Conceitos base da linguagem JAVA

Profª. Heloisa Moura

Histórico e estrutura da linguagem Java

- Começou a ser desenvolvida em 1991 pela empresa Sun Microsystems
- Foi lançada oficialmente em 1995.
- Foi comprada pela empresa Oracle em 2009 por erca de 8 bilhões de dólares.
- Atualmente está na versão 17, porém a versão recomendada para download e que tem sido mais utilizada no mercado é a versão 1.8 (ou versão 8).

Histórico e estrutura da linguagem Java

Características:

- É uma linguagem multiplataforma (compatível com diferentes sistemas operacionais)
- É derivada da linguagem C/C++

Histórico e estrutura da linguagem Java

Versões:

- JSE – Java standard Edition: Aplicações desktop.
- JEE – Java Enterprise Edition: Aplicações Web
- JMEE - Java Mobile Edition: Aplicações para dispositivos móveis

Vamos utilizar a versão JSE

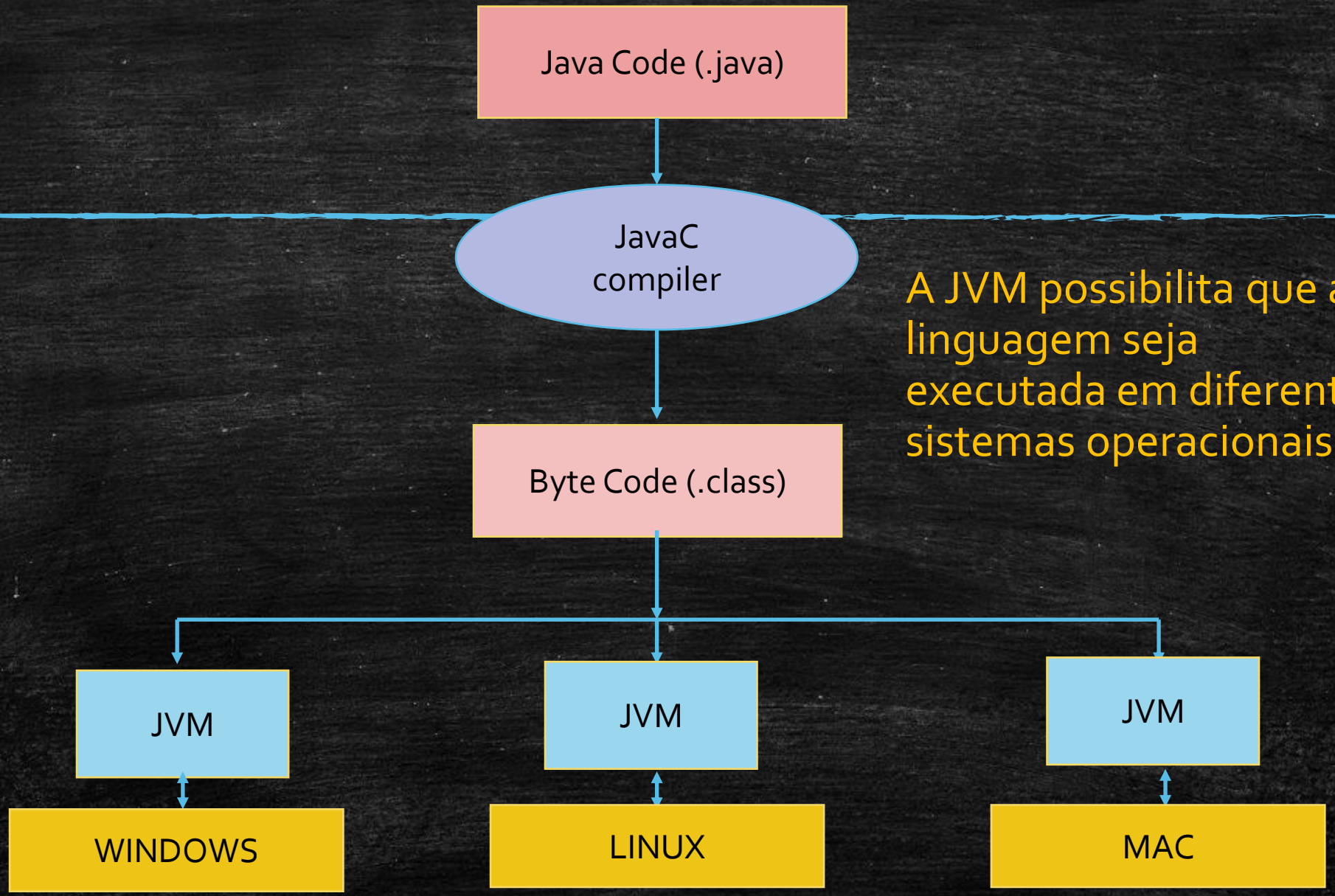
Histórico e estrutura da linguagem Java

Como funciona:

