LINGUAGEM DE PROGRAMAÇÃO 2
02– INTRODUÇÃO AO JAVA

Francisco Barretto – francisco.barretto@udf.edu.br

Roteiro

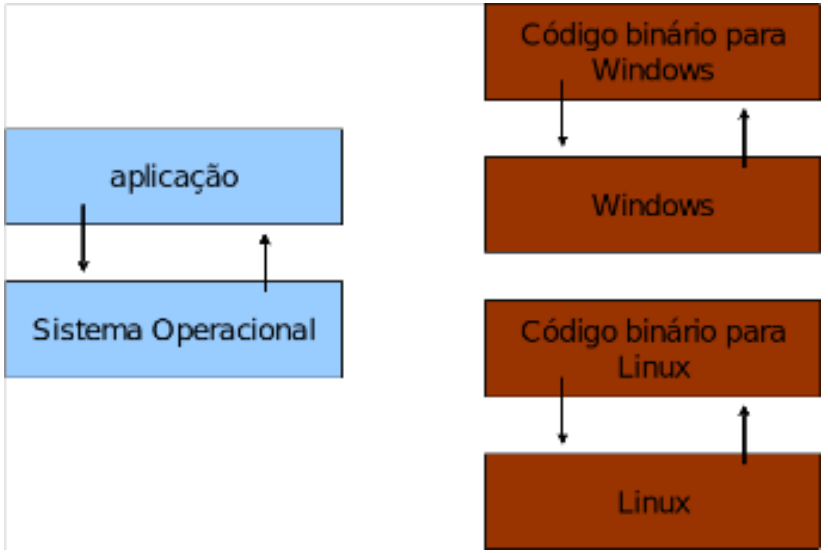
- Maquina Virtual Java (JVM)
- Plataforma Java (Java Platform)
- Hotspot e JIT
- Histórico de Versões (Java Version History)
- JVM/JRE/JDK
- Garbage Collector
- Onde/Quando Usar Java?
- Hello World
- Bytecode

Maquina Virtual Java (JVM)

- Em uma linguagem de programação como C e Pascal, temos a seguinte situação quando vamos compilar um programa:

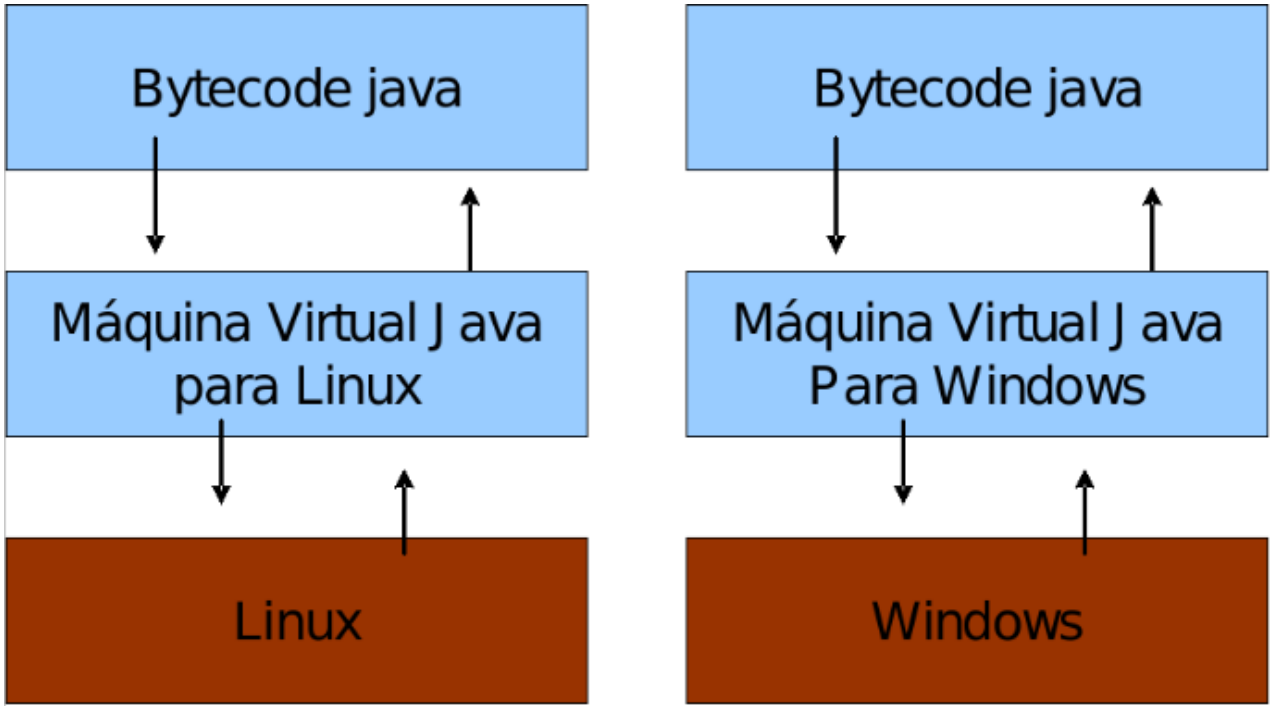
Maquina Virtual Java (JVM)

