**01Versões**

**Transcrição**

Você deve estar muito ansioso para instalar o Java e o ambiente de programação para compilar e executar seu primeiro programa!

Mas antes de todo este processo de instalação e configuração, gostaria de falar 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.

Estes tais de ***Streams***, de ***Templates Generics***, serão vistos durante o curso - há até um [curso específico sobre estes novos recursos do Java 8](https://cursos.alura.com.br/course/java8-lambdas). 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.

Todos os conceitos focados neste curso, que envolvem Orientação a Objeto, uso da herança, polimorfismo e as principais bibliotecas, são os mesmos para muitas versões da linguagem.

Ou seja, a dica é focar naquilo que é importante, que é o que passaremos 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.

Minha recomendação é a de 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.

**02Instalação do JDK no Windows**

**Transcrição**

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](http://www.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**.

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.

Na descrição, lê-se "8u112", 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.

Neste caso, optaremos por jdk-8u121-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](http://www.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.

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.

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.Copiar código

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.

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. 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**Copiar código

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.

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.

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!

**03Mão na massa: instalando o JDK**

Vamos instalar o JDK! Para isso, siga os passos abaixo de acordo com seu sistema operacional. E, após instalado, prossiga para as configurações finais.

**Instalando o JDK no Linux**

1) No Linux, primeiramente vamos ter que executar o seguinte comando no terminal:

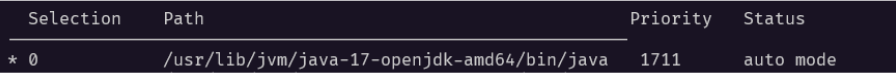
sudo apt updateCopiar código

2) Feito isso, agora vamos instalar a versão 17 do JDK, que hoje é a mais recente, para isso, vamos executar o seguinte comando no terminal:

sudo apt install openjdk-17-jdkCopiar código

Caso queira, você poderá instalar uma versão mais antiga do JDK, você só terá que mudar o número da versão que fica entre openjdk- e -jdk, por exemplo, você pode instalar a versão 11 com: openjdk-11-jdk e a versão 8 com: openjdk-8-jdk, mas recomendamos que você instale a versão mais recente, no caso, a versão 17.

3) Terminado o processo de instalação, chegou a hora de configurar a variável de ambiente **JAVA\_HOME**, que é utilizada para indicar o caminho de instalação do JDK. Em uma tradução literal seria o “lar do java” e ela será necessária para utilizarmos os recursos do JDK como o **javac**. Para isso, execute o comando sudo update-alternatives --config java que mostrará o caminho onde o JDK foi instalado. O resultado será parecido ou igual a esse:



Agora, copie esse caminho até /bin, por exemplo: /usr/lib/jvm/java-17-openjdk-amd64/bin e digite no seu terminal o comando: export JAVA\_HOME=, sendo que após o sinal de igual, sem deixar espaços, cole o caminho que você copiou da instalação do JDK. Por exemplo:

export JAVA\_HOME=/usr/lib/jvm/java-17-openjdk-amd64/bin

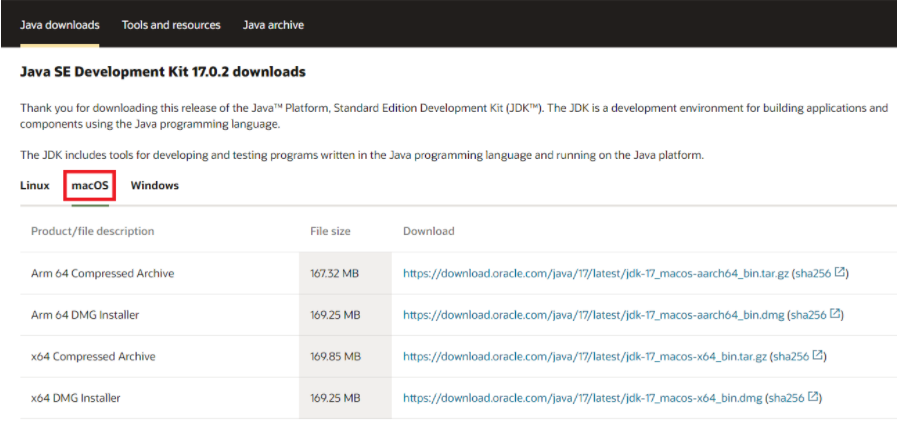
Pressione enter e pronto, seu JDK está instalado e configurado. Para testá-lo, digite os seguintes comandos no terminal:

javac -version

java -versionCopiar código

**Instalando o JDK no Mac**

No Mac você pode baixar a versão mais atual do JDK [aqui!](https://www.oracle.com/java/technologies/downloads/#jdk17-mac)



Escolha a opção de download compatível com a configuração do seu computador e faça a instalação a partir do instalador do JDK. Caso queira instalar uma versão mais antiga do JDK, descendo um pouco a página você encontrará opções de downloads para versões mais antigas.

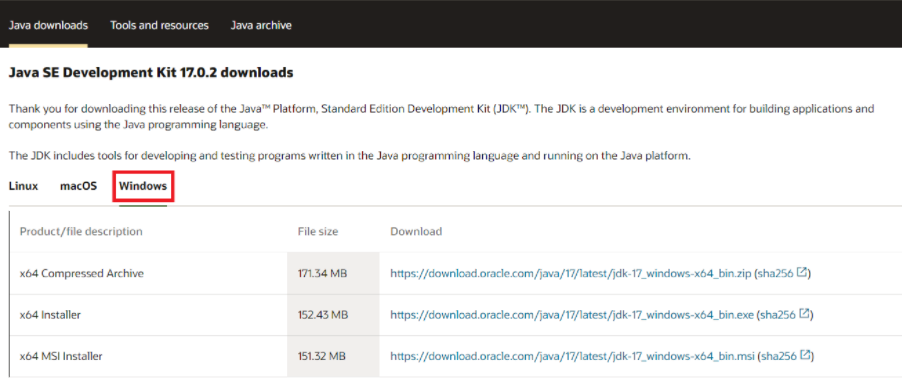
Após a instalação no seu Mac, teste seu Java com os seguintes comandos no terminal:

javac -version

java -version

**Instalando o JDK no Windows**

Por fim, para instalar no Windows, acesse [esse site](https://www.oracle.com/java/technologies/downloads/#jdk17-windows).

