

Paulino Ng

# **Apostila de Netbeans para Novatos**

Brasil

2019, v-0.5.0



Paulino Ng

## **Apostila de Netbeans para Novatos**

Esta apostila visa mostrar aos novatos em programação como usar o Netbeans da Oracle como ambiente de desenvolvimento para programação Java.

Pang Co.

Treinamento de Computação

Brasil

2019, v-0.5.0

Paulino Ng

Apostila de Netbeans para Novatos/ Paulino Ng. – Brasil, 2019, v-0.5.0-  
45p. : il. (algumas color.) ; 30 cm.

Apostila – Pang Co.

Treinamento de Computação, 2019, v-0.5.0.

1. Netbeans. 2. Programação. 2.1. Java. 2.2. C/C++ I. Pang Co. II. Apostila Netbeans

*“Don’t Panic!*

*Don’t PANIC!*

*DON’T PANIC!*

*(The Hitchhikers’ Guide to the Galaxy, Douglas Adams)*

# Resumo

Esta apostila mostra rapidamente como começar a programar usando o Netbeans. Exemplos são dados tanto para a programação na linguagem Java, como em C/C++.

**Palavras-chave:** Netbeans. Programação. IDE. Java. C/C++.



# Lista de ilustrações

Figura 1 – Localização da instalação da JDK e da JRE sem espaços nos nomes de diretórios . . . . .	16
Figura 2 – Seleção de um projeto de Aplicação Java . . . . .	16
Figura 3 – Nome e localização da aplicação Java . . . . .	17
Figura 4 – Tela do projeto Alo com a tela de edição. . . . .	17
Figura 5 – Tela do projeto Alo com a tela de edição. . . . .	18
Figura 6 – Tela de saída da execução do programa Alo. . . . .	19
Figura 7 – Tela de edição do arquivo Alo.java com erro na linha 18. . . . .	19
Figura 8 – Arquivos para distribuição. . . . .	20
Figura 9 – Execução do arquivo JAR. . . . .	20
Figura 10 – Página para baixar o JDK SE . . . . .	21
Figura 11 – Localização da instalação da JDK e da JRE sem espaços nos nomes de diretórios . . . . .	22
Figura 12 – Configuração das variáveis de sistema no MS Windows . . . . .	23
Figura 13 – Acesso ao compilador Java e à variável JAVA_HOME . . . . .	23
Figura 14 – Edição do programa Alo no bloco de notas do MS Windows. . . . .	24
Figura 15 – Compilação e execução de Alo.java no Prompt do DOS. . . . .	24





# Lista de tabelas

Tabela 1 – Criação de Novo Projeto no NetBeans . . . . .	15
--	----



# Lista de abreviaturas e siglas

IDE	Integrated Development Environment, Ambiente de Desenvolvimento Integrado
JDK	Java Development Kit, Conjunto de Desenvolvimento Java
JRE	Java Runtime Environment, Ambiente de Execução (Tempo Real) do Java
JVM	Java Virtual Machine, Máquina Virtual Java
SE	Standard Edition, Edição Padrão
EE	Enterprise Edition, Edição Empresa
ME	Mobile Edition, Edição Portátil
C/C++	Linguagens de programação C e C++



# Sumário

<b>1</b>	<b>INTRODUÇÃO</b>	<b>13</b>
<b>1.1</b>	<b>Conceitos básicos sobre IDEs</b>	<b>13</b>
<b>1.2</b>	<b>Tutorial de NetBeans</b>	<b>15</b>
1.2.1	Configuração do Projeto	15
1.2.2	Adição de código ao arquivo fonte gerado	17
1.2.3	Compilação e execução da aplicação	18
1.2.4	Construção e distribuição da aplicação	19
<b>2</b>	<b>INSTALAÇÃO DO JDK</b>	<b>21</b>
<b>I</b>	<b>RESULTADOS</b>	<b>25</b>
<b>3</b>	<b>LECTUS LOBORTIS CONDIMENTUM</b>	<b>27</b>
<b>3.1</b>	<b>Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae</b>	<b>27</b>
<b>4</b>	<b>CONCLUSÃO</b>	<b>29</b>
	<b>REFERÊNCIAS</b>	<b>31</b>
	<b>APÊNDICES</b>	<b>33</b>
	<b>APÊNDICE A – QUISQUE LIBERO JUSTO</b>	<b>35</b>
	<b>APÊNDICE B – NULLAM ELEMENTUM URNA VEL IMPERDIET SODALES ELIT IPSUM PHARETRA LIGULA AC PRETIUM ANTE JUSTO A NULLA CURABI- TUR TRISTIQUE ARCU EU METUS</b>	<b>37</b>
	<b>ANEXOS</b>	<b>39</b>
	<b>ANEXO A – MORBI ULTRICES RUTRUM LOREM.</b>	<b>41</b>

