



# Capacitação Java Atos e UFN

Fabício Tonetto Londero,  
Lucas Alberto Schlestein.



# Apresentação do curso

## Objetivos:

“Possibilitar a utilização de conceitos avançados na programação de computadores, oferecendo conhecimentos em linguagens, técnicas e ferramentas Enterprise.”



# Apresentação do curso

## Conteúdo Programático:

- Java Virtual Machine.
- Introdução à Arquitetura.
- Orientação a Objetos.
- Gerenciamento de Projetos.
- Banco de Dados.
- JDBC.
- JPA/Hibernate.



# Apresentação do curso

## Conteúdo Programático:

- Servlet.
- JSF.
- Facelets.
- Spring Security.
- Gráficos e Relatórios.
- Webservices.
- Application Server: Tomcat, Wildfly.



# Apresentação do curso

## Conteúdo Programático:

- Logging, Debugging, Profiling.
- Spring Boot 2 Rest API with Swagger2.
- Spring Batch.
- Introdução Angular.
- Introdução ao Docker.
- Introdução ao Desenvolvimento Mobile.



## Caracterização geral da metodologia de ensino

O curso será desenvolvido com aulas interativas utilizando recursos que permitam uma visão abrangente sobre o desenvolvimento avançado de software. Serão apresentados e utilizados conceitos, tecnologias e ferramentas existentes na área, com o objetivo de que o aluno consiga propor soluções a partir das habilidades e competências desenvolvidas no curso.



## Caracterização geral da metodologia de ensino

A metodologia de trabalho contará com atividades práticas, estudos de casos, apresentações e, principalmente, metodologia ativa (com o processo de aprendizagem baseado em projetos). Além disso, o curso conta com o apoio do Ambiente Virtual de Aprendizagem (AVA) e de ferramenta para videoconferência utilizada para a realização dos encontros virtuais diários.



# Critérios de avaliação da aprendizagem

Os alunos serão avaliados pela *participação* nos encontros, *assiduidade* e pelo desenvolvimento dos *exercícios* e *projetos* solicitados ao longo do curso.





# Bibliografia

## Básica:

- DEITEL, H. M.; DEITEL. P. J. Como Programar em Java. PRENTICE HALL BRASIL, 2017.
- HORSTMANN, Cay S.; CORNELL, Gary. Core Java. PRENTICE HALL BRASIL, 2009.
- SANTOS, Rafael; Introdução à Programação Orientada a Objetos usando Java, Editora Campus, 2003.



# Bibliografia

## Complementar:

- DALL'OGGIO, Pablo. PHP: programando com orientação a objetos. São Paulo, SP : Novatec, 2007. 574 p.
- DEITEL, H. M.; DEITEL. P. J. Java TM: como programar. Porto Alegre, RS : Bookmann, 2001.
- ECKEL, Bruce. Thinking in Java. 3.ed. London: Prentice Hall, 2003.
- GOODRICH, Michael T.; TAMASSIA, Roberto. Estruturas de dados e algoritmos em java. Porto Alegre, RS : Bookman, 2002. 584 p.



# Bibliografia

## Complementar:

- HORSTMANN, Cay S.; CORNELL, Gary. Core Java 2. São Paulo, SP : Makron Books, 2001
- MECENAS, Ivan. Java 2: fundamentos, swing e JDBC. Rio de Janeiro, RJ : Alta Books, c2003. 378 p.
- METSKER, Steven John.. Padrões de projeto em Java. Porto Alegre: Bookman, 2004. 407p.

# Java Virtual Machine



O Java é uma linguagem de computador muito utilizada atualmente em diversos Sistemas Operacionais (quando falamos em sistemas operacionais, não está relacionado apenas com computadores pessoais, mas também com tablets, equipamentos de Blu-Ray e etc.) devido a sua facilidade de ter o código portado. Ouvimos sempre por aí que o Java “roda em qualquer lugar” (*write once, run everywhere*), ele consegue “rodar” em qualquer plataforma devido a sua Máquina Virtual.

# Java Virtual Machine



## História:

Em 1991, foi criada uma linguagem a partir do Green Project. James Gosling, Mike Sheridan e Patrick Naughton eram os idealizadores. O objetivo desse projeto não era criar uma nova linguagem de programação, mas sim, proporcionar a integração de computadores com outros tipos de equipamentos, como hardwares ou circuitos elétricos pré-fabricados.



