**01Introdução**

**Transcrição**

Meu nome é Paulo Silveira, e serei o instrutor destes cursos de **Java**! Se você está procurando várias linguagens para começar a estudar, não tem jeito, uma das que mais costumam chamar a atenção é o Java!

Seja na Alura ou em qualquer outro lugar que ensine programação, esta linguagem tem atraído muito as pessoas, porque se você quiser desenvolver um site web, ou lidar com a parte do servidor, de back-end, de uma aplicação complicada, ou mesmo desenvolver uma aplicação para um dispositivo móvel, será preciso entender e conhecer não só sua sintaxe básica - que é o que veremos neste primeiro curso, entendendo as estruturas do if, for, conhecendo algumas das "pegadinhas" que aparecem com frequência. É necessário entender muito bem o conceito de orientação a objetos, polimorfismo, classe abstrata, interface, e as principais classes do Java.

Na documentação do Java - o Javadoc -, você verá todas as classes da biblioteca. São mais de dez mil! Então, dominar a linguagem, conhecer à fundo esta API, é algo que beira o impossível.

No entanto, no decorrer destes primeiros cursos de Java, conheceremos bibliotecas muito importantes a serem utilizadas no dia a dia, independentemente do uso, se para web ou desenvolvimento de aplicativo móvel.

Espero que esteja preparado para vir conosco nesta jornada, pois o Java, por ser uma linguagem estaticamente fortemente tipada, requer muito código, mesmo para criar programas simples. Mas ela é incrível!

E se você ainda não domina orientação a objetos, garanto que estudando aqui, você enxergará o que há de muito legal e interessante nisso, e por quê devemos focar nestes conceitos de encapsulamento, responsabilidades, que têm muita vantagem em relação à tal da programação procedural.

Vamos começar a entender um pouco sobre o Java para conseguirmos passar à instalação e compilação do nosso primeiro programa!

**02A plataforma Java**

**Transcrição**

Antes de mais nada, vamos ver um pouco do que é o Java, o qual te trouxe até aqui: há cerca de vinte anos, quando a linguagem Java nasceu, ela chamava a atenção por conta das seguintes características:

* Orientado a Objeto (O.O.)
* Muitas bibliotecas
* Parece com C++ (hoje em dia isso pode até ser uma desvantagem)
* Roda em vários sistemas operacionais

Você pode estar pensando "poxa, mas a linguagem que uso no dia a dia, atualmente, já possui estas características!". É verdade. É por isto que queremos focar na **plataforma Java**, e não especificamente na linguagem em si, algo que ficará mais claro no decorrer do curso, e até mesmo nesta aula!

A plataforma Java traz:

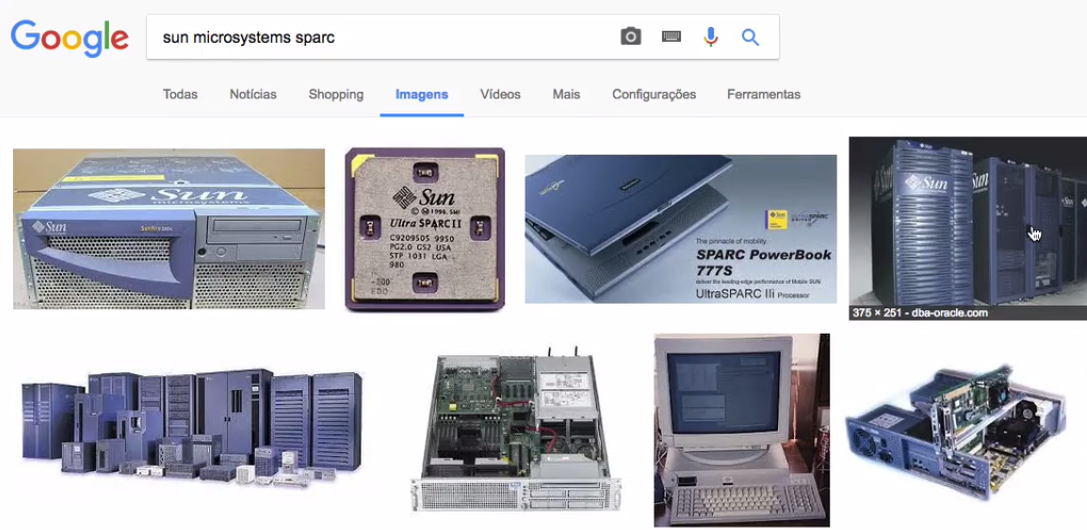
* Portabilidade
* Fácil acesso e desenvolvimento
* Segurança
* Onipresença

Você pode dar uma olhada no [site oficial](https://java.com/), porém ele ajuda mais o usuário do Java, do que aqueles que irão compilar e escrever programas.

