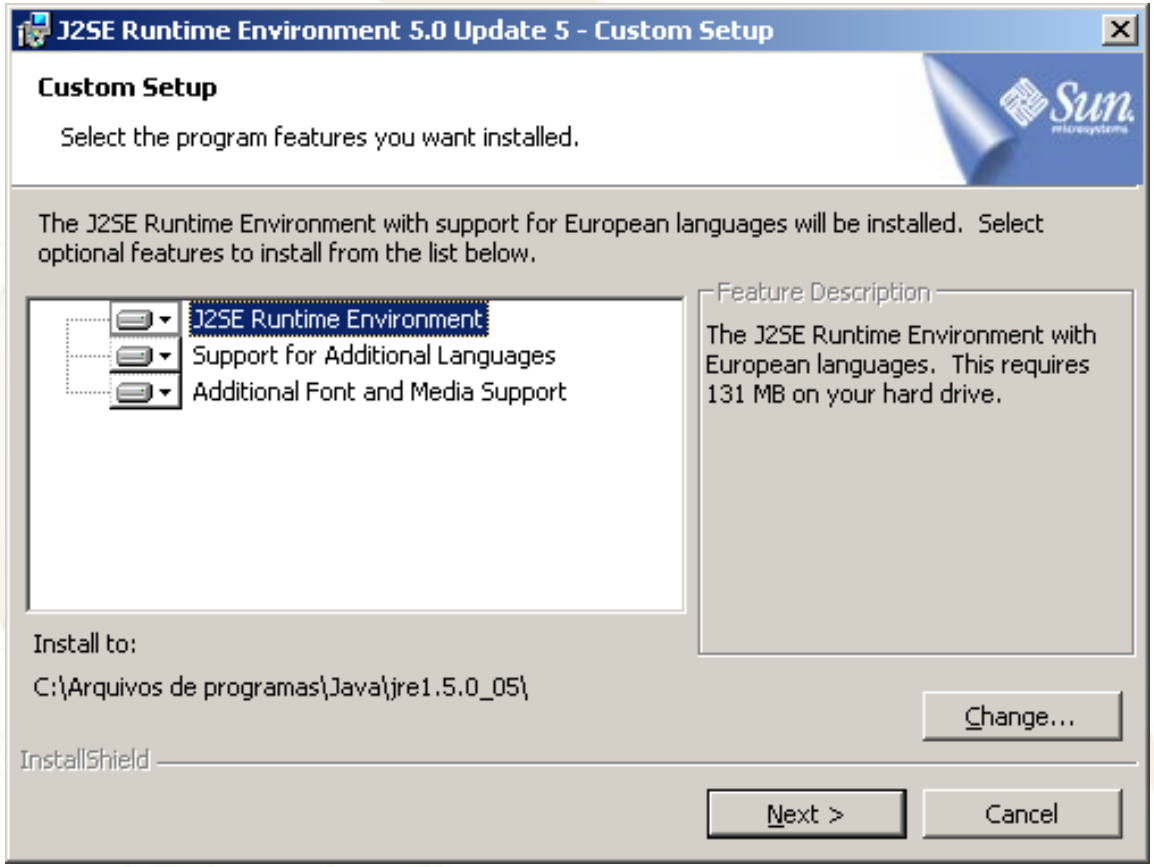


Instalação

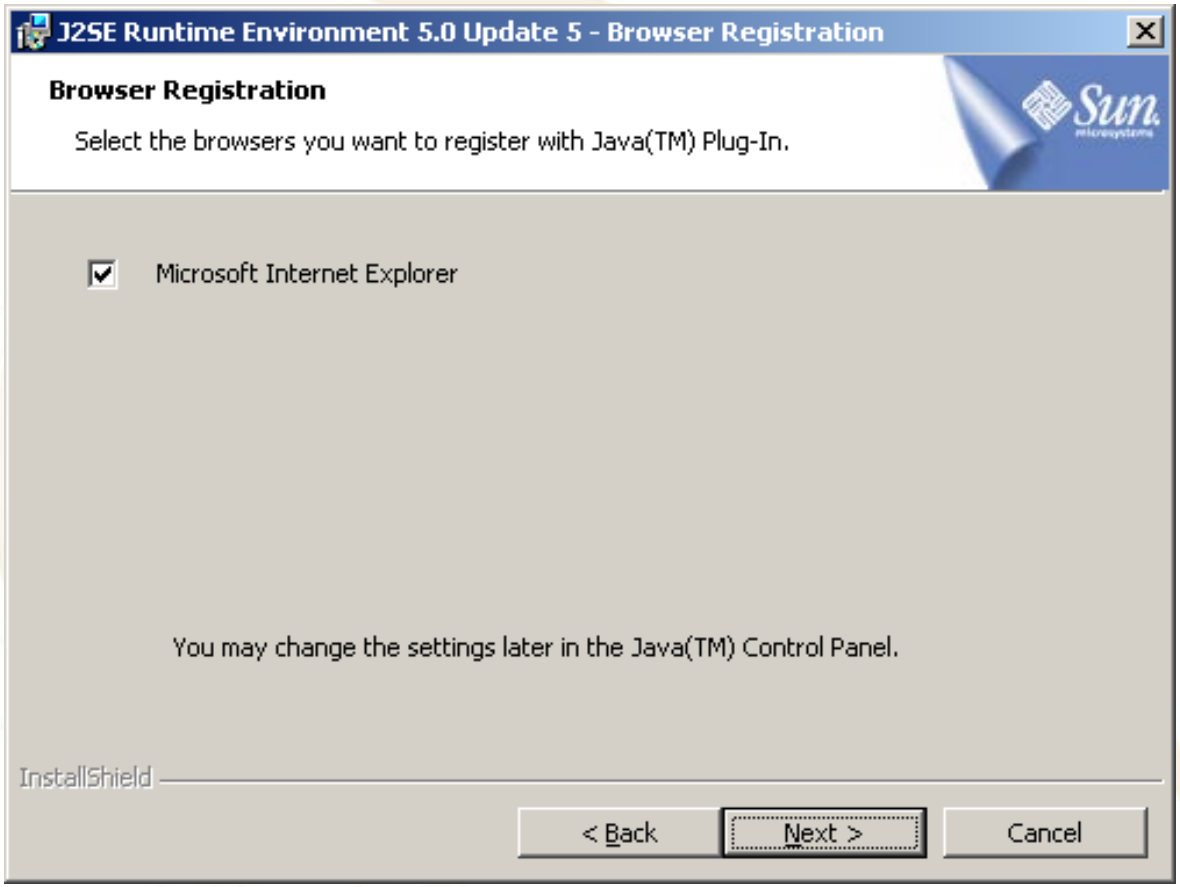

J2SE Development Kit 5.0 Update 5 - Progress
Installing
The program features you selected are being installed.
Sun microsystems
Please wait while the Install Wizard installs J2SE Development Kit 5.0 Update 5. This may take several minutes.
Status:
InstallShield
< Back
Next >

Instalação

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



Nesta tela selecione todos os navegadores disponíves.