A linguagem Java é executada em cima de uma máquina virtual chamada JVM (Java virtual machine). Essa JVM tem a função de pegar o código complexo da linguagem (chamado byte-codes) e gerar um código executável pela máquina ou sistema operacional.

Histórico e estrutura da linguagem Java

- É uma linguagem compilada e interpretada. O compilador Java, chamado *javac*, compila o código-fonte do Java para um código de nível intermediário chamado *códigos de bytes*. Esses códigos de bytes não são diretamente executáveis em qualquer plataforma de hardware existente, esses códigos são interpretados pelo interpretador Java que é a JVM.
- O fato de Java ser tanto compilado quanto interpretado proporciona ao programador de Java o melhor de ambos os mundos. Um programa escrito em Java é eficiente, por ser compilado, e capaz de ser executado em múltiplas plataformas, por ser interpretado.



A JVM possibilita que a linguagem seja executada em diferentes sistemas operacionais

Introdução ao ambiente de desenvolvimento

- Códigos Java podem ser implementados em qualquer simples editor de texto.
- Mas existem as IDEs (Integrated Development Environment) que facilitam e agilizam o processo de desenvolvimento do software.
- Um ambiente de desenvolvimento integrado (IDE) é uma configuração de programação única, na qual você tem todas as ferramentas necessárias à sua disposição. Geralmente, um editor de código-fonte que é cercado por um compilador, um depurador e outras ferramentas de desenvolvimento. Alguns ambientes de desenvolvimento integrado permitem uma certa flexibilidade, assim o desenvolvedor pode personalizar seu ambiente de trabalho de modo a obter maior produtividade.

Introdução ao ambiente de desenvolvimento

Principais IDEs:

- NetBeans
- **Eclipse**
- Jbuilder

Nós vamos trabalhar com o Eclipse

Introdução ao ambiente de desenvolvimento

Java: Download e instalação
Para fazer o download do Java acesse:
https://www.java.com/pt-BR/download/

Mas antes do download do Java precisamos saber o que é o JRE e o JDK
JRE - Java Runtime Environment: Que serve só para executar aplicações escritas em Java. É composto por bibliotecas e pela JVM.

JDK – Java Development kit: Que serve para o desenvolvimento Java escrever, compilar e executar códigos em Java.

Quando se instala o JDK, o JRE já vem junto. Instala automaticamente. Como vamos desenvolver, então precisamos fazer o download do JDK.

Introdução ao ambiente de desenvolvimento

Eclipse IDE

- **Eclipse IDE é** um Ambiente de Desenvolvimento Integrado de código aberto e gratuito, que reúne ferramentas para apoiar o desenvolvimento de softwares em diversas linguagens de programação, como por exemplo, Java, JavaScript/TypeScript, PHP e entre outras.
  A plataforma é um projeto criado pela IBM em 2001 e suportado por um consórcio de fornecedores de software. Em 2004, a Eclipse Foundation foi criada, uma corporação independente sem fins lucrativos e que atua como administradora da comunidade Eclipse.
- Dessa forma, a fundação permite que a comunidade em torno do software seja neutra, aberta e transparente, usado por milhões de desenvolvedores.

Iniciando no Java - Conceitos base da linguagem

Operadores

Os operadores de atribuição, aritméticos,relacionais e lógicos no Java são utilizados principalmente na etapa de processamento - para a construção da lógica - possibilitando realizar ações específicas sobre os dados. Adição, subtração, multiplicação, comparação são apenas alguns exemplos.

