Introdução ao JAVA

Professor Leandro

Java
Porque usar?

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.

Java
Porque usar?

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.

Java
O que é o Java?

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

Java
O que é o Java?

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!

A plataforma Java traz:

- Portabilidade
- Fácil acesso e desenvolvimento
- Segurança
- Onipresença

Java
O que é o Java?

Você pode dar uma olhada no site oficial, 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.

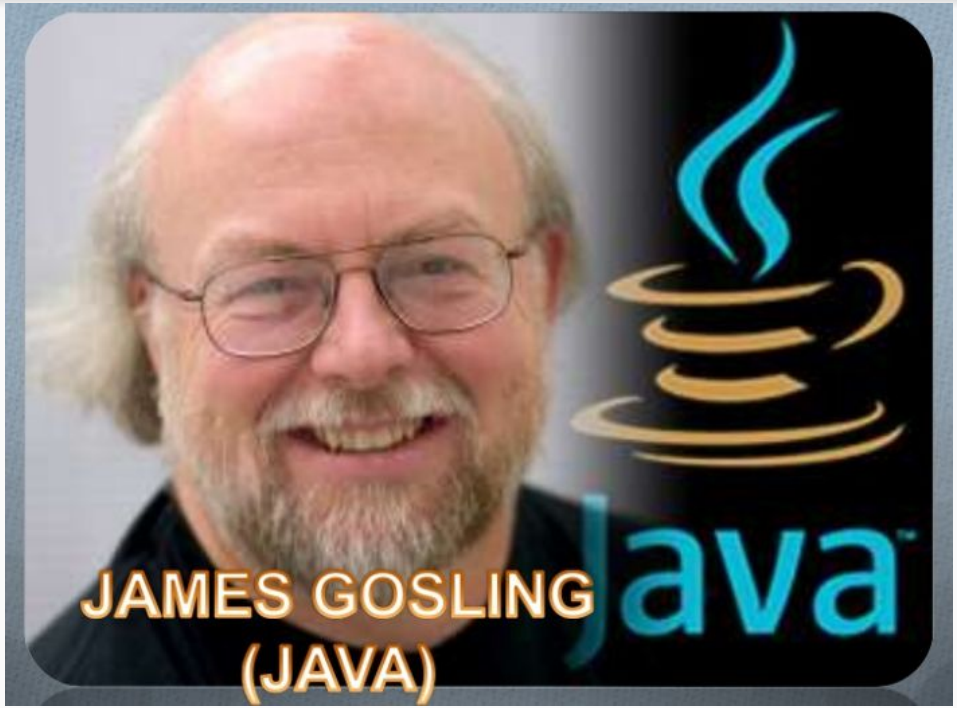
Java
O que é o Java?

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.