ANEXO B – CRAS NON URNA SED FEUGIAT CUM SOCIIS NA- TOQUE PENATIBUS ET MAGNIS DIS PARTURI- ENT MONTES NASCETUR RIDICULUS MUS . . .	43
ANEXO C – FUSCE FACILISIS LACINIA DUI . . . . .	45

# 1 Introdução

A maioria dos estudantes de cursos superiores da área de computação do Brasil, a partir dos anos 2000 aprende na faculdade a programar em Java. Na proposta inicial da linguagem, o Java era uma linguagem orientada a objetos relativamente simples. A primeira versão pública de Java, a 1.0, é de 1995. O *kit de desenvolvimento Java*, o JDK, 1.0 saiu em janeiro de 1996. No início, a biblioteca contava padrão contava com menos de 1000 classes, nas últimas versões no início de 2019, esta biblioteca tem quase um milhão de classes.

No prólogo do livro ([BARNES; KÖLLING, 2009](#)), James Gosling, o chefe do projeto que criou o Java, fala da dificuldade encontrada pela filha dele numa disciplina de programação Java da faculdade. Ele percebeu que a dificuldade não era com a linguagem, mas com a IDE usada no curso dela. Uma IDE, ambiente integrado de desenvolvimento, é um software bastante complexo que integra diversos programas/ferramentas, como um editor inteligente, os compiladores, um depurador, um navegador de arquivos do projeto, gerenciador de projetos, gerenciador de versões, construtor de distribuição, ... No início, perde-se mais tempo aprendendo a usar corretamente uma IDE do que com a programação. Existem IDEs mais simples como a do BlueJ do livro citado. Vamos cobrir o NetBeans nesta apostila porque das IDEs usadas pelos profissionais que programam em Java, ela é uma das mais simples (com curva de aprendizado menor), poderosa e gratuita. As opções são:

1. IDEA da IntelliJ é, provavelmente, a IDE mais completa e poderosa, mas sua versão completa, é paga e a versão comunitária, perde muito das suas vantagens. A versão completa pode ser usada gratuitamente por estudante, participantes de projetos de código aberto e professores.
2. Eclipse foi durante muito tempo a principal IDE para desenvolvedores Java, mas sua curva de aprendizado é bastante longa. O Eclipse é extremamente poderoso implementando todas as *refatorações* propostas por Fowler.

## 1.1 Conceitos básicos sobre IDEs

Antes de começar a estudar a IDE, vamos rever alguns conceitos básicos. O Java é uma linguagem orientada a objetos, compilada e interpretada. Os aspectos de orientação a objetos interessam mais para projeto de software e podem ser vistos nos livros, ou cursos, sobre Engenharia de Software e Programação Orientada a Objetos. Algo que pode causar estranheza aos novatos é o fato do Java ser compilado e interpretado. Os programas em

Java definem classes e essas classes devem ser compiladas para serem executadas. Diferente de C/C++, Objective-C, Fortran, Pascal, etc., o código compilado não é executável na plataforma<sup>1</sup>. O código é compilado em Bytecodes que devem ser interpretados por uma JVM, Java Virtual Machine. A vantagem teórica desta estratégia é que o código compilado pode ser executado em qualquer plataforma com uma JVM compatível.

*Java é compilado*, isto significa, que para executar código Java é necessário compilar um código fonte. Uma IDE deve sempre incluir, as vezes, é só o que ela tem, um **editor de texto** para facilitar a edição do código fonte. O código fonte é um texto na linguagem de programação. O editor da IDE deve entender partes da sintaxe e gramática da linguagem e com isto auxiliar o programador. A maioria dos editores gráficos das IDEs conhece as palavras chaves da linguagem de programação e usa esquemas de cores para identificar mais facilmente estas palavras-chaves. Muitos editores de IDEs possuem também a capacidade de completar automaticamente um texto. À medida que o programador vai datilografando o código, o editor compara o texto com textos existentes na sua memória e propõe maneiras de completá-lo. O programador pode aceitar alguma proposta, ou simplesmente continuar datilografando e ignorar as propostas.