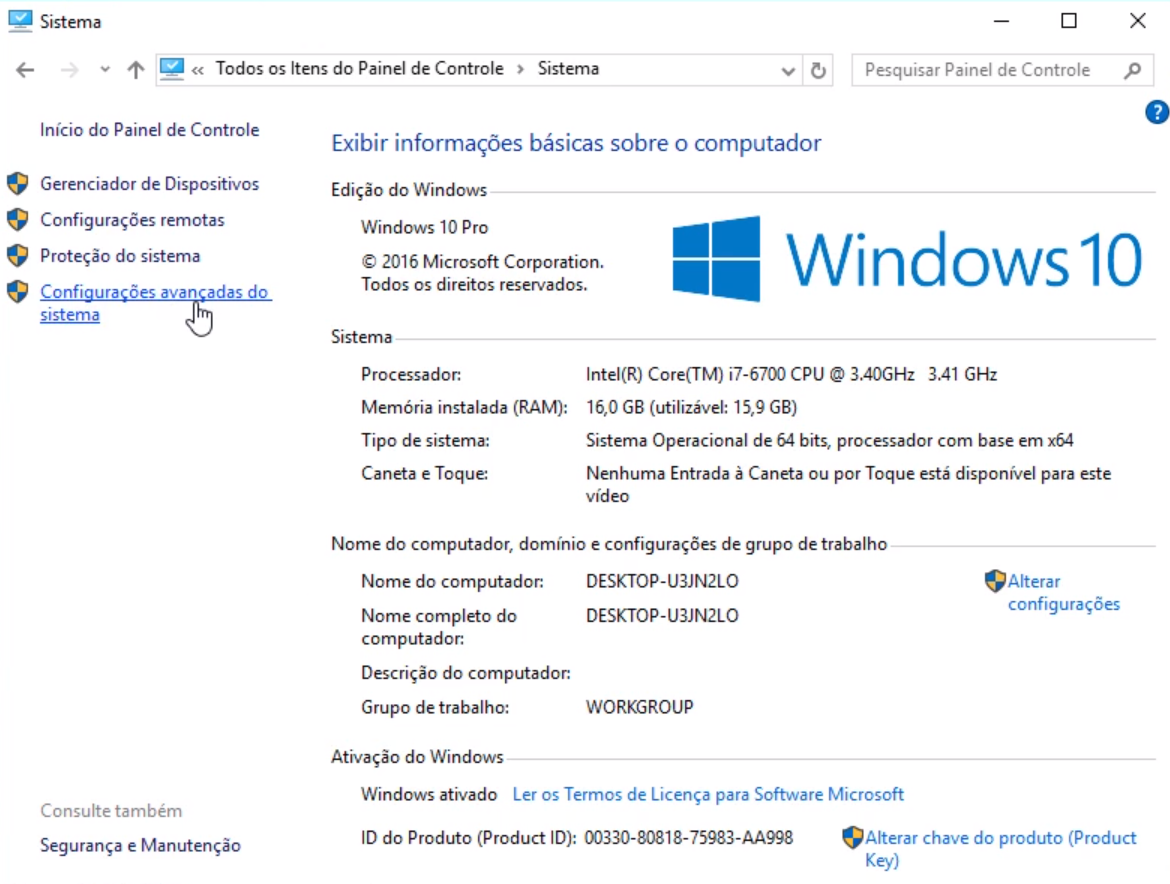


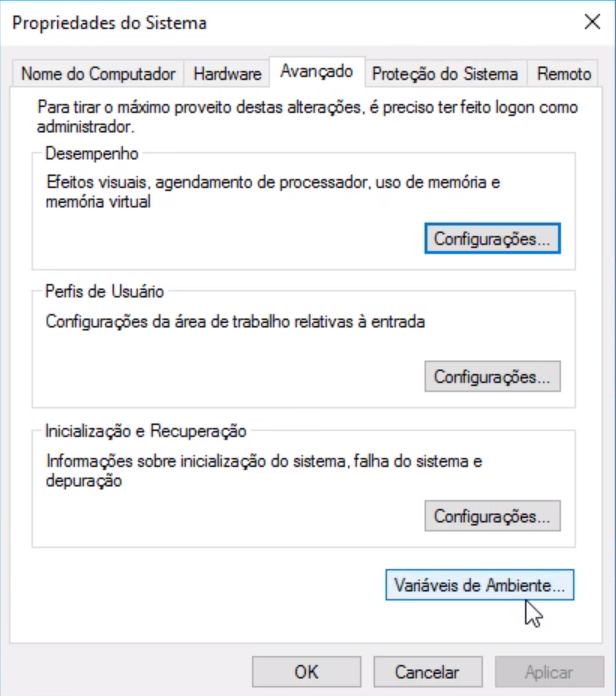
Escolha então o JDK (Java Development Kit) e, por fim, a versão do seu sistema operacional. Após baixado, execute o instalador e faça a instalação.