- Esse código executável (binário) resultante será executado pelo sistema operacional e, por esse motivo, ele deve saber conversar com o sistema operacional em questão.



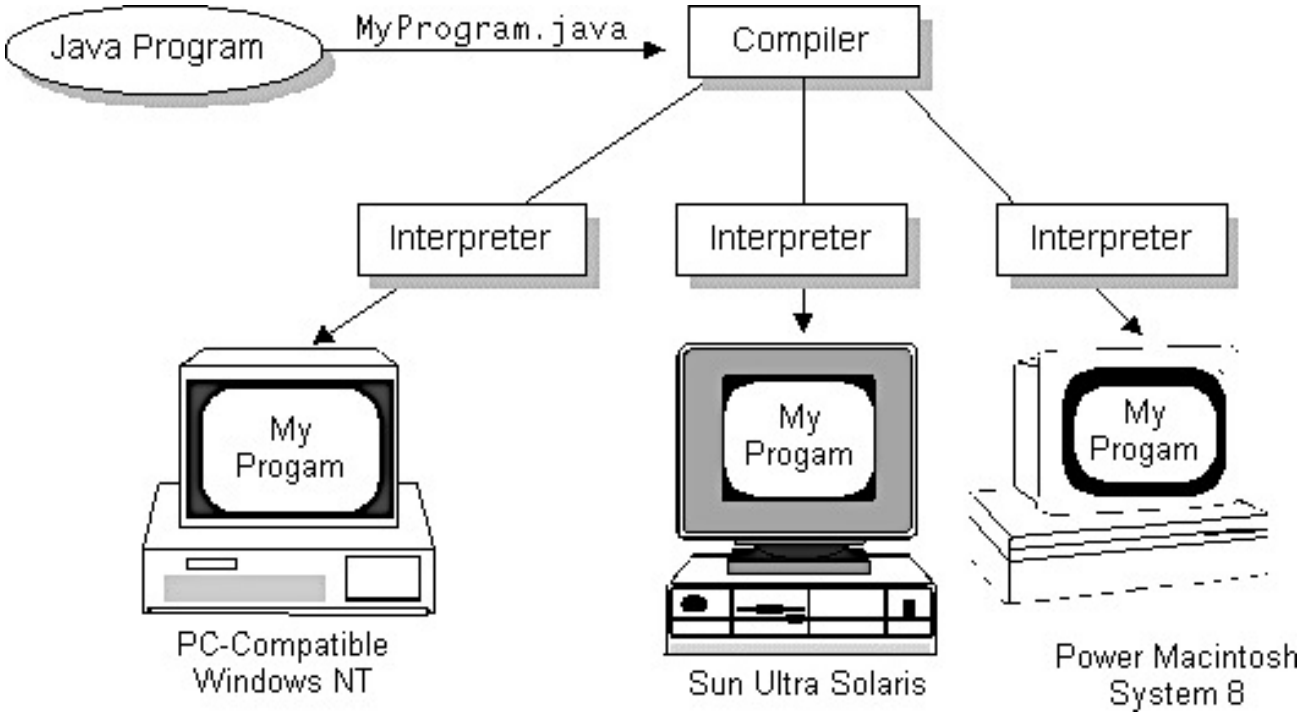
Maquina Virtual Java (JVM)