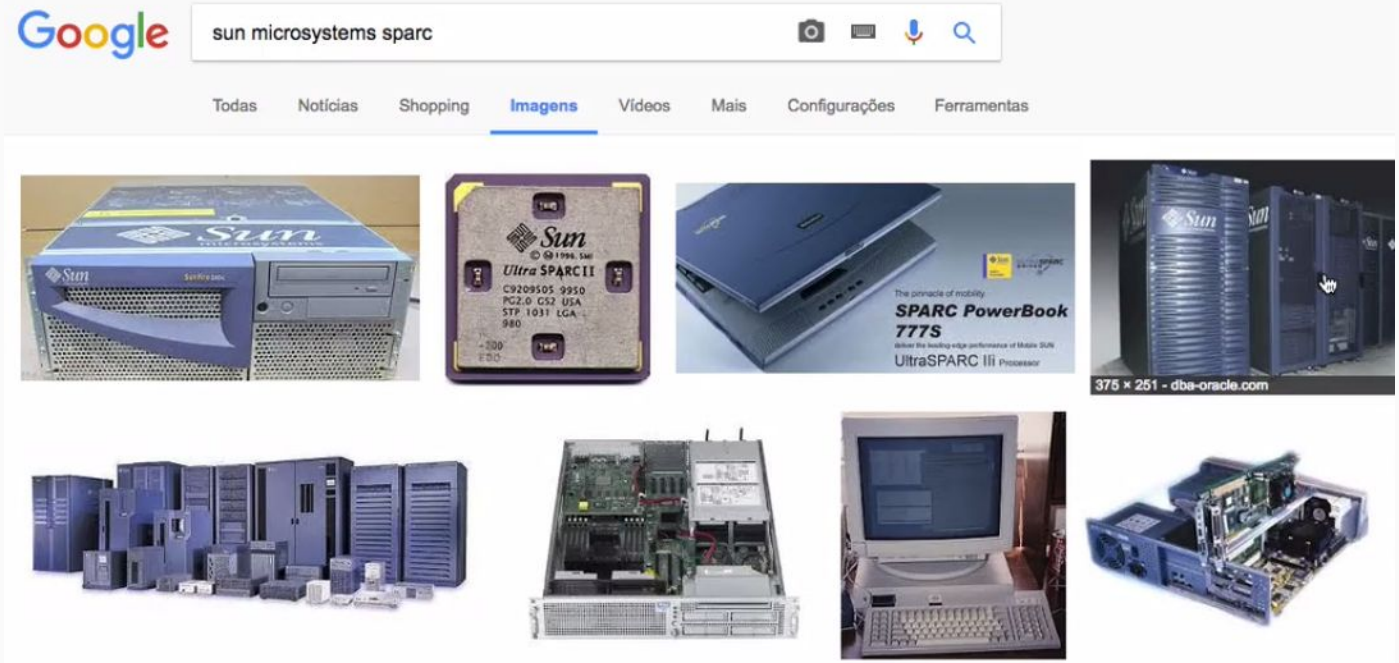
Eles possuíam um microcomputador, o Sun Microsystems SPARC, que hoje em dia já não aparecem em lugar algum, grandes servidores denominados "micro":

Java

Quem criou ?



Java
O que é o Java?



Java
Um pouco da história

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.

Java
Um pouco da história

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!

Java
Um pouco da história

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.

Java
Um pouco da história

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.

Java
Um pouco da história

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.

Java
Virtual Machine

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.

Java
Virtual Machine

Quem conhece a linguagem de *Assembly* talvez identifique a semelhança, mas este código não parece ser de fácil leitura e compreensão.

Veja um exemplo deste formato entendido pela *virtual machine* (JVM), o *bytecode* :

Java
Virtual Machine

```
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 VilaNova...
     11: putfield      #5
     // Field Onibus.linha:Ljava/lang/String;
     14: return
}
```