Falando sobre a história da linguagem: James Gosling é considerado um dos "gênios da computação", sendo considerado o "pai do Java", apesar da linguagem ter sido criada por um grupo, normalmente considerado de quatro pessoas.

Em 1992, o James Gosling trabalhava em uma empresa atualmente inexistente chamada Sun Microsystems (sendo que Sun é acrônimo para *Stanford University Network*), uma dessas *startups* da década de 60, 70, para lidar mais com hardware, que é o que estava dando mais dinheiro.

Eles possuíam um microcomputador, o ***Sun Microsystems SPARC***, que hoje em dia já não aparecem em lugar algum, grandes servidores denominados "micro":



Sendo a Sun uma empresa mais focada em hardware, naquela época, a IBM e a Microsoft começaram a crescer vendendo softwares. Os softwares que a Sun utilizava no sistema deles, o UNIX (o tal de Solaris), eram disponibilizados gratuitamente.

Um dia, esses executivos, dentre os quais o próprio James Gosling, se perguntaram como poderiam lucrar com softwares, já que eles o disponibilizavam de graça, e fizeram um retiro de um mês para tentarem chegar a uma conclusão.

A ideia que eles tiveram envolvia um problema de eletrônicos da década de 90: havia muitos deles sendo criados naquela época, como o VHS que, para quem não sabe, é o videocassete. Era a época de surgimento de TVs, videogames, liquidificadores e geladeira.

Cada um deles possui seu código fonte, necessitando de uma linguagem própria para funcionar, e escrever o código para cada um, reescrevendo-o quando tivessem que passar por uma troca de chip, por exemplo, não fazia muito sentido! A linguagem utilizada neles, ***Assembly***, que hoje em dia é raramente usada, precisava ser reescrito várias vezes, imagine o trabalho!

O James Gosling e sua equipe pensaram em escrever um único código que gerasse um "executável" - entre aspas porque após a compilação ele estará em um formato não exatamente compreensível pelo aparelho em si, mas por um intermediário, no caso, um processador ou uma placa de hardware, para que, aí sim, passe o código aos aparelhos.

Trata-se de algo que realmente simula um computador bem simples e traduz esta linguagem "executável" de acordo com o aparelho em questão. Isto é, esta "máquina de mentira" traduzirá tudo, como se fosse um sistema operacional.

É por isto que surgiu o nome **máquina virtual**, pois veio da ***virtual machine***!

A ideia deles foi, então, criar uma placa pequena, um hardware, que é uma máquina real e compõe todo liquidificador, computador, videocassete, e por aí vai. Desta forma, as pessoas poderão escrever em apenas uma linguagem, que na época se chamava ***Oak*** e depois se tornou Java.

Isso pareceu muito bom, mas acabou fracassando de maneira retumbante, pois era muito caro produzir chips distintos para cada aparelho, cada qual adaptado a uma determinada linguagem.

Então, em 1995, com o *boom* da Web e o surgimento de mais navegadores, como Mosaic, Netscape e posteriormente Internet Explorer, a ideia de máquina virtual foi visualizada como um problema interessante pelo Gosling.

Assim como na atualidade, existia uma variedade relevante de navegadores e sistemas operacionais. E, para escrever um código para Windows, utilizava-se a linguagem no Microsoft Visual Basic, que por sua vez era compilado por um executável (um EXE, no caso do Windows).

Isto é, ele só funciona neste sistema operacional, com determinadas DLLs na máquina, e assim por diante. O executável e o código fonte ficavam atrelados a uma plataforma específica, um conjunto de sistema operacional, hardware e outros detalhes.