A IDE deve facilitar o acionamento das ferramentas de desenvolvimento como: *compilador, depulador, ...* Isto é, deve ser possível acionar estas ferramentas de dentro da IDE. Nos velhos tempos, quando não existiam IDEs, era necessário editar o texto (sem auxílios de linguagem). Salvá-lo, sair do editor, compilar na interface de linha, observar os erros e as linhas suspeitas anunciadas pelo compilador, voltar ao editor, encontrar as linhas com problemas, corrigir, salvar, sair do editor, recompilar e repetir este processos até o programa funcionar. Com as IDEs, a compilação do programa e sua execução podem ser feitas diretamente dentro da IDE sem ter de sair e rodar outros programas um de cada vez. Algumas IDEs já vem com algumas de suas ferramentas embutidas (o Visual Studio Code da Microsoft e o Xcode da Apple, por exemplo), outras vezes, a IDE depende da pré instalação destas ferramentas, caso do Netbeans e do Eclipse.

Algumas IDEs se prestam apenas para programar numa linguagem, outras podem ser usadas com diversas linguagens de programação. O Netbeans se presta para programar Java, aplicações JEE (Java Enterprise Edition), HTML, CSS, PHP, C e C++. Como Gosling identificou, o aprendizado de IDEs pode ser muito longo, assim, se precisamos aprender várias linguagens, é mais eficiente usar uma mesma IDE para todas. Observe que Eclipse, IDEA e outras IDEs podem ser usadas para várias linguagens de programação. O artigo [https://en.wikipedia.org/wiki/Comparison\\_of\\_integrated\\_development\\_environments](https://en.wikipedia.org/wiki/Comparison_of_integrated_development_environments) da wikipedia dá uma visão mais completa das IDEs e linguagens de programação.

---

<sup>1</sup> Uma plataforma é constituída pela dupla arquitetura do processador e sistema operacional. Assim, nos PCs, em geral, temos a plataforma Intel 64 bits e MS Windows, ou Intel 64 bits e Linux e nos computadores da Apple, processador Intel 64 bits e MacOS.



Esta apostila não vai tratar de todos os aspectos do NetBeans, existem muitos tutoriais na Internet para tanto, consulte a própria documentação nos sites: <https://netbeans.org/kb/index.html> e <http://netbeans.apache.org/kb/docs/java/index.html>.

Esta apostila começa com partes do tutorial ([NETBEANS.ORG, 2011](#)). Ele é bastante sucinto e a apostila expande alguns dos tópicos dele.

## 1.2 Tutorial de NetBeans

O tutorial, *quickstart*, é composto dos seguintes tópicos:

1. Configuração do projeto
2. Adição de código ao arquivo fonte gerado
3. Compilação e execução da aplicação
4. Construção e distribuição da aplicação

Ao concluir esta seção, você terá criado seu primeiro projeto com o NetBeans, uma aplicação em Java que escreve na saída padrão (a console) a mensagem: "Alô, Mamãe!".

Para fazer o que o tutorial propõe, é necessário que o software a seguir esteja instalado <sup>2</sup>, <sup>3</sup>:

Software requerido	Versão
IDE NetBeans ( <a href="https://netbeans.org/downloads/8.2/">https://netbeans.org/downloads/8.2/</a> )	8.2
JDK ( <a href="http://java.sun.com/javase/downloads/">http://java.sun.com/javase/downloads/</a> )	8

Tabela 1 – Criação de Novo Projeto no NetBeans

Fonte: ([NETBEANS.ORG, 2011](#))

### 1.2.1 Configuração do Projeto

Para criar um projeto na IDE:

1. Inicie a IDE NetBeans
2. Na IDE, selecione na barra de Menu: **File > New Project**, ou usando o teclado, **ctrl-shift-N**. A figura 1 mostra a seleção do Novo Projeto nos menus da IDE.

<sup>2</sup> Veja a página 21 onde a instalação destes softwares é apresentada.

<sup>3</sup> Nesta apostila não usaremos a última versão do NetBeans, a 10, que está entre os projetos incubados no Apache depois que a Oracle parou de manter a versão comunitária do NetBeans.

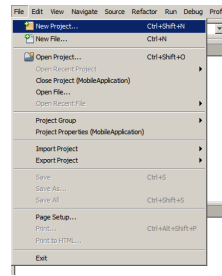


Figura 1 – Localização da instalação da JDK e da JRE sem espaços nos nomes de diretórios

Fonte: ([NETBEANS.ORG](http://NETBEANS.ORG), 2011)

3. No *wizard* para New Project, você verá a ficha para selecionar o tipo de projeto: Java e Java Application, figura 2. Acione o **Next**.