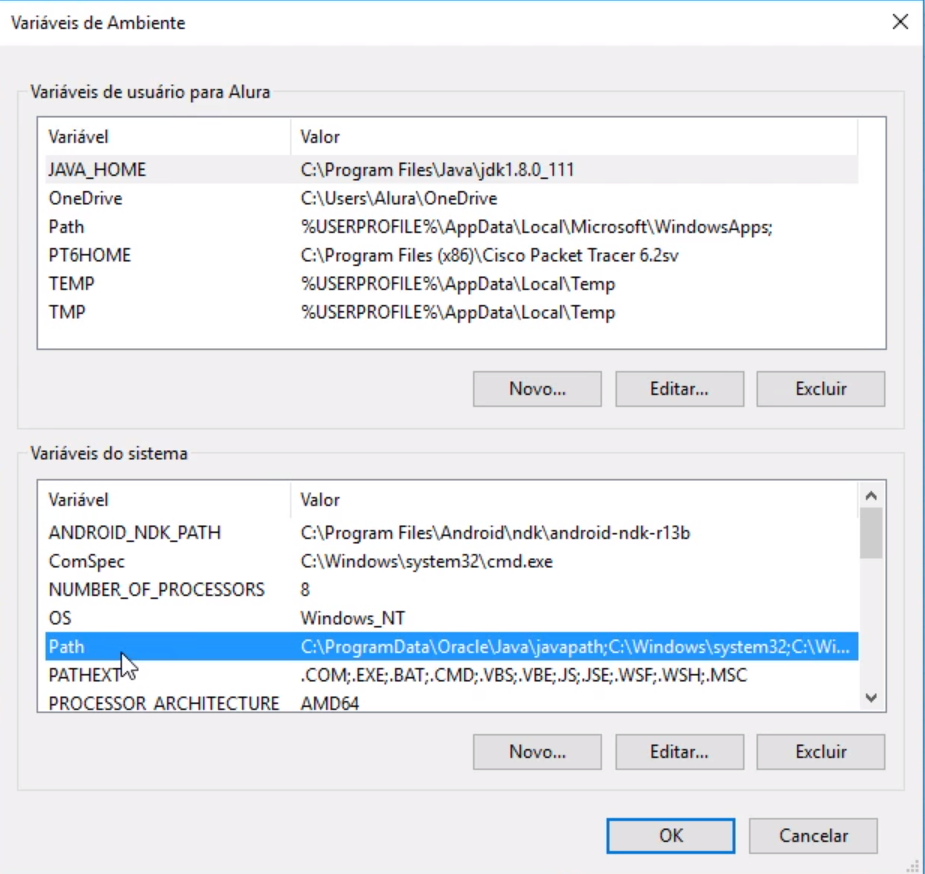
E, novamente, caso queira instalar uma versão mais antiga do JDK, descendo um pouco a página você encontrará opções de downloads para versões mais antigas.

**Configurações do PATH(Variável de Ambiente) no Windows**

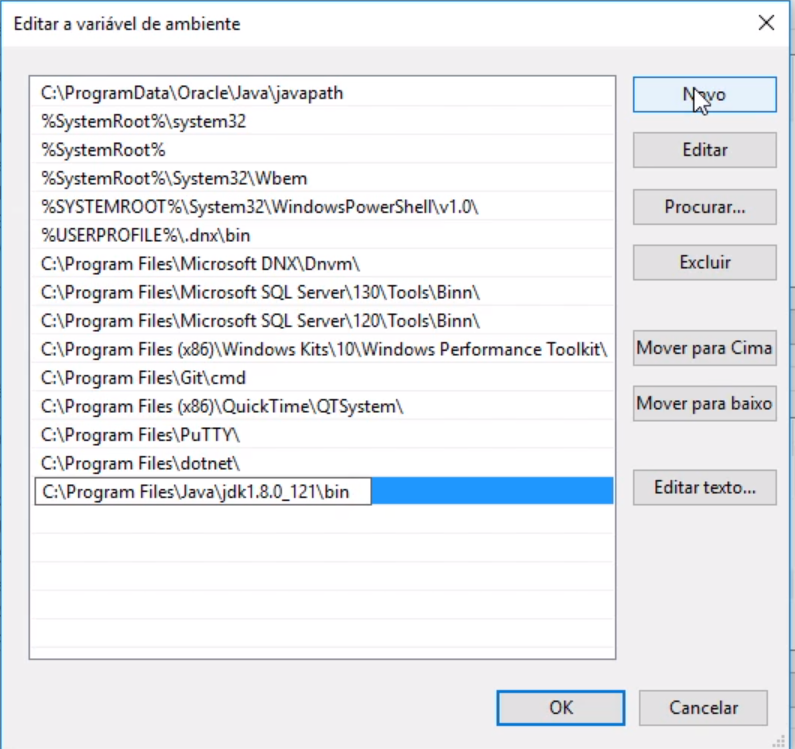
A partir daqui, caso queira assistir essas configurações em vídeo, no vídeo [**Instalação do JDK no Windows**](https://cursos.alura.com.br/course/java-primeiros-passos/task/29328) temos o passo a passo, ou acompanhe as configurações pelo tutorial abaixo:

1) Vá em painel de controle e procure por Sistema, após isso, clique em **Configurações avançadas do sistema**

2) Na aba "Avançado", clique em **variáveis de Ambiente...**

3) Na janela que foi aberta, na parte inferior da tela, selecione a variável de ambiente chamada **Path** e clique em **Editar**

4) Nessa nova janela, clique no botão **novo** e nessa linha que foi selecionada (última), coloque o caminho para seu diretório bin dentro da pasta jdk, que por sua vez, está dentro da pasta java



5) Salve a mudança clicando em OK e feche todas as janelas.

6) Após feito isso, feche seu prompt de comando e o abra novamente.

Teste os seguintes comandos:

java -version

javac -version

Tudo ocorreu como esperado?

**Opinião do instrutor**

Caso tenha dúvidas, pergunte no nosso fórum!

**04JRE vs JDK**

Quais das afirmações abaixo são verdadeiras referente a JDK e JRE?

Parte superior do formulário

* O JRE é o ambiente para executar uma aplicação Java.

Caso queira executar uma aplicação Java apenas, basta o JRE (Java Runtime Environment).

* Alternativa correta

O JDK é o ambiente para executar uma aplicação Java e possui várias ferramentas de desenvolvimento.

O JDK são as ferramentas de desenvolvimento (como o compilador) mas também tem JRE embutido!

* Alternativa correta

O JDK é o compilador para gerar o Bytecode.

* Alternativa correta

O JRE é o ambiente para gerar e executar o Bytecode.

Parte inferior do formulário

Parabéns, você acertou!

**05Para saber mais: JVM vs JRE vs 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 de desenvolvimentoCopiar código

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.

Assim podemos simplificar e dizer:

JRE = JVM + bibliotecasCopiar código

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.

**06Compile e rode seu primeiro programa Java**

**Transcrição**

Como falamos no início, em um primeiro contato, o código em Java pode ser complicado de ser escrito e compreendido. Às vezes precisamos escrever um pouco mais do que gostaríamos para fazer algo.

Antes de usarmos um IDE para lidarmos com o código, é legal que você o faça em um sistema bem simples, como o bloco de notas - outras opções são o TextPad, Atom, Visual Studio Code, Sublime, ou qualquer outro.

Nosso primeiro código Java será feito no editor de texto mais simples possível, em *plain text*. Faremos o "Olá mundo" para testarmos e vermos como funciona a compilação e execução de programas Java.

O Java veio da linguagem C na década de 90, então, não é tão simples quanto digitarmos print("olá mundo"). A linha que faz um print na tela, por exemplo, é

System.out.println("olá mundo");Copiar código

Nesta linguagem, toda instrução que damos sem as chaves necessita do ponto e vírgula (**;**). Todo código Java também precisa estar dentro de uma classe, que pode ser uma interface, um Enum. Neste caso, ele se insere na classe Programa.

Uma instrução como esta, com System.out.println(), precisa estar dentro de um método chamado main, que ainda não vimos, acompanhado de outros termos que também aprenderemos depois.

É muito comum o uso de public antes de class Programa, e embora isto não seja estritamente necessário no nosso caso, vamos colocá-lo para quando formos ler códigos de outros programadores e IDEs.

public class Programa {

public static void main(String[] args) {

System.out.println("olá mundo");

}

}Copiar código

No momento, focaremos na linha System.out.println("olá mundo");, que poderá ser considerado um comando apesar de não ser um, e mostrará algo na saída padrão, no caso o prompt do MS-DOS.

O menor programa Java seria similar ao código acima. Vamos tentar ver como funciona sua compilação e execução? Antes disso, salvaremos o arquivo nomeando-o com "Programa.java", em uma nova pasta denominada "java-codigo".

O nome do arquivo é muito importante - entenderemos melhor o motivo mais adiante, mas ele precisa ser o mesmo da class inserida no código.