Para tentar resolver este problema, que geraria um código e um executável diferentes para cada sistema operacional existente, o Gosling desengavetou a ideia da máquina de verdade, do chip, que eles haviam criado anteriormente.

Com um código fonte único, teríamos um intermediário que soubesse traduzir ou instruir o sistema operacional acerca dos comandos a serem enviados e recebidos. Este meio de campo seria realizado pela **Máquina Virtual Java (JVM)**, que não é meramente um interpretador por conta de alguns detalhes internos que vão além da interpretação.

O código, então, seria a linguagem Java, e o código "executável", quando compilado, não geraria um .exe (pois este seria lido apenas pelo Windows), e sim um formato chamado ***bytecode Java***, de extensão .class, lido pela Máquina Virtual Java, que passaria a informação aos sistemas operacionais.

Um exemplo deste formato entendido pela *virtual machine* (JVM), o *bytecode*, é o seguinte:

Compiled from "Onibus.java"

class Teste {

public static void main(java.lang.String);

Code:

0: new #2 // class Onibus

3: dup

4: invokespecial #3 Onibus."<init>":()V

7: astore\_1

8: aload\_1

9: ldc #4 // String Jabaquara...

11: putfield #5

// Field Onibus.linha:Ljava/lang/String;

14: return

}Copiar código

Quem conhece a linguagem de ***Assembly*** talvez identifique a semelhança, mas este código não parece ser de fácil leitura e compreensão. Para meios de comparação, segue um exemplo de um arquivo .java, a ser compilado e traduzido para .class, o tal do bytecode:

public class Onibus {

String nome;

String linha;

}

class Teste {

public static void main(String args) {

Onibus o = new Onibus();

o.linha = "Jabaquara-Liberdade";

}

}Copiar código

Então, em 1995 surgiu o Java, capaz de rodar em vários dispositivos e sistemas operacionais, com foco de criar *applets*, quando ainda tínhamos que instalar o Java para rodá-lo dentro do navegador.

O Java nasceu com um propósito, mas acabou se fortalecendo em ***server-side***, pois quando escrevemos uma aplicação, um site web ou sistema grande, não queremos ficar dependendo de diferentes sistemas operacionais, em implantações e *deploys*.

O Java traz liberdade, quebrando nossa dependência em relação às versões de sistema operacional e navegadores. Empresas grandes, como bancos e o governo, não querem ficar engessados - o que é conhecido por *Vendor lock-in*.

As principais características do conceito de Máquina Virtual Java são:

* Multiplataforma
* Gerenciamento de memória
* Segurança
* Sandbox
* Otimizações
* JIT Compiler

Hoje, mais do que na linguagem Java em si, o enfoque está na plataforma, no **ecossistema Java**! A *virtual machine* é interessante para as empresas pois elas não dependem do que se encontra abaixo da sua *stack*, ou pilha de tecnologia, além do acesso a uma grande variedade de bibliotecas, e as linguagens Java que rodam nesta plataforma.

Não é à toa que há programas que lidam com linguagens Ruby, Clojure ou Scala, por exemplo, e geram o bytecode Java. Depois, basta a Máquina Virtual Java, JVM, trabalhar de acordo com o sistema operacional desejado.

**03Benefício da JVM**

Qual é o grande benefício de uma Máquina Virtual Java (JVM)?

Parte superior do formulário

* Compilar o código independente do sistema operacional.
* Alternativa correta

Executar o código independente do sistema operacional.

Correto, no mundo Java você terá sempre o mesmo "executável" ou Bytecode que será executado pela Maquina Virtual Java, totalmente independente do sistema operacional. Assim não é preciso rescrever ou adaptar o código para rodar em um outro sistema operacional. Temos um único executável para todos os sistemas!

* Alternativa correta

Executar o código usando sempre o mesmo chip hardware.

Parte inferior do formulário

Parabéns, você acertou!

Este conteúdo foi útil para o seu aprendizado?

Sim

Não muito

**04Quais características?**

A linguagem Java nasceu com algumas características que são mantidas até hoje.

Selecione todas que se aplicam:

Parte superior do formulário

* Parecido com C++.

Correto, Java nasceu sendo parecido com C++ que era a linguagem mais popular na época.

* Alternativa correta

Muitas bibliotecas.