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

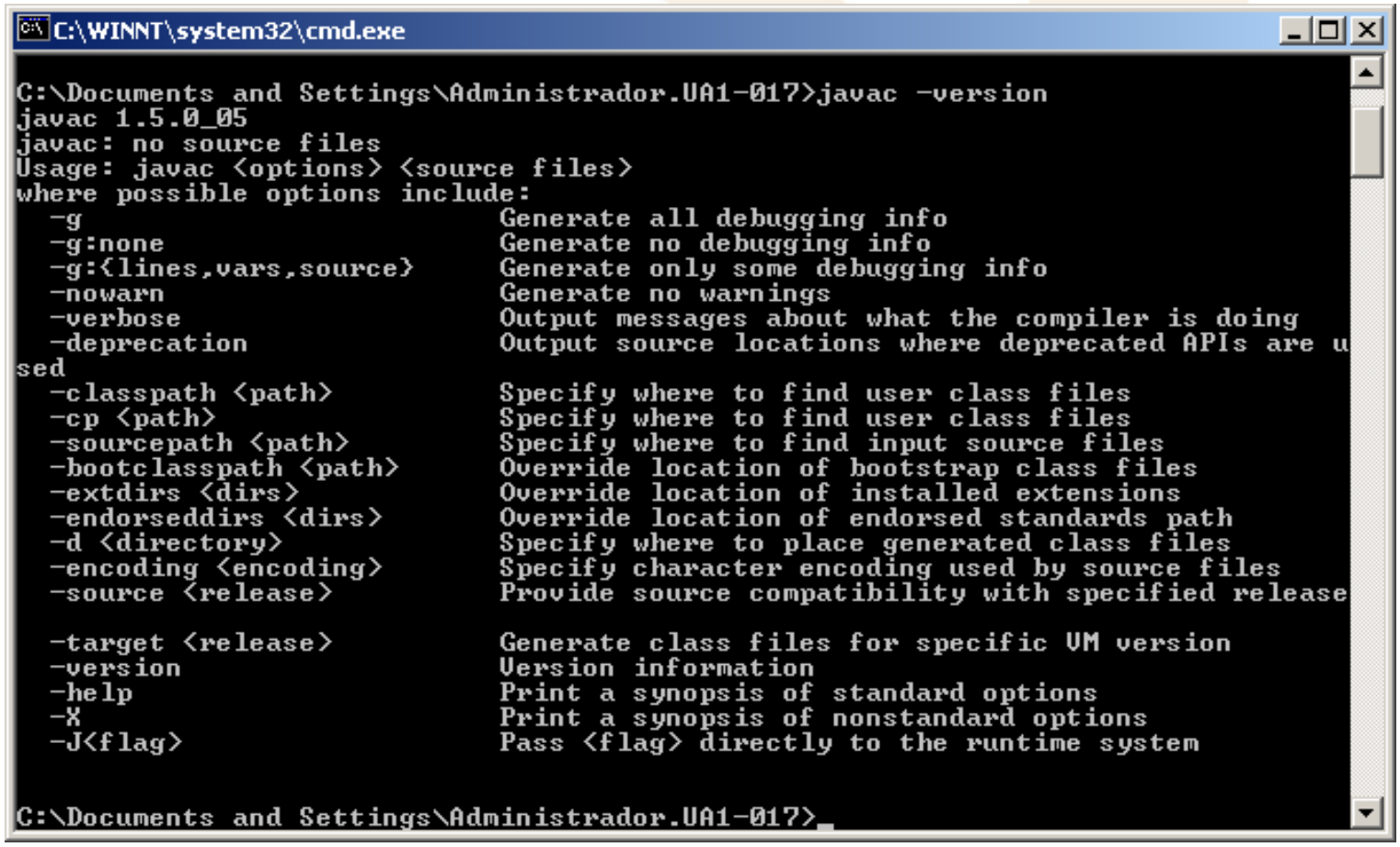
Configuração

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 váriavel PATH em “Variáveis do sistema”;
7.1. Adicione o seguinte endereço ao campo Valor da variável:
7.2.%JAVA_HOME%\bin

Testando o Compilador

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



Primeira Aplicação

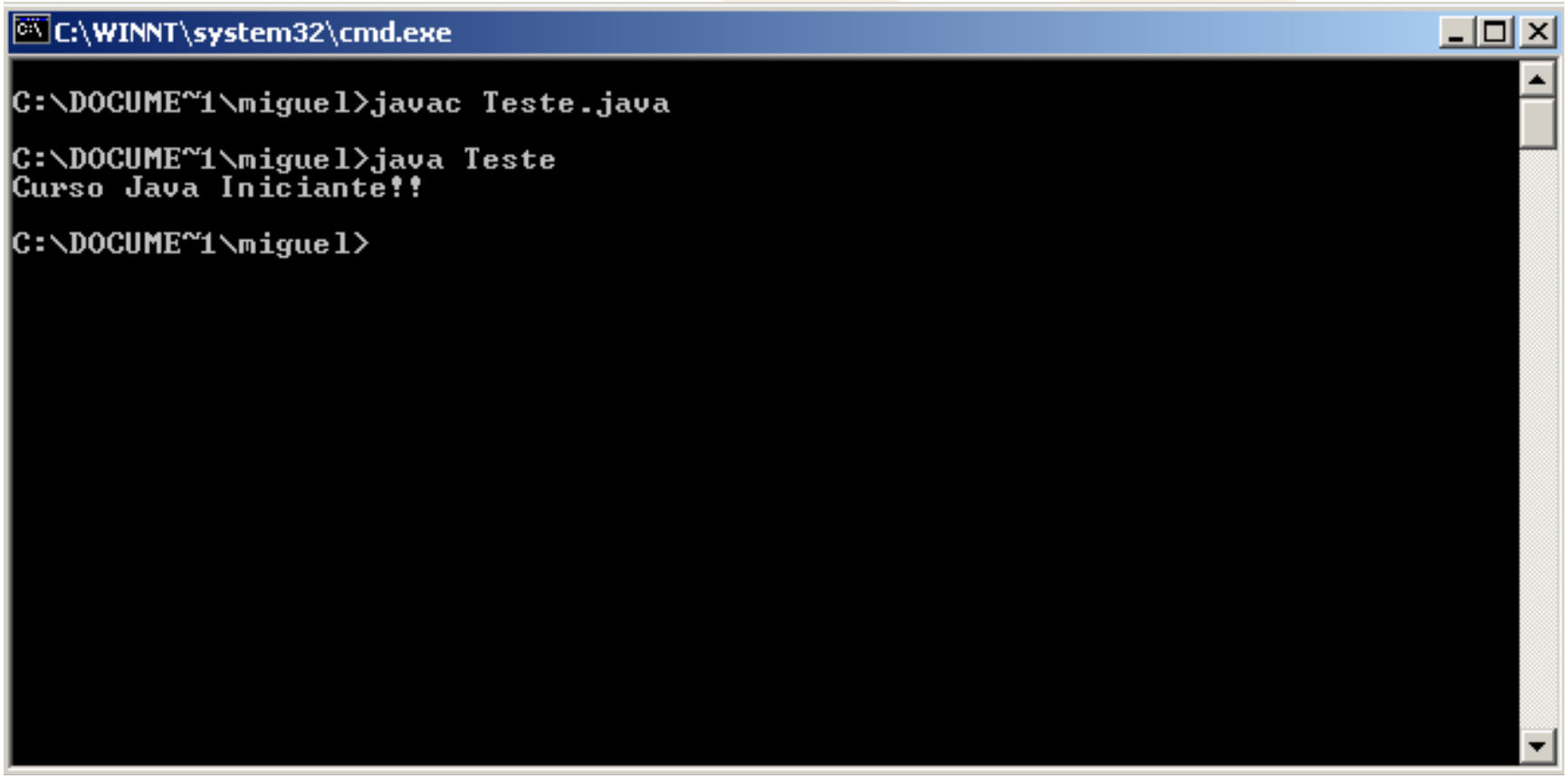
Vamos criar a nossa primeira aplicação Java utilizando Note Pad