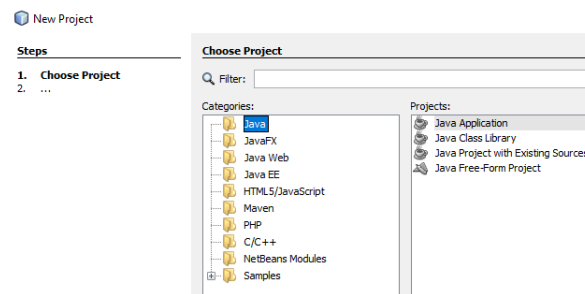


Figura 2 – Seleção de um projeto de Aplicação Java

Fonte: Snapshot do autor

4. Na ficha **Name and Location**, você deve selecionar um nome para a aplicação um diretório onde os arquivos do projeto devem ser colocados. A figura 3 mostra o projeto Alo cujos arquivos serão colocados no diretório

`C:\Users\Pang\Documents\NetBeansProjects.`

Observe que a caixa de **Create Main Class** está selecionada e tem o nome da classe `alo.Alo`. Isto significa que o NetBeans vai criar o arquivo `Alo.java` com um esqueleto da declaração da classe `Alo` em Java com um método `main()` nela. Clique no botão **Finish** para começar a editar o código Java.

5. O projeto foi criado e é o que está atualmente na área de trabalho do NetBeans. A figura 4 mostra os diferentes componentes da área de trabalho.

- A janela Projects contém uma vista em árvore dos componentes do projeto: arquivos fontes (**Source Packages**), bibliotecas (**Libraries**), etc.
- O editor de arquivos Fontes com o arquivo `Alo.java` aberto nele e

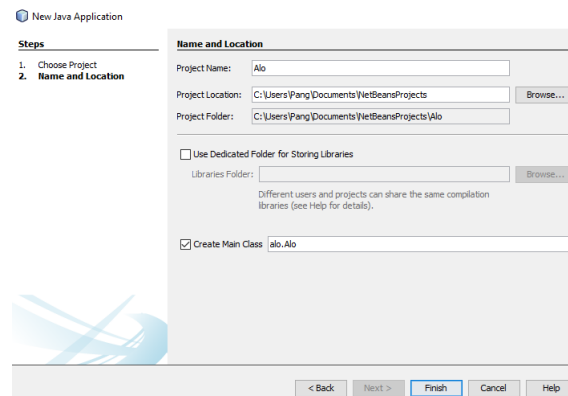


Figura 3 – Nome e localização da aplicação Java

Fonte: Snapshot do autor

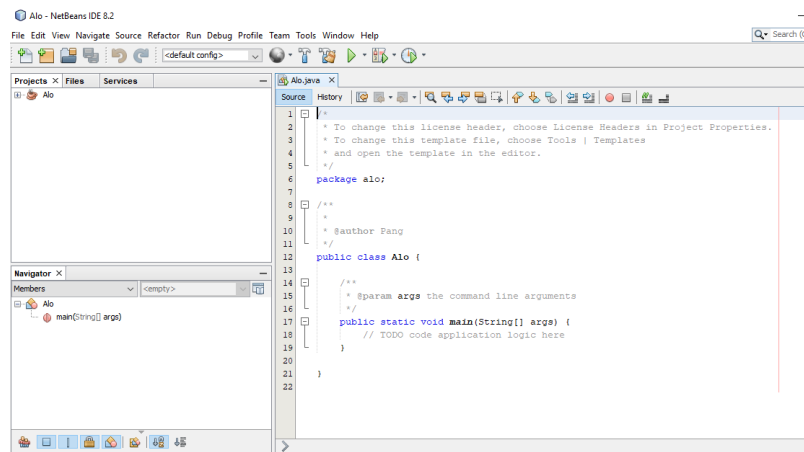


Figura 4 – Tela do projeto Alo com a tela de edição.

Fonte: Snapshot do autor

- A janela de Navegação que permite navegar rapidamente entre os elementos dentro da classe selecionada.