# Java Virtual Machine



## História:

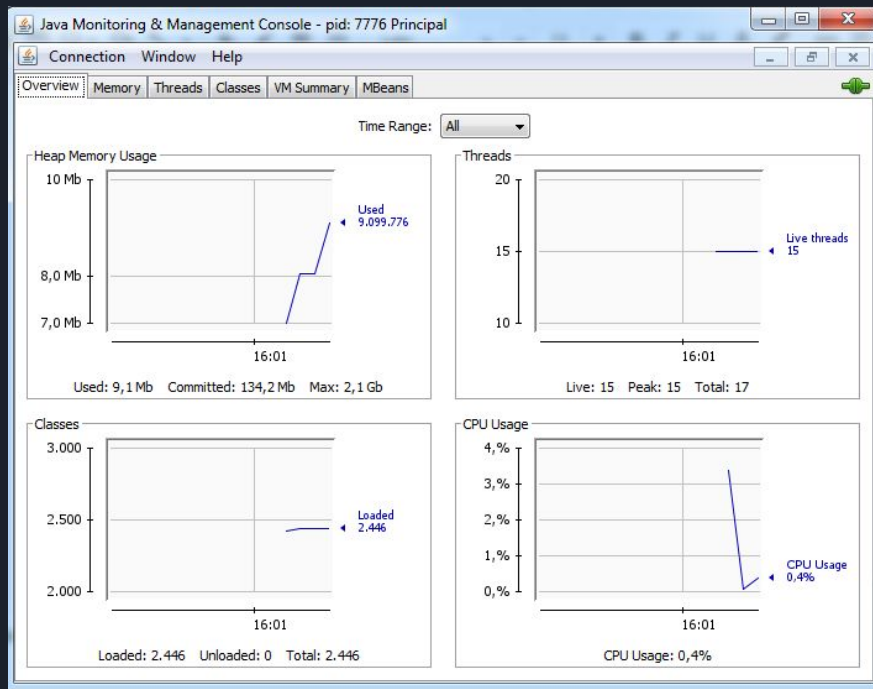
Em 1995, a empresa Sun Microsystems introduziu a plataforma JAVA (batizada posteriormente) no mercado. Ela é constituída pela linguagem de programação Java, sua máquina virtual e muitas APIs de controle e desenvolvimento.

# Java Virtual Machine



Antes de falarmos sobre a **JVM** (Java Virtual Machine), vamos pensar no conceito de máquina virtual. Uma máquina virtual é um software que simula uma máquina física e consegue executar vários programas, gerenciar processos, memória e arquivos. Resumindo, ele constitui de uma plataforma, onde a memória, o processador e seus outros recursos, são totalmente virtuais, não dependendo de hardwares.

# Java Virtual Machine JConsole







# Java Virtual Machine



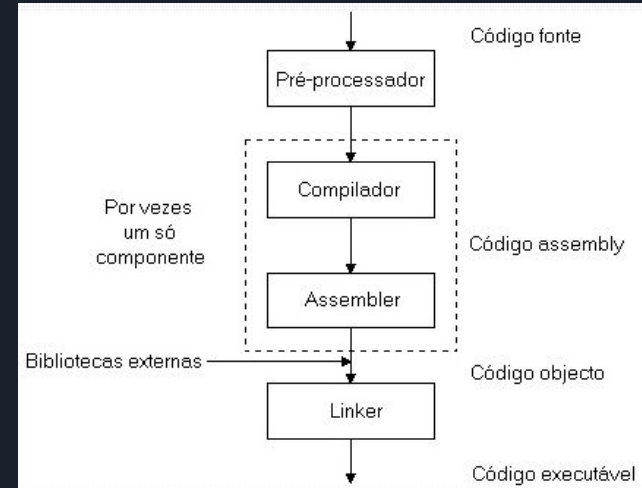
Em uma linguagem como a C, o código é compilado para uma máquina específica, ou seja, quando seu código é compilado, ele poderá ser executado apenas naquele sistema operacional. Para executarmos este código em outro Sistema Operacional, temos que ajustar as bibliotecas de acordo com as necessidades e recompilar.

obs: Compilação é o ato / processo de traduzir um programa feito em uma linguagem de alto nível para uma linguagem de máquina, para que suas instruções sejam executadas pelo processador, ou seja, criar o executável de um programa escrito em uma linguagem de alto nível.