No Prompt de Comando, digitaremos cd .. duas vezes, seguidos de "Enter", e dir, para a listagem de todos os diretórios. Depois, usaremos cd java-codigo para acessar o diretório, e em seguida digitaremos dir novamente.

Dica: é possível usar a tecla TAB para autocompletar palavras!

Ali, é listado um arquivo "Programa.java"! No Windows, há um comando chamado type (equivalente ao cat do terminal do Linux), o qual permite a visualização do conteúdo do arquivo. Neste caso, usaríamos type Programa.java.

A extensão .java não é entendida pela *virtual machine*, que entende o formato "meio máquina" de *Virtual Machine Java*, o ***bytecode***, um arquivo com extensão .class.

A seguir, usaremos o comando javac Programa.java, e daremos um "Enter", com o qual serão mostradas as mensagens de erro de compilação, fundamentais para o aprendizado.

Apesar de não entendermos o que é public class ou static void main ainda, sabemos que System.out.println() seguido de aspas e o conteúdo, irá mostrar uma mensagem.

Por meio de dir no prompt, você verá que há dois arquivos: "Programa.java" e "Programa.class", este último no formato binário, em *bytecode*. E para chamarmos a *virtual machine*, usaremos o comando java Programa, e veremos a impressão de "olá mundo". Trata-se da primeira execução do nosso programa Java!

Agora, veremos os principais erros e características deste código. O primeiro surge ao digitarmos java Programa.class, o que traz a seguinte mensagem de erro na execução do programa:

Erro: Não foi possível localizar nem carregar a classe principal Programa.classCopiar código

Isto acontece porque o programa não se chama "Programa.class", e sim simplesmente "Programa", apesar de estar contido no arquivo "Programa.class".

Outros erros mais comuns são os de compilação, como quando esquecemos de colocar o ponto e vírgula no fim da linha. Além disso, o Java possui palavras chave (*keywords*, ou palavras reservadas), dentre os quais utilizamos "public", "class", "static" e "void", que devem estar em letra minúscula, uma vez que o Java é ***case sensitive*** (reconhece o uso de letras maiúsculas ou minúsculas).

Em um ambiente mais complexo, veremos que isto ficará mais claro e fácil de ser trabalhado. É importante **praticar e não ter medo das mensagens de erro de compilação**.

As chaves abrem e fecham os blocos de códigos, indicando por exemplo que tudo aquilo que se encontra em public static void main pertence ao public class Programa, da mesma forma que System.out.println() pertence ao public static void mainvisível também por meio das indentações.

O Java possui outras particularidades, como o "Enter" e a barra de espaço serem opcionais; são convenções do código. Agora, o importante é escrever, entendendo o que está por trás do código, errar e fazer vários testes!

**07Entrada da aplicação**

A Ana está começando com desenvolvimento Java e já aprendeu que a entrada de uma aplicação é sempre a função (ou método) **main**. Porém, ela não se lembra qual era a definição correta (palavras chaves e parâmetros) dessa função/método:

class Programa {

??? main ???{

System.out.println("Você pode ajudar a Ana?");

}

}Copiar código

Qual é a definição correta?

Parte superior do formulário

* void main(String[] args)
* Alternativa correta

public static void main(String[])

* Alternativa correta

public static void main()

* Alternativa correta

public static void main(String[] args)

Correto! Ainda não sabemos o que significam todas essas palavras, mas fique tranquilo pois vai ficar claro para você. Nesse momento basta saber que a entrada de uma aplicação Java é sempre uma função/método public static void main(String[] args).

Parte inferior do formulário

Parabéns, você acertou!

Este conteúdo foi útil para o seu aprendizado?

Sim

Não muito

**08Sobre a compilação e execução**

Veja as afirmações abaixo sobre a compilação e execução de código Java:

**1 -**Durante a compilação acontece uma verificação sintática do código fonte.

**2 -**Na compilação e execução podem aparecer erros.

**3 -**A JVM lê e executa o Bytecode.

**4 -**O compilador gera o Bytecode caso não tenha nenhum erro sintático no código fonte.

Quais afirmações são verdadeiras?