### 1.2.2 Adição de código ao arquivo fonte gerado

Por ter deixado a caixa **Create Main Class** selecionada, o *wizard* **New Project** criou um esqueleto da classe. Agora é só acrescentar a mensagem "Alô, Mamãe!" ao código. No lugar de:

```
// TODO code application logic here
```

coloque a linha:

```
System.out.println("Alô, Mamãe");
```

Salve as alterações com **File > Save**, ou **ctrl-S**, ou acione o botão(). A figura 5 mostra o editor com as mudanças;

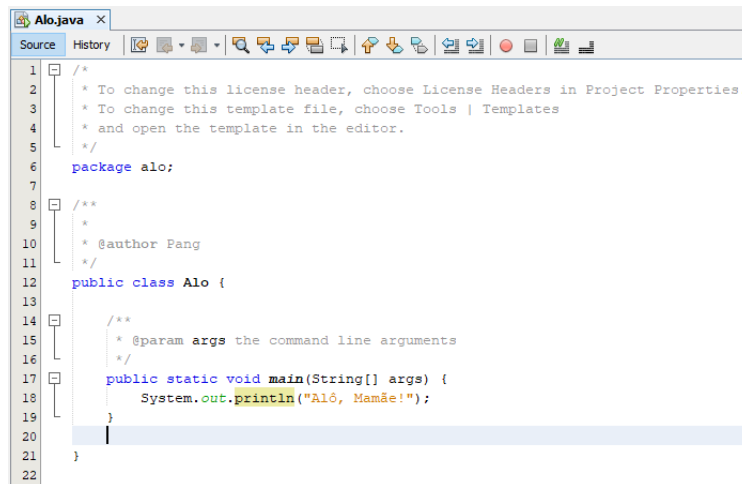



Figura 5 – Tela do projeto Alo com a tela de edição.

Fonte: Snapshot do autor

Observe que a cor do botão para salvar () muda da cor azul para cinza quando os arquivos estão salvos. Além disso, o título da aba do arquivo no editor está em negrito quando o arquivo não está salvo. Ele deixa de estar em negrito quando as mudanças estão salvas.

### 1.2.3 Compilação e execução da aplicação

Como a IDE tem a propriedade *Compile on Save*, não é necessário compilar o arquivo para executar o programa. Esta propriedade é configurável no menu do projeto (com o mouse sobre o ícone do projeto na janela de projetos, clique o botão da direita do mouse e selecione **Properties** para entrar na tela de configuração).

#### Para rodar o programa:

1. Selecione o menu **Run > Run Project**, ou acione **F6**.

Ao executar o programa, o NetBeans vai colocar uma janela da saída (ou de console, se o programa tiver entradas do usuário) abaixo da janela de edição com o conteúdo mostrado na figura 6.

O editor de código fonte entende a linguagem Java e enquanto o programa é digitado, ele vai verificando a sintaxe. Se ele encontra erros, ele coloca uma marca na numeração da linha de código onde está o erro, figura 7.

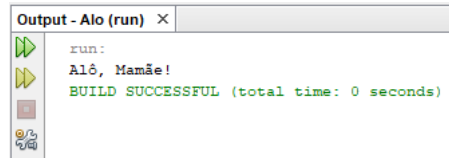


Figura 6 – Tela de saída da execução do programa Alo.

Fonte: Snapshot do autor

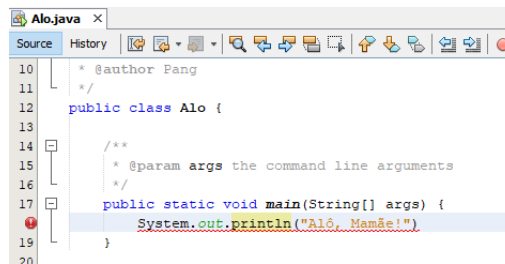


Figura 7 – Tela de edição do arquivo Alo.java com erro na linha 18.

Fonte: Snapshot do autor

### 1.2.4 Construção e distribuição da aplicação

Depois de escrever e testar sua aplicação, você pode usar o comando **Clean and Build** para construir sua aplicação para a distribuição. Programas Java são distribuídos em arquivos **JAR**. Ao executar o comando **Clean and Build**, a IDE executa um script que:

1. Remove os arquivos de compilações anteriores e a saída de outras construções e
2. Recompila a aplicação e constrói um arquivo **JAR** contendo os arquivos compilados.

Para construir a aplicação:

1. Selecione no menu **Run > Clean and Build Project**.

Pode-se ver as saídas da construção através da janela de **Files**, selecione a aba **Files** na janela **Projects** conforme mostra a figura 8. Expanda os nós **dist** e **Alo.jar**.

O arquivo compilado em bytecode **Alo.class** está dentro do nó **build/classes/alo**. O arquivo distribuível **JAR** contém, também, o arquivo **Alo.class** compactado como mostra a figura 8. O arquivo **README.TXT** mostra como usar o arquivo **JAR**. Como ilustrado na figura 9, onde o arquivo de distribuição **Alo.jar** é copiado para um outro diretório, **\Windows\Temp**, e executado com o interpretador Java.