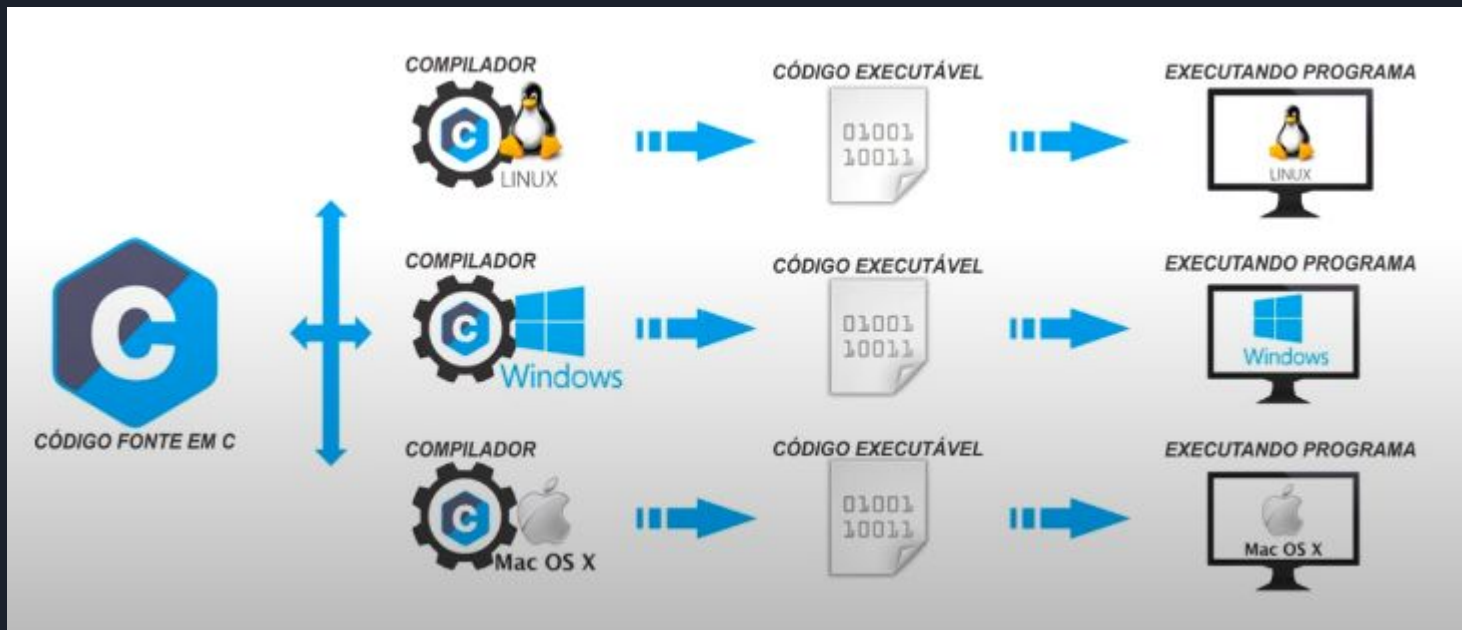
# Java Virtual Machine



Ao lado temos um diagrama simples de códigos compilados, onde é gerado um código específico para a máquina em questão pelo compilador e o assembler, já o linker irá “linkar” as bibliotecas necessárias para o código se tornar executável. Após ele se tornar executável, ele será executado na plataforma para o qual foi compilado.