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

Salve como Teste.java

Primeira Aplicação

Depois abra o Prompt do MS-DOS:



Método MAIN()

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

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.

Passo 2: classe principal

- A classe especificada deve ser
  - carregada (load)
  - ligada (link) a outros tipos que necessita
  - inicializada

Benvindo.class String.class System.class

```
class Benvindo {
public static void main(String[] args) {
        System.out.println("Benvindo a Java ");
    }
```

Convenção para Nomes

- Java, como C/C++ distingue entre letras maiúsculas e minúsculas
  - Exemplo: benvindo difere de Benvindo
- Nomes de classes iniciam com maiúscula
  - Exemplo: class Benvindo
- Nomes de variáveis iniciam com minúsculas
  - Exemplo: int peso;
- Nomes de métodos são verbos que iniciam com minúscula e após usam maiúsculas
  - Exemplo: alteraPeso
- Representação: Unicode (16 bits - 65.536 caracteres)

Classe Java: com dois métodos

```
class Benvindo2 {
        static void imprime(){
          System.out.println
          (“Benvindo a Java.”);
        }
public static void main_(String[] args) {
        imprime();
                }
}
```

método static: método de classe não aplicável sobre objetos

Servidores

O que são ?

É um software que disponibiliza um ambiente para a instalação e execução de certas aplicações. Os servidores de aplicação também são conhecidos como software de middleware.

O objetivo do servidor de aplicações é disponibilizar uma plataforma que abstraia do desenvolvedor de software algumas das complexidades de um sistema computacional.

Principais servidores comerciais:

- **WebSphere Application Server - IBM**
- **WebLogic Server – Oracle**

Principais servidores de software livre:

- **Glassfish, JBoss, JOnAS e Apache Geronimo.**
- **O Tomcat é um exemplo de container de software livre, onde os módulos Web podem ser publicados.**

DUVIDAS?

CONTATO:

alexandre@verticaltraining.com.br

(11) 8015-8856

Exemplo 1

- Programa que gerencia uma conta só. O usuário usa o sistema para sacar, depositar e consultar saldo;
- A conta contém dinheiro;
- Usuário não precisa se logar;

Exemplo 1

- Identificando classes a partir dos requisitos:
  - Conta;
  - Dinheiro;
  - Usuário.
- Estas são classes do **modelo de domínio**;
- De acordo com os requisitos, não precisaremos das classes Usuário e Dinheiro.

Exemplo 1

- Logo, temos apenas a classe conta.
- Agora, classes de auxílio:
  - Menu;
- Menu é responsável por realizar as ações que o usuário quer;
- Padrão Command.

Exemplo 2

- Melhore o sistema anterior para que o usuário tenha mais de uma conta;
- Precisaremos armazenar as contas em um banco – classe Banco;
- Podemos colocar uma descrição em cada conta para que o usuário possa identificá-las;
- Mas opções no menu.