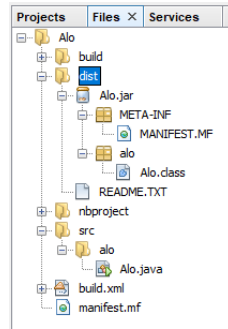


Figura 8 – Arquivos para distribuição.

Fonte: Snapshot do autor

```
C:\Windows\Temp>cd \Windows\Temp
C:\Windows\Temp>copy \Users\Pang\Documents\NetBeansProjects\Alo\dist\Alo.jar
Overwrite C:\Windows\Temp\Alo.jar? (Yes/No/All): yes
1 file(s) copied.
C:\Windows\Temp>java -jar Alo.jar
Alô, Mamãe!
```

Figura 9 – Execução do arquivo JAR.

Fonte: Snapshot do autor

A seguir, a instalação dos softwares é detalhada e explicações sobre o Java são apresentadas junto com os exemplos. Para concluir esta breve introdução à programação java, veremos como desenvolver e distribuir uma pequena aplicação.

## 2 Instalação do JDK

Para poder programar em Java, o computador do aluno precisa ter instalado o JDK-SE, Java Development Kit - Standard Edition. O JDK inclui o programa compilador (javac), a biblioteca padrão chamada de Edição Padrão (SE - Standard Edition), esta é a biblioteca para desenvolvimento de aplicações para computadores de mesa, desktops, e computadores portáteis, laptops e notebooks. As outras bibliotecas são a EE, Enterprise Edition, para o desenvolvimento de aplicações para empresas e a ME, Mobile Edition, para o desenvolvimento de aplicações para dispositivos móveis. O JDK inclui também a JRE, Java Runtime Environment, que seu computador já deve ter para rodar aplicações Java, como o programa da receita federal. É importante que a JRE seja compatível com o compilador e as outras bibliotecas.

No momento da escrita desta apostila, existem diversos JDK SE, versões que vão da 6 a 11. Como uma extensão da disciplina, vai levar ao uso do Java EE, a versão mais avançada do Java EE é a 8. Assim, será exemplificado a seguir a instalação do JDK 8. Que pode ser baixado de: <https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>.

<https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

JDK 8u201 [checksum](#)  
JDK 8u202 [checksum](#)

**Java SE Development Kit 8u201**

You must accept the [Oracle Binary Code License Agreement for Java SE](#) to download this software.

Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.

Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	72.98 MB	<a href="#">jdk-8u201-linux-arm32-vfp-hflt.tar.gz</a>
Linux ARM 64 Hard Float ABI	69.92 MB	<a href="#">jdk-8u201-linux-arm64-vfp-hflt.tar.gz</a>
Linux x86	170.98 MB	<a href="#">jdk-8u201-linux-i586.rpm</a>
Linux x86	185.77 MB	<a href="#">jdk-8u201-linux-i586.tar.gz</a>
Linux x64	168.05 MB	<a href="#">jdk-8u201-linux-x64.rpm</a>
Linux x64	182.93 MB	<a href="#">jdk-8u201-linux-x64.tar.gz</a>
Mac OS X x64	245.92 MB	<a href="#">jdk-8u201-macosx-x64.dmg</a>
Solaris SPARC 64-bit (SVR4 package)	125.33 MB	<a href="#">jdk-8u201-solaris-sparcv9.tar.Z</a>
Solaris SPARC 64-bit	88.31 MB	<a href="#">jdk-8u201-solaris-sparcv9.tar.gz</a>
Solaris x64 (SVR4 package)	133.99 MB	<a href="#">jdk-8u201-solaris-x64.tar.Z</a>
Solaris x64	92.16 MB	<a href="#">jdk-8u201-solaris-x64.tar.gz</a>
Windows x86	197.66 MB	<a href="#">jdk-8u201-windows-i586.exe</a>
Windows x64	207.46 MB	<a href="#">jdk-8u201-windows-x64.exe</a>

Figura 10 – Página para baixar o JDK SE

Fonte: Snapshot do autor

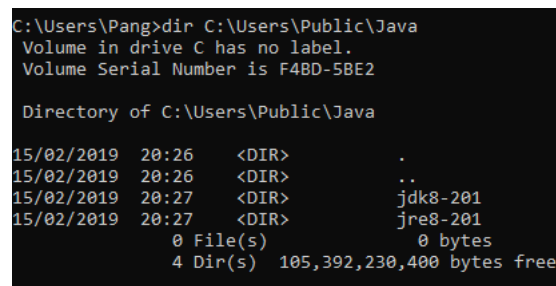
Baixe o arquivo adequado para o seu sistema, a figura 10 ilustra a página de download. Observe que para a instalação do JDK, é necessário usar o usuário com direitos administrativos do seu computador. Nesta apostila, as imagens mostram a execução dos

programas dentro de um MS Windows 10, em língua inglesa. A instalação do JDK passa por duas etapas: na primeira é instalado o JDK e na segunda é instalado o JRE.