# Java Virtual Machine



# Java Virtual Machine



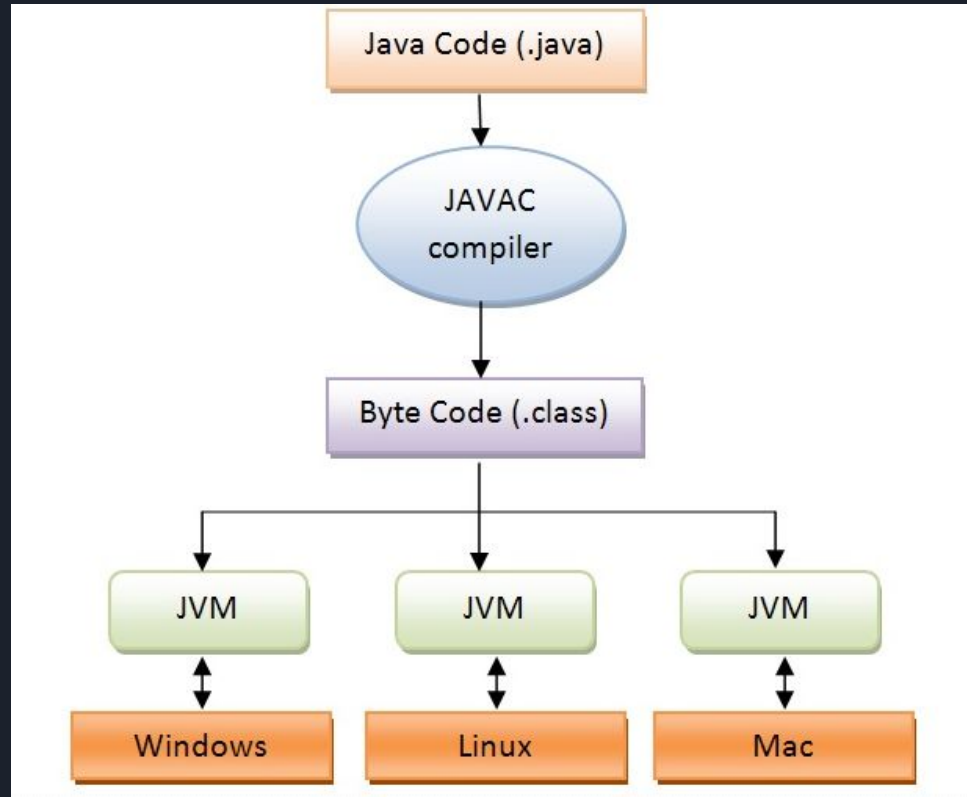
O Java não tem esse problema, pois sua execução não está diretamente relacionada com o Sistema Operacional, ele conversa diretamente com a **JVM** (*Java Virtual Machine*), possibilitando assim a portabilidade de seu código. O que for escrito em um sistema operacional Windows, irá rodar em um sistema operacional Linux (salvo algumas exceções de códigos nativos).

# Java Virtual Machine



Esse processo cria uma independência do Sistema Operacional, dando ao desenvolvedor liberdade em desenvolver para múltiplas plataformas sem preocupação se o código irá funcionar corretamente. A **Virtual Machine** sim é desenvolvida em código nativo, pois ela conversa diretamente com o sistema operacional para que o programa Java funcione na máquina.

# Java Virtual Machine





# Java Virtual Machine



Você pode estar pensando agora: **“Ah, entendi. A Java Virtual Machine é um interpretador de código”**.

# Java Virtual Machine



Não, a JVM é muito mais do que isso, além de interpretar código, é também responsável pela execução das pilhas, gerenciamento de memória, threads e etc., ou seja, é um “**computador virtual**”.

A **JVM** não entende código Java, e sim um código específico chamado **ByteCode**, que é gerado pelo compilador Java (javac). Esse código é o que será traduzido pela Virtual Machine para o código de cada máquina em questão.





# Java Virtual Machine



Uma pergunta muito comum entre os desenvolvedores iniciantes de Java é: “Quero começar a programar em Java, o que devo baixar? JDK ou JRE?”.

# Java Virtual Machine



**JDK**, que é o kit de desenvolvimento (*Java Development Kit*), que é o utilizado pelo desenvolvedor, pois ele possui pacotes que possibilitam o desenvolvimento de aplicações Java em nosso ambiente, já possuindo a JVM.

**JRE**, é o ambiente de execução (*Java Runtime Environment*). Todas as máquinas que rodam uma aplicação Java precisam desse runtime, pois é onde a JVM estará contida, e como já comentado anteriormente, irá fazer todo o controle das aplicações Java.



# Java Virtual Machine



Com isso podemos perceber, mesmo que superficialmente, que a *Java Virtual Machine* faz um trabalho gigantesco “por detrás dos panos”, não permitindo ao desenvolvedor se preocupar com coisas que “atrapalhariam” o desenvolvimento de uma aplicação.

# Java Virtual Machine



Mas ao mesmo tempo, é importante que se saiba o que acontece lá no “baixo” nível, se o desenvolvedor quer ser realmente completo. Com o decorrer do tempo, o Java irá evoluir como tudo nesse mundo e muitas mudanças podem ocorrer, o que nunca irá mudar (pelo menos espera-se) é o seguinte: ***write once, run anywhere (WORA)***.



Eclipse IDE





IntelliJ IDE





VsCode IDE





# Eclipse IDE



O que é **Eclipse IDE** e para que serve?

É uma IDE para desenvolvimento Java, porém suporta várias outras linguagens a partir de plugins como C/C++, PHP, ColdFusion, Python, Scala e Kotlin.