- Atribuição
- Aritméticos
- Relacionais
- Lógicos

Iniciando no Java - Conceitos base da linguagem

Operadores de atribuição em Java

- O operador de atribuição é utilizado para definir o valor inicial ou sobrescrever o valor de uma variável. Em seu uso, o operando à esquerda representa a variável para a qual desejamos atribuir o valor informado à direita.
- Exemplo de uso:

```
1  int lado = 2;
2  float pi = 3.1426F;
3  String texto = “DevMedia”;
4  lado = 3;
```

Nesse exemplo iniciamos as variáveis lado, pi e texto, sobrescrevendo a variável lado em seguida.

Iniciando no Java - Conceitos base da linguagem

Operadores aritméticos

- Os operadores aritméticos realizam as operações fundamentais da matemática entre duas variáveis e retornam o resultado. Caso seja necessário escrever operações maiores ou mais complexas, podemos combinar esses operadores e criar expressões, o que nos permite executar todo tipo de cálculo de forma programática.
- Exemplo de uso:

```java
int area = 2 * 2;
```

Esse código demonstra como calcular a área de um quadrado de lado igual a 2.

Iniciando no Java - Conceitos base da linguagem

Operadores aritméticos

- Também podemos utilizar os operadores aritméticos em conjunto com o operador de atribuição, realizando, em uma mesma instrução, as ações de calcular o valor e atribuí-lo à variável.
- Exemplo de uso:

```
1 | int area = 2;
2 | area *= 2;
```

Nota: *A segunda linha desse código é equivalente a* area = area * 2.

Iniciando no Java - Conceitos base da linguagem

Opções de operadores aritméticos

A tabela abaixo apresenta os **operadores aritméticos** da linguagem **Java**:

+	**OPERADOR DE ADIÇÃO**
-	**OPERADOR DE SUBTRAÇÃO**
*	**OPERADOR DE MULTIPLICAÇÃO**
/	**OPERADOR DE DIVISÃO**
%	**OPERADOR DE MODULO(OU RESTO DA DIVISÃO)**

Iniciando no Java - Conceitos base da linguagem

Operadores de incremento e decremento

Os **operadores de incremento** e decremento também são bastante utilizados. Basicamente temos dois deles:++ e --, os quais podem ser declarados antes ou depois da variável e incrementam ou decrementam em 1 o valor da variável.

Exemplo de uso:

```
1  int numero = 5;
2  numero++;
3  numero--;
4  //numero continuará sendo 5.
```

Iniciando no Java - Conceitos base da linguagem

Operadores de incremento e decremento

Quando declaramos esse operador antes da variável, o incremento é realizado antes do valor da variável ser lido para o processamento ao qual a instrução pertence. Quando declarado depois, ocorre o contrário: lê-se o valor da variável para processamento e só então o valor da variável é incrementado. Com base nisso, suponha que temos o código abaixo:
Exemplo de uso:

```
1  int desafioUm = 5;
2  System.out.println(desafioUm  += ++desafioUm );
3
4  int desafioDois = 5;
5  System.out.println(desafioDois  += desafioDois++);
```

Iniciando no Java - Conceitos base da linguagem

Operadores de incremento e decremento

- Quais valores serão impressos no console? 10 e 10, 10 e 11, 11 e 10 ou 11 e 11? A resposta é 11 e 10.
- No primeiro println(), desafioUm é incrementado antes de seu valor ser lido para compor a instrução de soma. Sendo assim, temos desafioUm = 5 + 6. Já no segundo println(), primeiro o valor é lido, resultando em desafioDois = 5 + 5. Somente após a leitura desafioDois é incrementado, e depois, recebe o valor da soma, tendo seu valor sobrescrito com o número 10.

```
1  int desafioUm = 5;
2  System.out.println(desafioUm  += ++desafioUm );
3
4  int desafioDois = 5;
5  System.out.println(desafioDois  += desafioDois++);
```

Iniciando no Java - Conceitos base da linguagem

- **Operadores de igualdade**

- Os operadores de igualdade verificam se o valor ou o resultado da expressão lógica à esquerda é igual (“==”) ou diferente (“!=”) ao da direita, retornando um valor booleano.

Iniciando no Java - Conceitos base da linguagem

Operadores de igualdade

- Exemplo de uso:

```java
int valorA = 1;
int valorB = 2;

if(valorA == valorB){
                System.out.println("Valores iguais");
} else {
                System.out.println("Valores diferentes");
}
```

Esse código verifica se duas variáveis contêm o mesmo valor e imprime o resultado. Uma vez que as variáveis valorA e valorB possuem valores diferentes, o trecho de código presente no else será executado.