Java
Virtual Machine

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 = "Vila Nova-Estação";
  }
}
```

Java
Virtual Machine

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*.

Java
Virtual Machine

As principais características do conceito de Máquina Virtual Java são:

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

Java
Virtual Machine

As principais características do conceito de Máquina Virtual Java são:

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

Java
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**. :)

Java
O que aprendemos?

Nesta introdução já aprendemos vários assuntos fundamentais sobre Java. Falamos sobre as **principais características** da linguagem Java como:

1. orientado a objetos
2. parecido com C++
3. muitas bibliotecas e grande comunidade

Java
O que aprendemos?

Além disso, aprendemos:

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

Java
O que aprendemos?

Vimos também os **principais componentes** da plataforma Java que são:

1. Java Virtual Machine (JVM)
2. linguagem Java
3. Bibliotecas Java (API)

Java
JVM x JRE x JDK

O mundo Java é cheio de siglas com 3 ou 4 letras começando com **J**. Você já conhece duas famosas: o **JRE** e **JDK**. O primeiro é o ambiente de execução, o segundo são as ferramentas de desenvolvimento *junto* com o ambiente de execução. Simplificando podemos dizer:

JDK = JRE + ferramentas desenvolvimento

Java
JVM x JRE x JDK

Assim podemos simplificar e dizer:

JRE = JVM + bibliotecas

É importante entender que você não pode baixar a JVM apenas. Você sempre baixa o JRE que tem a JVM e as bibliotecas em conjunto.

Java
JVM x JRE x JDK

Existe uma terceira sigla, **JVM** (*Java Virtual Machine*), que também já usamos durante o curso. A responsabilidade da Java Virtual Machine é executar o Bytecode! Então qual é diferença entre JVM e JRE? Ambos executam o Bytecode, certo?

A resposta é simples: O JRE (o nosso ambiente de execução) contém a JVM, mas também possui um monte de bibliotecas embutidas. Ou seja, para rodar uma aplicação Java não basta ter apenas a JVM, também é preciso ter as bibliotecas.

Java
JVM x JRE x JDK

Assim podemos simplificar e dizer:

JRE = JVM + bibliotecas

É importante entender que você não pode baixar a JVM apenas. Você sempre baixa o JRE que tem a JVM e as bibliotecas em conjunto.

Java
Versões

Antes do processo de instalação e configuração, falaremos sobre versões, uma vez que é comum encontrarmos vários números e versões e ficarmos perdidos sem saber por onde iniciar no Java.

Apesar da última versão lançada ser o 9, lançada em 2017, a linguagem, surgida em 1995, teve mudanças consideráveis na versão 5, que saiu em 2004, e na 8, de 2014. Nelas, apareceram muitos recursos na linguagem, novos comandos, palavras-chave e conceitos.

Java
Versões

Estes tais de *Streams*, de *Templates Generics*, serão vistos durante o curso - há até um curso específico sobre estes novos recursos do Java 8. Nas versões 9 e 7, houve mudanças pequenas e pontuais, além de bibliotecas.

Então, não se preocupe, você pode, sim, focar na versão 8, pois você verá que muitas empresas grandes inclusive ainda não alcançaram esta versão (o que é uma pena).

Aqui, usaremos a versão *Neon* do *Eclipse*, mas existe uma versão mais recente, *Oxygen*, que está sendo trabalhada para dar suporte ao Java 9. Até o momento, não há versão oficial do *Eclipse* que dê suporte para a última versão disponível do Java.

Java
Versões

Todos os conceitos focados neste curso, são os mesmos para muitas versões da linguagem.

Ou seja, a dica é focar naquilo que é importante, que é o que estamos vendo aqui, e não nas versões mais recentes. A versão 10, provavelmente virá com muito menos novidades, já que as versões seguirão a tendência de serem lançadas mais rapidamente, não de 3 em 3 anos, e sim de 6 em 6 meses.gramação e nunca viu Java antes, indica-se a utilização das versões citadas neste curso.