- Java utiliza do conceito de máquina virtual, onde existe, entre o sistema operacional e a aplicação, uma camada "extra" responsável por "interpretar/traduzir" o que sua aplicação deseja executar



Maquina Virtual Java (JVM)

- Cada interpretador é uma implementação da JVM - *Java Virtual Machine* (ferramenta, browser, hardware)
- "*Write Once, Run Anywhere*"



Maquina Virtual Java (JVM)

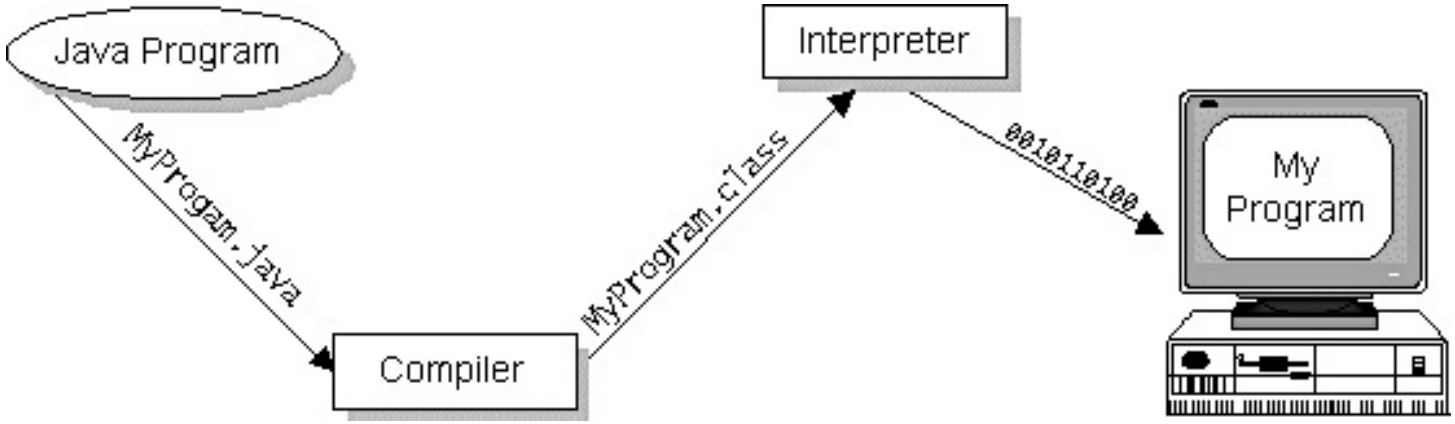
- Independência do Sistema Operacional
- Independência de Plataforma
- JVM é como um computador "de mentira"
  - Tem tudo que um computador/SO: gerencia memória, threads, pilha de execução, etc.
  - Aplicação roda no JVM, sem contato direto com o SO.
- Se tudo passa pela JVM, ela pode tirar métricas, decidir onde melhor alocar a memória, etc.
- Se a JVM termina abruptamente, as aplicações também terminam.

Maquina Virtual Java (JVM)

- JVM em Servidor separa a aplicação do hardware de fato.
- JVM não entende código Java, mas o bytecode.
- Bytecode é gerado pelo compilador Java (javac);
- Existem menos de 256 códigos de operação dessa linguagem (cada "opcode" ocupa um byte).