Iniciando no Java - Conceitos base da linguagem

Opções de operadores de igualdade

- A tabela abaixo apresenta os **operadores de igualdade** do **Java**:

==	Utilizado quando desejamos verificar se uma variável é igual a outra.
!=	Utilizado quando desejamos verificar se uma variável é diferente de outra.

Nota: *Os operadores de igualdade normalmente são utilizados para comparar tipos primitivos (byte, short, int, long, float, double, boolean e char). No entanto, também podemos utilizá-los para saber se duas instâncias estão apontando para o mesmo objeto.*

Iniciando no Java - Conceitos base da linguagem

Operadores relacionais

- Os **operadores relacionais**, assim como os de igualdade, avaliam dois operandos. Neste caso, mais precisamente, definem se o operando à esquerda é menor, menor ou igual, maior ou maior ou igual ao da direita, retornando um valor booleano.

Iniciando no Java - Conceitos base da linguagem

Exemplo de uso:

```java
int valorA = 1;
int valorB = 2;

if (valorA > valorB) {
                System.out.println(“maior”);
}

if (valorA >= valorB) {
                System.out.println(“maior ou igual”);
}

if (valorA < valorB) {
                System.out.println(“menor”);
}

if (valorA <= valorB) {
                System.out.println(“menor ou igual”);
}
```

Iniciando no Java - Conceitos base da linguagem

Esse código realiza uma série de comparações entre duas variáveis para determinar o que será impresso no console. Uma vez que o valor da variável valorA é menor que valorB serão impressas as mensagens “menor” e “menor ou igual”.

Iniciando no Java - Conceitos base da linguagem

Opções de operadores relacionais

A tabela abaixo apresenta os operadores relacionais do Java:

>	**Utilizado quando desejamos verificar se uma variável é maior que outra.**
>=	**Utilizado quando desejamos verificar se uma variável é maior ou igual a outra**
<	**Utilizado quando desejamos verificar se uma variável é menor que outra.**
<=	**Utilizado quando desejamos verificar se uma variável é menor ou igual a outra.**
>	**Utilizado quando desejamos verificar se uma variável é maior que outra.**

Iniciando no Java - Conceitos base da linguagem

Opções de operadores de lógicos

A tabela abaixo apresenta os operadores lógicos do Java:

&&	**Utilizado quando desejamos que as duas expressões sejam verdadeiras.**
||	**Utilizado quando precisamos que pelo meno um das expressões seja verdadeira.**

Iniciando no Java - Conceitos base da linguagem

Precedência de operadores

- Uma vez que os operadores aritméticos buscam reproduzir as operações matemáticas fundamentais, é natural que eles mantenham as suas regras de precedência, que podem ser manipuladas pelo programador com o uso de parênteses.
- Por exemplo, a expressão 1 + 1 * 2, quando analisada pelo compilador, vai retornar o valor 3, porque a multiplicação será resolvida antes da adição. Usando parênteses, a expressão (1 + 1) * 2 retornará o valor 4, pois a adição, por estar dentro dos parênteses, será resolvida primeiro.
  Exemplo de uso:

```
1 | if ((1 != (2 -1)) || (2 == (1+1))) {
2 |   System.out.println(“iguais”);
3 | }
```

Nota: *Para facilitar a leitura das expressões e evitar erros de lógica, é recomendado o uso dos parênteses para separar e agrupar as condições.*

Iniciando no Java - Conceitos base da linguagem

Exemplo prático

- Suponha que você precisa programar um código simples para definir o salário dos funcionários de uma empresa considerando o tempo que cada um tem nessa empresa e o número de horas trabalhadas. Para tanto, podemos utilizar alguns dos operadores apresentados nessa documentação.
  Exemplo de uso:

```java
int quantidadeAnos = 5;
int horasTrabalhadas = 40;
int valorHora = 50;
int salario = 0;

if (quantidadeAnos <= 1) {
    salario = 1500 + (valorHora * horasTrabalhadas);
} else if ((quantidadeAnos > 1) && (quantidadeAnos < 3)) {
    salario = 2000 + (valorHora * horasTrabalhadas);
} else {
    salario = 3000 + (valorHora * horasTrabalhadas);
}
```