Um problema com o instalador no MS Windows é sua insistência em instalar os programas no diretório:

`C:\Program Files\Java`

Em português, o diretório é o `C:\Arquivo De Programas\Java`. Ambos os nomes de diretórios possuem espaços no nome. Isto provoca problemas desnecessários para programas que usam o JDK e o JRE. Para evitar esta situação, na instalação, faça os programas serem instalados num diretório tal que não haja espaço no nome dos diretórios. Por exemplo, a figura 11 mostra que a instalação foi feita nos diretórios `C:\User\Public\Java\jdk8-201` e `C:\User\Public\Java\jre8-201`.



```

C:\Users\Pang>dir C:\Users\Public\Java
Volume in drive C has no label.
Volume Serial Number is F4BD-5BE2

Directory of C:\Users\Public\Java

15/02/2019  20:26    <DIR>          .
15/02/2019  20:26    <DIR>          ..
15/02/2019  20:27    <DIR>          jdk8-201
15/02/2019  20:27    <DIR>          jre8-201
               0 File(s)              0 bytes
               4 Dir(s) 105,392,230,400 bytes free
  
```

Figura 11 – Localização da instalação da JDK e da JRE sem espaços nos nomes de diretórios

Fonte: Snapshot do autor

A existência de espaços nos nomes provocam problemas principalmente com o *Tomcat* e outros servidores que usam o Java. Para estes programas funcionarem, é necessário configurar uma variável de ambiente, a `JAVA_HOME` que deve dar o diretório de base do JDK.

A figura 12 mostra o acesso à configuração das variáveis de sistema. Para chegar a estas telas, acionar as teclas janela+pause (aperte simultaneamente na tecla windows e na tecla pause) dá acesso ao painel de controle do Sistema, selecione o **Ajuste de Sistema Avançado** e você terá a tela da figura 12. Crie uma **Nova** variável de sistema com o nome `JAVA_HOME`. No valor a ser atribuído à variável, coloque o diretório onde foi instalado o JDK. Para completar, configure a variável `PATH`. Selecione-a e clique em **Editar**. Na tela de edição, acrescente:

`%JAVA_HOME%\bin`



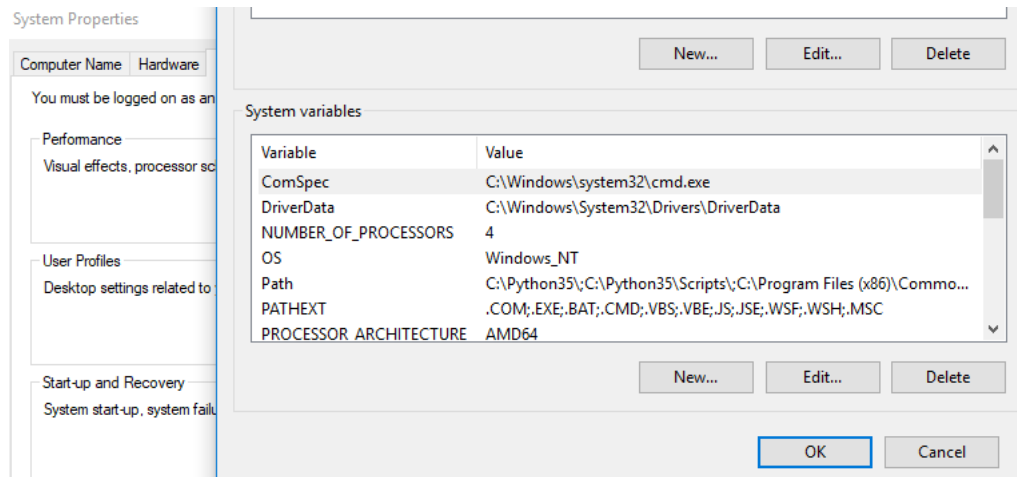


Figura 12 – Configuração das variáveis de sistema no MS Windows

Fonte: Snapshot do autor

Dê OK para as mudanças. Feche a janela de Prompt do DOS e abra uma nova. Você deve ser capaz agora de acionar o compilador na linha de comando como mostrado na figura 13.

```
C:\Users\Pang>javac -version
javac 1.8.0_201

C:\Users\Pang>echo %JAVA_HOME%
C:\Users\Public\Java\jdk8-201
```