Parte superior do formulário

* Apenas as afirmações 1, 2 e 4.
* Alternativa correta

Todas as afirmações são verdadeiras.

Correto!

* Alternativa correta

Apenas as afirmações 1, 2 e 3.

* Alternativa correta

Apenas as afirmações 2 e 3.

Parte inferior do formulário

Parabéns, você acertou!

Este conteúdo foi útil para o seu aprendizado?

Sim

Não muito

**09Compilar e executar**

O Pedro está trabalhando pela primeira vez com o sistema operacional Linux, mas está estranhando, pois, o computador não possui uma interface gráfica (só funciona na linha de comando!).

Ou seja, ele precisa compilar e executar um código Java abaixo na linha de comando, no entanto esqueceu os comandos :(

Arquivo Programa.java:

class Programa {

public static void main(String[] args) {

System.out.println("Funcionou!!");

}

}Copiar código

Quais comandos ele deve usar para compilar e rodar esse código Java?

Parte superior do formulário

* javac Programa.java
* java Programa

Correto! Repare que passamos a extensão do arquivo (.java) para o comando javac:

javac Programa.javaCopiar código

E para chamar a JVM usamos *apenas* o nome da classe (sem extensão):

java ProgramaCopiar código

* Alternativa correta
* javac Programa

java Programa

* Alternativa correta
* javac Programa.java

java Programa.class

Parte inferior do formulário

Parabéns, você acertou!

Este conteúdo foi útil para o seu aprendizado?

Sim

Não muito

**10Mão na massa: Escrevendo nosso primeiro código**

Chegou a hora de escrever seu código! Para isso, siga os passos abaixo:

1) Como visto no vídeo, o comando utilizado para imprimir algo na tela é o System.out.println, mas apenas ele não será suficiente para que o programa compile. Então até o momento temos:

System.out.println("ola mundo");Copiar código

2) Para que nosso código compile, primeiramente envolva-o com uma classe, ficando dessa forma:

public class Programa{

System.out.println("ola mundo");

}Copiar código

3) Nosso programa ainda não compila, para funcionar, coloque o método main, não se preocupe com o main agora, no decorrer do curso veremos os detalhes e entenderemos esse método. Teremos um código assim:

public class Programa{

public static void main(String[] args){

System.out.println("ola mundo");

}

}

Copiar código

4) Após feito isso, salve o arquivo como Programa.java em algum diretório. O nome deve ser exatamente igual ao nome da classe criada, inclusive as letras maiúsculas e minúsculas! Lembre-se que o Java é **case-sensitive**.

5) Agora, compile o código utilizando o compilador de Java da Oracle, chamado de javac. Ele nos dará o **bytecode**:

javac Programa.javaCopiar código

6) Após compilado, podemos ver que o **bytecode** foi gerado com o mesmo nome, mas com .class no final.

7) Execute o programa já compilado escrevendo:

java ProgramaCopiar código

Quando escrevemos apenas java, estamos invocando a máquina virtual para interpretar nosso programa.

Após feito isso, teremos como saída a frase ola mundo.

E aí, tudo certo? :)

**Opinião do instrutor**

Qualquer dúvida nos pergunte no fórum!

**11O que aprendemos?**

Nessa aula você escreveu o seu primeiro código Java e aprendemos:

* qual é a diferença entre JRE e JDK
* como compilar um código fonte Java na linha de comando (javac)
* como executar o Bytecode na linha de comando (java)
* um programa Java deve estar escrito dentro de uma classe (class)
* toda instrução Java deve ser finalizada com ;
* para abrir e fechar um bloco usaremos as chaves {}
* um programa Java possui uma entrada que é uma função (método) main
* para imprimir algo no console usamos a instrução System.out.println()

Este conteúdo foi útil para o seu aprendizado?

Sim

Não muito

**12Arquivos do projeto atual**

No link abaixo, você encontra o projeto até o momento atual do curso.

<https://github.com/alura-cursos/Curso-Java-parte-1-Primeiros-passos/archive/capitulo2.zip>

**Opinião do instrutor**

Lembre-se de usar o fórum em caso de dúvidas.