Correto, desde o início o Java possui muitas bibliotecas que ainda muito aumentaram devido as milhares de contribuições da comunidade open source.

* Alternativa correta

Procedural.

* Alternativa correta

Multiplataforma.

Correto, Java é multiplataforma, roda em qualquer sistema operacional desde que exista a máquina virtual Java.

Parte inferior do formulário

Parabéns, você acertou!

**05Quais sistemas?**

Dependendo da linguagem de programação existem vários tipos de sistemas que você pode desenvolver, como por exemplo:

a) Sistemas para Web (Web Sites ou Web Apps)

b) Sistemas puramente "Server-Side"

c) Aplicativos para Android

d) Applets que rodam no navegador

Quais tipos de sistemas você pode criar com a linguagem Java?

Parte superior do formulário

* a), b) e d)
* Alternativa correta

Todos.

Correto, seja web ou puramente server-side, aplicativo Android ou um antigo applet, Java atende todas essas áreas! No entanto, os sistemas web e Android são os que mais fizeram sucesso no mundo Java e mais empregam desenvolvedores.

* Alternativa correta

Apenas a).

* Alternativa correta

Apenas a) e b).

Parte inferior do formulário

Parabéns, você acertou!

Este conteúdo foi útil para o seu aprendizado?

Sim

Não muito

**06Bytecode vs EXE?**

Qual é a diferença entre o executável do Windows (exe) e o executável do mundo Java (Bytecode)?

Parte superior do formulário

* Os executáveis do Windows são portáveis, os do mundo Java não.
* Alternativa correta

Os executáveis do mundo Java são portáveis, os do mundo Windows não.

Correto, lembrando *portável* significa que podem ser executados em vários sistemas operacionais diferentes (Windows, Linux, Mac, etc).

* Alternativa correta

Os executáveis do mundo Windows podem ser executados diretamente no sistema operacional, os do mundo Java precisam da máquina virtual.

Correto! Os executáveis Java precisam da máquina virtual Java.

* Alternativa correta

Os executáveis do mundo Java podem ser executados diretamente no sistema operacional, os do Windows precisam da máquina virtual.

Parte inferior do formulário

Parabéns, você acertou!

**07Sobre o Bytecode**

Todas as afirmações abaixo sobre o Bytecode são verdadeiras, **exceto**:

Parte superior do formulário

* O Bytecode é parecido com o Assembly (código de montagem que é traduzido para código de máquina).
* Alternativa correta

O Bytecode é o código fonte.

A afirmação "O Bytecode é o código fonte" não é verdadeira. O código fonte é para o desenvolvedor entender, o Bytecode foi criado e otimizado para a máquina virtual.

* Alternativa correta

O Bytecode é executado (interpretado) pela JVM.

* Alternativa correta

O Bytecode é independente do sistema operacional.

Parte inferior do formulário

Parabéns, você acertou!

Este conteúdo foi útil para o seu aprendizado?

Sim

Não muito

**08Para saber mais: o nome Bytecode**

Já falamos um pouco sobre o Bytecode que é um código de máquina parecido com o Assembly. Talvez você (como eu!) estranhou o nome Bytecode, no entanto, tem uma explicação bem simples para tal. Existe um conjunto de comandos que a máquina virtual Java entende. Esses comandos também são chamados de opcodes (operation code), e cada opcode possui o tamanho de exatamente 1 Byte! E aí temos um **opcode de 1 Byte** ou, mais simples, **Bytecode**. :)

**09O que aprendemos?**

Nessa aula introdutória já aprendemos vários assuntos fundamentais sobre Java.

Falamos sobre as principais características da linguagem Java como:

* orientado a objetos
* parecido com C++
* muitas bibliotecas e grande comunidade

Além disso, aprendemos:

* a diferença entre o código fonte e o *Bytecode*
* para executar o *Bytecode* é preciso ter a máquina virtual java
* o *Bytecode* é independente do sistema operacional

Vimos também os principais componentes da plataforma Java que são:

* Java Virtual Machine (JVM)
* linguagem Java
* Bibliotecas Java (API)

Na próxima aula já vamos escrever o primeiro código Java! Vamos continuar?

Este conteúdo foi útil para o seu aprendizado?

Sim

Não muito