Java
Versões

Recomendo que você siga os passos feitos neste curso, respeitando a instalação do Java 8 e do *Eclipse Neon*. No entanto, se você realmente quiser utilizar a versão mais recente de cada um deles, por sua própria conta e risco, vá em frente. É bem provável que você não encontre problemas!

Porém, se você é iniciante em programação e nunca viu Java antes, indica-se a utilização das versões citadas neste curso.

Java
Versões

Java
Instalação do JDK no Windows

Vamos instalar o Java e tudo aquilo de que precisamos para começarmos a trabalhar! Usando o Windows, iremos fazer uma busca no Google por "Java" para ver as opções fornecidas para download.

Um dos primeiros resultados mostrados é o java.com, com uma aparência um tanto ultrapassada, e o botão "Download Gratuito do Java", em português ou inglês. Indo por este caminho, você perceberá que baixará uma versão que costumamos usar para **rodar uma aplicação Java**.

Para nós, desenvolvedores, baixar isto não é o suficiente. A versão recomendada que aparece na página de download, no caso "Version 8 Update 121", é o que chamamos de ***Java Runtime Environment***, ou "ambiente de execução Java", que é necessário para **executar uma aplicação Java**.

Java
Instalação do JDK no Windows

Lembra da época dos *applets*, em que precisávamos instalar plugins e similares para serem rodados no browser, ou em aplicativos? É para estes casos que a instalação desse ambiente de execução serve, o tal do **JRE**, que vem com a *virtual machine* e as bibliotecas.

Como desenvolvedores, precisaremos do **JDK**, ou ***Java Development Kits***, o "kit para desenvolver aplicações Java".

Pesquisando no Google por "download java jdk" ou simplesmente "jdk", aí sim, cairemos em um link mais específico, como no da ***Oracle***, com diversas opções. Queremos a versão 8, ou outra mais recente.

Java
Instalação do JDK no Windows

Na descrição, lê-se "8u221", por exemplo, em que "u" indica "*update*", ou "atualização" em português, que tem a ver com correção de *bugs*. Nesta página, estão disponíveis para download o JDK, bem como o JRE, visto no link anterior.

A opção que queremos baixar trará, além da *virtual machine* e das bibliotecas, o compilador, que pegará o código Java e o transformará em um formato que ele entenderá. Vamos fazer o download do JDK de acordo com o sistema operacional - no Mac ou no Linux pode ser que já venha instalado, ou seja mais fácil de se baixar.

Java
Instalação do JDK no Windows

Neste caso, optamos por jdk-8u221-windows-x64.exe, não esquecendo de aceitar os termos da Oracle. Terminado o download, abriremos o arquivo executável, a ser salvo em um diretório apropriado seguindo os passos de instalação no modo *default*.

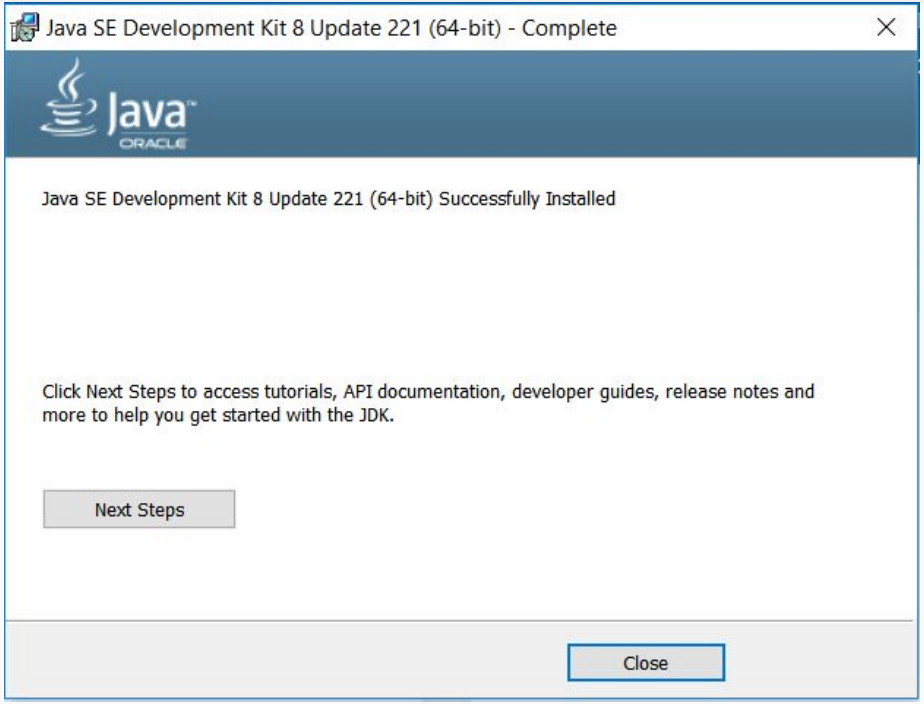
O JDK, o compilador, nada mais é do que uma versão menor daquilo que existe no site java.com, **mais** as ferramentas para o desenvolvimento de aplicações Java. Em seguida, continuaremos instalando o JRE, com a *virtual machine* e tudo o mais que é necessário para rodar esta aplicação.

Java
Instalação do JDK no Windows

Confirmaremos a instalação acessando o prompt do MS-DOS, que é algo muito similar ao terminal do Linux, Bash, Shell, e do Mac. O PowerShell da Microsoft hoje em dia é mais raro, mas ainda existe. Não se preocupe, muito em breve estaremos utilizando uma IDE, o Eclipse. Neste momento, porém, queremos enxergar o que está "por trás".

Pode-se pesquisar por "cmd" para acessar o prompt, uma tela preta em que digitaremos comandos, sendo o primeiro deles aquele que chamará o Java, `java`, seguido da tecla "Enter". Ele retorna uma mensagem dizendo para usarmos um `class`, porém ainda veremos sobre.