O Eclipse foi feito em Java e segue o modelo *open source* de desenvolvimento de *software*.

“Um ambiente de desenvolvimento integrado (**IDE** - *Integrated Development Environment*) é um software para criar aplicações que combina ferramentas comuns de desenvolvimento em uma única interface gráfica do usuário (**GUI**).”





# Instalação de ferramentas

JDK: Compilador + bibliotecas java:

<https://www.oracle.com/java/technologies/downloads/#java17>

Windows:

<https://www.oracle.com/java/technologies/downloads/#jdk17-windows>

Linux:

<https://www.oracle.com/java/technologies/downloads/#jdk17-linux>

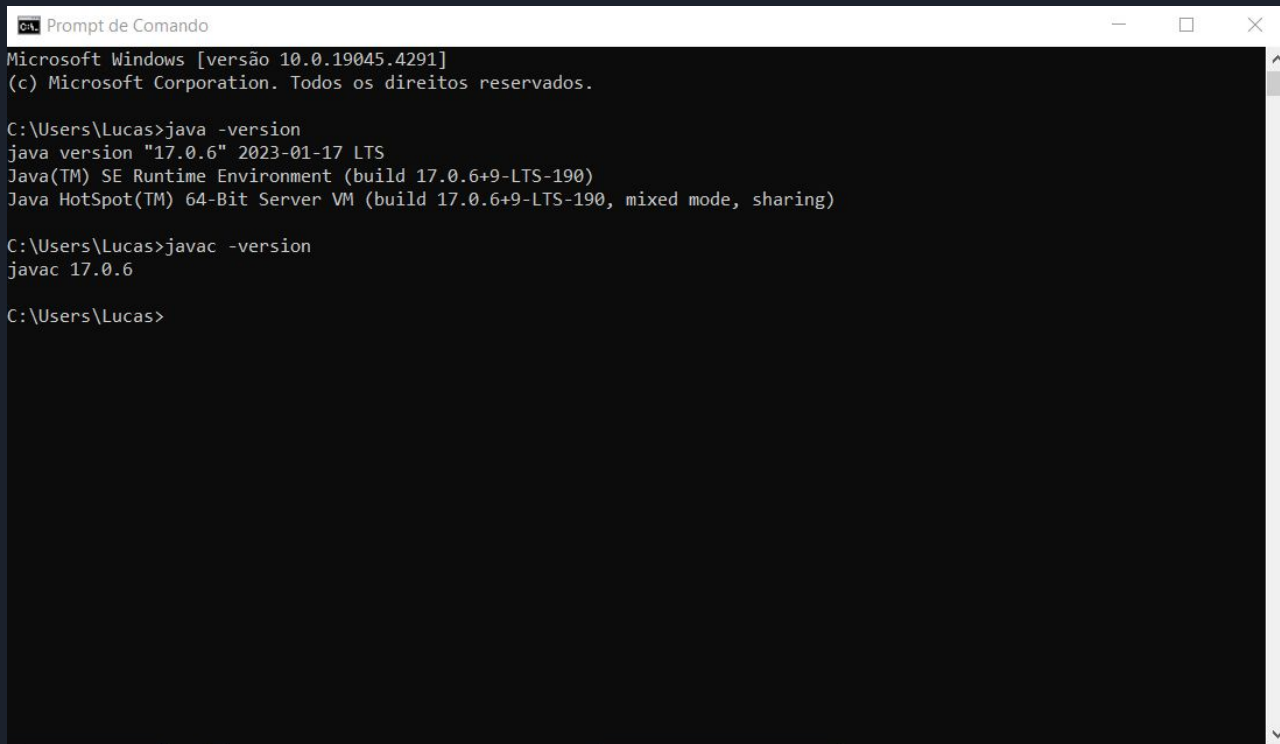
Eclipse IDE:

<https://www.eclipse.org/>

IntelliJ IDEA

<https://www.jetbrains.com/idea/download/?section=windows>

# Testar a Instalação do JDK



```
CL Prompt de Comando
Microsoft Windows [versão 10.0.19045.4291]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\Lucas>java -version
java version "17.0.6" 2023-01-17 LTS
Java(TM) SE Runtime Environment (build 17.0.6+9-LTS-190)
Java HotSpot(TM) 64-Bit Server VM (build 17.0.6+9-LTS-190, mixed mode, sharing)

C:\Users\Lucas>javac -version
javac 17.0.6

C:\Users\Lucas>
```