Maquina Virtual Java (JVM)

- Todos os programas Java são compilados e interpretados
- O compilador transforma o programa em *bytecodes* independentes de plataforma
- O interpretador testa e executa os *bytecodes*



Maquina Virtual Java (JVM)

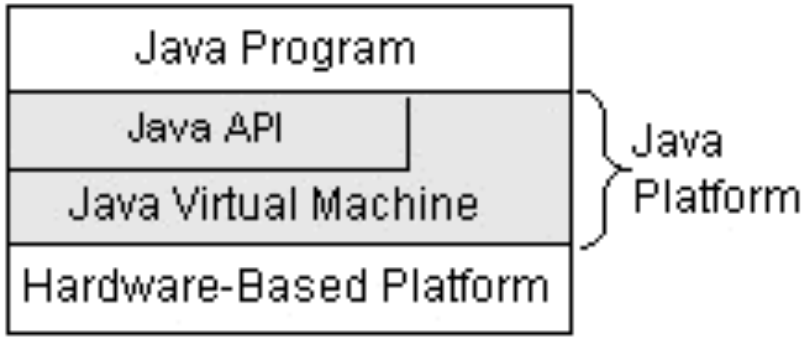
- JVM é uma especificação;
- Descreve como o bytecode deve ser interpretado;
- A Oracle provê a Oracle JVM;
- Existem outras JVMs disponiveis como:
  - JRockit da BEA
  - J9 da IBM
  - OpenJDK
- A implementação da JVM pode ser alterada, caso a empresa não esteja satisfeita com alguma coisa na JVM, deseje melhorar o desempenho em algum ponto ou apenas adquirir a JVM de uma empresa que preste suporte.
- Independência de Hardware/Software/Fabricante (vendor)

Plataforma Java (Java Platform)

- Uma plataforma é o ambiente de hardware e software onde um programa é executado
- A plataforma Java é um ambiente somente de software
- Componentes:

  Java Virtual Machine (Java VM)

  Java Application Programming Interface (Java API)



Hotspot e JIT

- *Hotspot* é a tecnologia que a JVM utiliza para detectar *pontos quentes* da sua aplicação: código que é executado muito, provavelmente dentro de um ou mais loops.
- Este código pode ser, então, compilado para instruções realmente nativas da plataforma.
- O JIT (Just in Time Compiler) é o responsável por realizar esta compilação.
- A adequação é dinâmica, dependendo da necessidade de cada bytecode.

Java Versions

1. JDK Alpha and Beta (1995)
2. JDK 1.0 (January 23, 1996)
3. JDK 1.1 (February 19, 1997)
4. J2SE 1.2 (December 8, 1998)
5. J2SE 1.3 (May 8, 2000)
6. J2SE 1.4 (February 6, 2002)
7. J2SE 5.0 (September 30, 2004)
8. Java SE 6 (December 11, 2006)
   - Java 6 updates
9. Java SE 7 (July 28, 2011)
   - Java 7 updates
10. Java SE 8 (March 18, 2014)
    - Java 8 updates
11. Java SE 9 (September, 2016)
12. Java SE 10

Java Versions

1. JDK Alpha and Beta (1995)
2. JDK 1.0 (January 23, 1996)
   - É a 1ª versão sendo hoje usada para compatibilidade de browsers mais antigos;
3. JDK 1.1 (February 19, 1997)
   - Obteve muitas bibliotecas adicionadas das quais se destacaram o Java RMI, JavaBeans, novo modelo de eventos, JDBC (driver para conexão com banco de dados).
4. J2SE 1.2 (December 8, 1998)
   - Com o tempo surgiu a versão do Java 1.2, que obteve um grande aumento das classes na biblioteca Java (API), ficando considerada a versão da mudança do nome para as versões do produto (JDK) e também sendo optada pela divisão de 3 tipos de plataformas. O principal motivo para essa ação foi que muitos desenvolvedores e usuários estavam confundindo a linguagem Java da linguagem Javascript, que são diferentes. A partir daqui todas as versões Java foram denominadas de Java 2 Standard Edition, que passaram a ter apelidos ou codinomes, esta versão ficou conhecida como Playground da qual foi adicionado o Framework Collections entre outros.
5. J2SE 1.3 (May 8, 2000)
   - Codinome Kestrel, inclusão das bibliotecas JNDI, JavaSound entre outros.
6. J2SE 1.4 (February 6, 2002)
   - Codinome Merlin, criada a palavra reservada “assert”, biblioteca NIO entre outros.