Figura 13 – Acesso ao compilador Java e à variável JAVA\_HOME

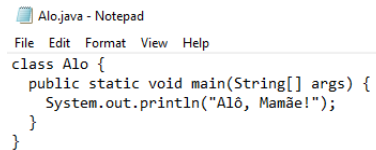
Fonte: Snapshot do autor

Para testar sua instalação, escreva o programa abaixo no Bloco de Notas (Notepad) do MS Windows. Veja a figura 14. Salve o arquivo Alo.java numa pasta na qual você pode escrever, no caso, o arquivo foi salvo em C:\Users\Pang\Alo.java.

```
class Alo {
    public static void main(String[] args) {
        System.out.println("Alô, Mamãe!");
    }
}
```

Compile e execute como mostrado na figura 15.

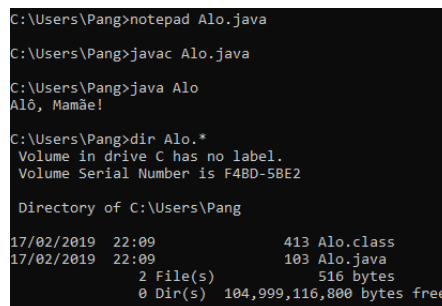
O comando `javac Alo.java` compila o arquivo `Alo.java` e gera o arquivo compilado em bytecode `Alo.class`. Para executar o bytecode, é necessário usar o programa



```
class Alo {  
    public static void main(String[] args) {  
        System.out.println("Alô, Mamãe!");  
    }  
}
```

Figura 14 – Edição do programa Alo no bloco de notas do MS Windows.

Fonte: Snapshot do autor



```
C:\Users\Pang>notepad Alo.java  
C:\Users\Pang>javac Alo.java  
C:\Users\Pang>java Alo  
Alô, Mamãe!  
  
C:\Users\Pang>dir Alo.*  
Volume in drive C has no label.  
Volume Serial Number is F48D-5BE2  
  
Directory of C:\Users\Pang  
  
17/02/2019  22:09                413 Alo.class  
17/02/2019  22:09                103 Alo.java  
                2 File(s)            516 bytes  
                0 Dir(s)  104,999,116,800 bytes free
```

Figura 15 – Compilação e execução de Alo.java no Prompt do DOS.

Fonte: Snapshot do autor

`java`, que implementa a JVM, com o `Alo` como argumento. Observe que não se escreve a extensão `.class`. A saída do programa é a mensagem "Alô, Mamãe!". O comando `dir Alo.*` lista os 2 arquivos: o código fonte, `Alo.java` e o compilado em *bytecode*, `Alo.class`. Podemos comparar os 2 programas Alo que foram escritos até agora e as ferramentas utilizadas. Ambos programas das páginas 18 e 23 fazem a mesma coisa, imprimem a mensagem "Alô, Mamãe!" na tela de saída.

A linguagem Java é orientada a objetos, isto significa que escrevemos programas Java, escrevendo classes. Uma aplicação Java é um programa que é executado como mostrado na figura 15. Para isto, o arquivo `.class` corresponde a uma classe definida num arquivo fonte com um método estático de nome `main`.

Parte I

Resultados



### 3 Lectus lobortis condimentum

- 3.1 Vestibulum ante ipsum primis in faucibus orci luctus et ultrices  
posuere cubilia Curae



## 4 Conclusão





## Referências

BARNES, D. J.; KÖLLING, M. *Programação orientada a objetos com JAVA - Uma introdução prática usando o BlueJ*. 4. ed. São Paulo: Prentice-Hall Brasil, 2009. ISBN 9788576051879. Citado na página 13.

NETBEANS.ORG. *NetBeans IDE Java Quick Start Tutorial*. Oracle, 2011. Disponível em: <<https://netbeans.org/kb/docs/java/quickstart.html>>. Acesso em: 15 fev 2019. Citado 2 vezes nas páginas 15 e 16.



## Apêndices



## APÊNDICE A – Quisque libero justo



APÊNDICE B – Nullam elementum urna vel  
imperdiet sodales elit ipsum pharetra ligula ac  
pretium ante justo a nulla curabitur tristique  
arcu eu metus





## Anexos



ANEXO A – Morbi ultrices rutrum lorem.



ANEXO B – Cras non urna sed feugiat cum  
sociis natoque penatibus et magnis dis  
parturient montes nascetur ridiculus mus



## ANEXO C – Fusce facilisis lacinia dui