Java
Instalação do JDK no Windows



Java
Instalação do JDK no Windows

O comando que usaremos em seguida será javac, de *java compiler*, o compilador que pegará o código em Java e "traduzirá" para o que a *virtual machine* entende. Porém, ao digitarmos isso, obteremos o seguinte:

'javac' não é reconhecido como um comando interno ou externo, um programa operável ou um arquivo em lotes.

Você deve estar se perguntando o que aconteceu, já que o JDK foi instalado, e é verdade, ele deveria ser executável. O que acontece é que ele está em outro local, e por isto não está sendo encontrado.

Java
Instalação do JDK no Windows

Abrindo o explorador de arquivos, em C:, "Arquivos de Programas" contém a pasta "Java", que por sua vez possui dois diretórios referentes a JRE (onde se encontra a *virtual machine*) e JDK (onde está o compilador). Por *default* de instalação, a Oracle modifica os arquivos de configuração do Windows e deixa apenas o Java do JRE pronto para ser chamado em linha de comando.

Se você for usar o Java em linha de comando, como não é tão raro de acontecer, precisaremos do "jdk1.8.0_121", dentro do qual há, em "bin" (de "binário"), o arquivo javac.exe.

Java
Instalação do JDK no Windows

Vamos selecionar o caminho do diretório onde se encontra este executável, e copiá-lo por meio do atalho "Ctrl + C" e, no Painel de Controle, informaremos ao Windows para que toda vez que inserirmos algum comando, o caminho abaixo seja consultado também:

C:\Program Files\Java\jdk1.8.0_121\bin

No Painel de Controle, portanto, selecionaremos "Sistema > Configurações avançadas do sistema" e, na nova janela, clicaremos em "Variáveis de Ambiente...", utilizável por programas como se fossem variáveis globais do Windows.

Java
Instalação do JDK no Windows

Veremos no box abaixo de "Variáveis do sistema" o "Path". Clicaremos em "Editar..." e observaremos todos os seus componentes. Queremos incluir mais um diretório nele, portanto clicaremos em "Novo" e usaremos o atalho "Ctrl + V" para colar o caminho que copiamos anteriormente. Selecionaremos "OK" em todas as janelas que ficaram abertas.

Teremos que reabrir o Prompt de Comando, após o qual digitaremos javac, que desta vez funcionará corretamente! Quando se instala uma nova linguagem de programação, é comum que a variável de ambiente seja alterada para que não haja necessidade de criarmos arquivos ou entrarmos em diretórios específicos para trabalhar.

Java
Instalação do JDK no Windows

A partir deste momento, então, temos não só o Java, mas o JDK, o Kit de Desenvolvimento do Java, instalado e configurado no Windows, tanto para execução quanto para compilação de uma aplicação Java!

Java
Instalação do JDK no Windows

Abrindo o explorador de arquivos, em C:, "Arquivos de Programas" contém a pasta "Java", que por sua vez possui dois diretórios referentes a JRE (onde se encontra a *virtual machine*) e JDK (onde está o compilador). Por *default* de instalação, a Oracle modifica os arquivos de configuração do Windows e deixa apenas o Java do JRE pronto para ser chamado em linha de comando.

Se você for usar o Java em linha de comando, como não é tão raro de acontecer, precisaremos do "jdk1.8.0_121", dentro do qual há, em "bin" (de "binário"), o arquivo javac.exe.

Java
Instalação do JDK no Windows

Vamos selecionar o caminho do diretório onde se encontra este executável, e copiá-lo por meio do atalho "Ctrl + C" e, no Painel de Controle, informaremos ao Windows para que toda vez que inserirmos algum comando, o caminho abaixo seja consultado também:

C:\Program Files\Java\jdk1.8.0_121\bin

No Painel de Controle, portanto, selecionaremos "Sistema > Configurações avançadas do sistema" e, na nova janela, clicaremos em "Variáveis de Ambiente...", utilizável por programas como se fossem variáveis globais do Windows.