Java Versions

1. J2SE 5.0 (September 30, 2004)
   - A versão mais usada, sendo conhecida com o codinome Tiger. Apesar da versão ser 1.5, agora é chamada apenas de 5. Adições importantes como: Enumeração, Autoboxing, Generics, for-each entre outros estão nela.
2. Java SE 6 (December 11, 2006)
   - Codinome Mustang, teve outras alterações que mudaram na nomenclatura (remoção do 2 - J2SE) e melhora significativa na performance e na estabilidade tendo o surgimento do JIT.
3. Java SE 7 (July 28, 2011)
   - Suporte ao uso de strings em condições do switch;
   - Inferência na criação de objetos com tipos genéricos;
   - Simplificação na invocação de métodos com parâmetros varargs e tipos genéricos;
   - Gerenciamento automático de recursos, tais como conexões a bancos de dados, I/O;
   - Possibilidade de tratar diversas exceções em um mesmo catch (Multicatch) entre outros;
4. Java SE 8 (March 18, 2014)
   - https://www.java.com/pt_BR/download/faq/release_changes.xml

JVM/JRE/JDK

- JVM = apenas a virtual machine (esse download não existe, ela sempre vem acompanhada).
- JRE = Java Runtime Environment, ambiente de execução Java, formado pela JVM e bibliotecas, tudo que você precisa para executar uma aplicação Java.
- JDK = Java Development Kit: Para desenvolvedores de aplicação Java. Neste caso, JDK do Java SE (Standard Edition), formado pela JRE somado a ferramentas, como o compilador.
- http://www.oracle.com/technetwork/java/index.html

JDK

- A formação da JDK é descrita pelas seguintes ferramentas:
- Compilador - javac
- Máquina Virtual (VM) - java
- Documentado de código - javadoc
- Java Debugger - jdb
- Decompilador - javap
- Visualizador de applets - appletviewer
- Depurador simbólico - jdb
- Armazena Recursos - jar
- Tratamento de chaves - keytool
- Políticas de segurança - policytool
- RMI - rmic, rmiregistry
- (...)

Garbage Collector

- Muitas linguagens de programação nos permitem alocar espaço na memória em tempo de execução, uma vez encerrado o programa deve haver uma maneira de liberar este espaço para que outras aplicações possam utilizá-lo.

Garbage Collector

- Em muitas das linguagens de programação, inclusive C e C++, a responsabilidade pela liberação do espaço que não mais será utilizado é do programador, no entanto, nem sempre é fácil gerenciar o que está e o que não está sendo utilizado.
- A má gerência da memória ocasiona muitas vezes o estouro de pilha (stack overflow) entre outros problemas.

Garbage Collector

- Na linguagem de programação Java a responsabilidade pela gerência da memória é do Coletor de lixo (Garbage Collector), desta forma, programadores Java ficam “livres” da preocupação de alocação e desalocação da memória.
- O Coletor de lixo é um processo que roda em segundo plano e é responsável pela liberação de memória alocada por variáveis que não mais serão utilizadas pela aplicação.

Garbage Collector

- A plataforma Java periodicamente libera a memória usada por objetos que não são mais necessários
- O Garbage Collector roda em uma *thread* de baixa prioridade e remove todos os objetos que não são mais referenciados

Onde/Quando Usar Java?

- Aplicações de médio a grande porte;
- Maior equipe de desenvolvedores;
- Linguagem madura;
- Facilidade de ajustes no sistema;
- Orientação a Objetos;

Onde/Quando Usar Java?

- Várias bibliotecas gratuitas (gerar relatorios, gráficos, sistemas de busca, geração de codigo de barras, manipulação de XML, tocadores de video, manipuladores de texto, persistência, impressão, etc.)
- Aplicações que devem crescer (escabilidade);
- Conectividade entre plataformas (multiplos ambientes e sistemas operacionais);
- WEB/Desktop/Mobile

Hello World!

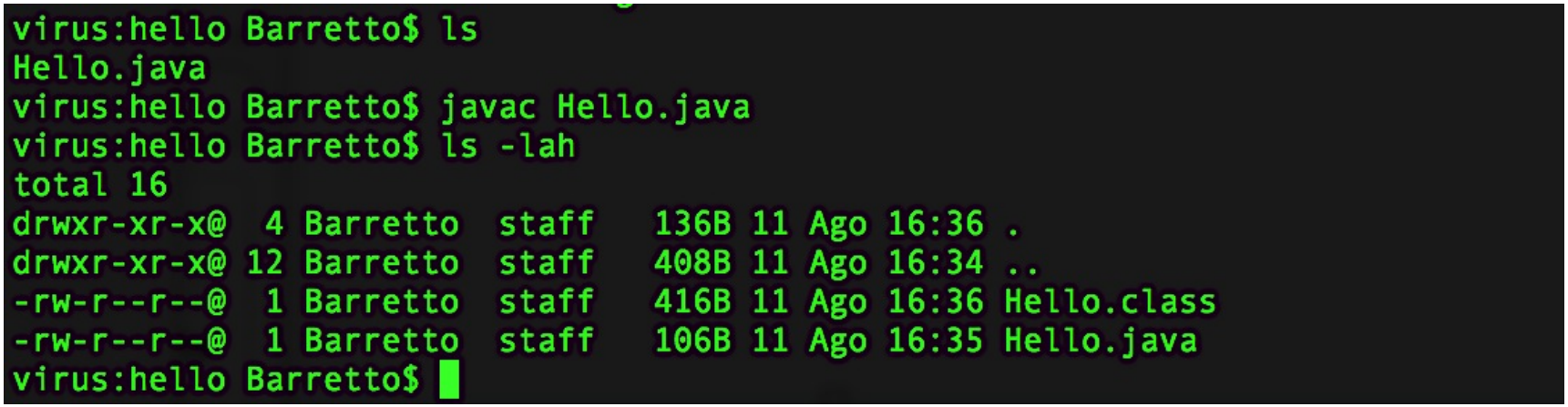
```
System.out.println(”Hello World!");
```

Hello World!

```
1 class Hello{
2    public static void main(String[] args) {
3       System.out.println(“Hello World!”);
4    }
5 }
```

Salve em um arquivo Hello.java em algum diretório.

Hello World!


virus:hello Barretto$ ls
Hello.java
virus:hello Barretto$ javac Hello.java
virus:hello Barretto$ ls -lah
total 16
drwxr-xr-x@ 4 Barretto staff   136B 11 Ago 16:36 .
drwxr-xr-x@ 12 Barretto staff   408B 11 Ago 16:34 ..
-rw-r--r--@ 1 Barretto staff   416B 11 Ago 16:36 Hello.class
-rw-r--r--@ 1 Barretto staff   106B 11 Ago 16:35 Hello.java
virus:hello Barretto$

Hello World!

```
virus:hello Barretto$ java Hello
Hello World!
virus:hello Barretto$
```

Hello World!

```
1 class Hello{
2   public static void main(String[] args) {
3
4     //programa começa aqui!
5     System.out.println(”Hello World!”);
6     // fim do programa
7
8   }
9 }
```

Bytecode?

- O Hello.class gerado não é legível por seres humanos (não que seja impossível). Ele está escrito no formato que a JVM entende e que foi especificado que ela entendesse.
- É como um assembly, escrito para esta máquina em específico. Podemos ler os mnemônicos utilizando a ferramenta javap que acompanha o JDK:
- javap –c Hello
- Um bytecode pode ser revertido para o .java original (com perda de comentários e nomes de variáveis locais).

Exercícios

1. Altere o nome do arquivo para Teste.java e tente compilar e executar novamente o programa.
2. Altere seu programa para imprimir uma mensagem diferente.
3. Altere seu programa para imprimir duas linhas de texto usando duas linhas de código System.out.
4. Sabendo que os caracteres \n representam uma quebra de linhas, imprima duas linhas de texto usando uma única linha de código System.out.

Exercícios

5. Modifique o seu programa para que ele imprima os seguintes "desenhos":

```
* * * * *          *
* * * * *        * *
* * * * *      * * *
* * * * *    * * * *
* * * * *  * * * * *
```

Bibliografia

Básica:

- DEITEL, H. M.; DEITEL, P. J. Java: Como Programar. 8. ed. São Paulo: Pearson Education, 2010
  - Cap 2

Contato

francisco.barretto@udf.edu.br