

The slide features a light gray background with several decorative elements: a large white circle in the top-left corner, a solid purple circle in the top-left area, a purple rounded rectangle in the top-right corner, a purple rounded rectangle in the bottom-left corner, a small purple circle in the bottom-right area, and a large white circle in the bottom-right corner. The main title is centered on the right side of the slide.

Programação II: Introdução a Java


Prof^a. Tainá Isabela



Plataforma Java

A plataforma Java tem como característica principal se desagregar do hardware em si, tornando-se uma plataforma de software que roda em cima de outras plataformas baseadas em hardware.

Essa independência de hardware obtida pela plataforma Java se deve a utilização do conceito de máquina virtual: a Java Virtual Machine (JVM).




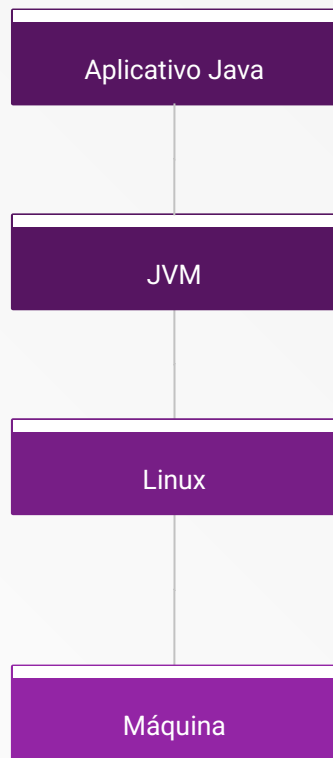
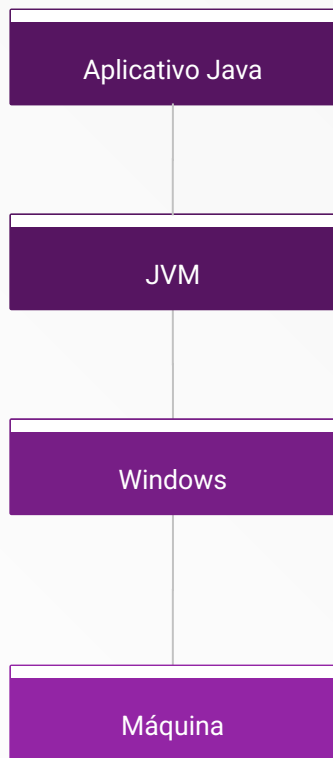


JVM

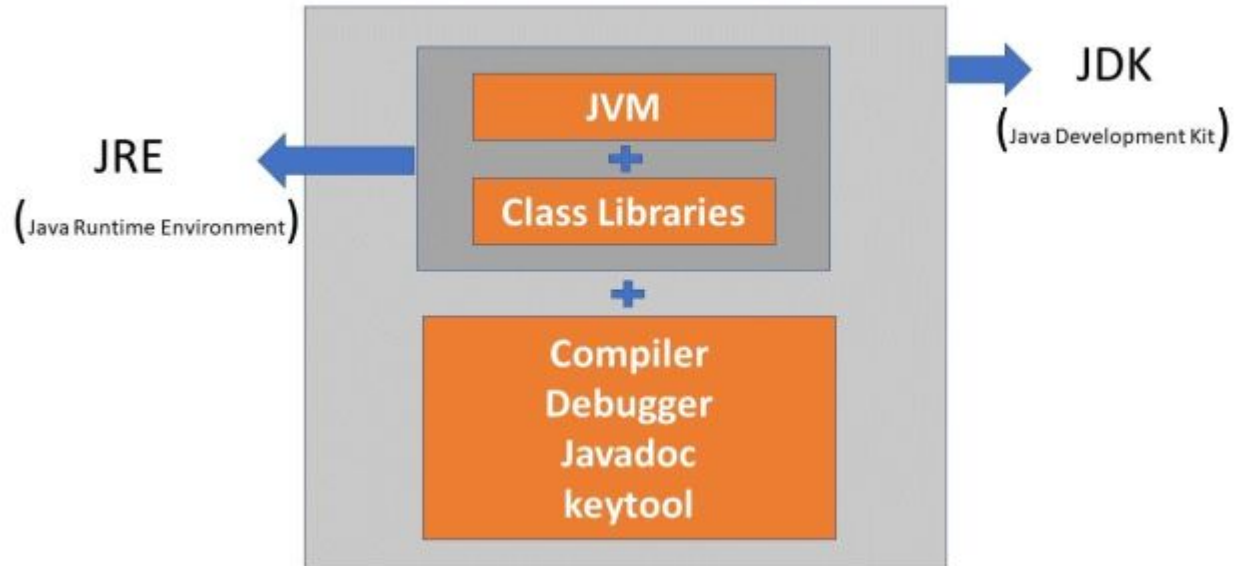
A JVM é um software que funciona sobre o sistema operacional, sendo responsável pelo processo de tradução de um programa Java para uma plataforma específica.

Dessa forma um programa em Java pode rodar em qualquer SO de qualquer arquitetura, desde que exista uma JVM implementada para ele.





JDK, JRE e JVM



Instalar Java

```
sudo apt update
```

```
sudo apt install openjdk-8-jdk -y
```

- <https://www.java.com/pt-br/download/manual.jsp>

Ambientes de desenvolvimento Java

Um programa Java precisa passar por um processo de compilação para ser analisada a existência de erros de sintaxe.

Esse processo de tradução dos códigos fontes para Java bytecodes é feito por um programa chamado **compilador**.

Então, é necessário que outra ferramenta chamada **interpretador** se responsabilize por interpretar esses bytecodes para o sistema operacional.


O conjunto de ferramentas necessárias para desenvolver, compilar e rodar aplicativos Java é disponibilizado em um kit conhecido como **Java Development Kit (JDK)**.

Exemplo em java

```
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Um exemplo de código java");  
    }  
}
```

Javac Exemplo01.java -> Exemplo01.class

java Exemplo01.class



A principal diferença que podemos notar já de início é que todo programa em Java inicia-se com a definição de uma classe. Uma classe é definida pela palavra reservada **class**, seguida pelo nome da classe.

javac Exemplo01.java -> Exemplo01.class

java Exemplo01.class



Variáveis e constantes

Em Java toda variável é identificada por um nome e tem um tipo definido no ato da declaração, que não pode ser alterado.

O tipo da **variável** define, além do tipo de dado que ela pode armazenar, o tamanho do espaço de memória que deve ser alocado para ela.

Em Java não há a definição de constantes. Quando queremos definir variáveis com valores constantes em Java utilizamos a palavra reservada **final**.

Quadro 1.1: Tipos primitivos da linguagem Java

Tipo primitivo	Valores a serem armazenados	Tamanho em <i>bits</i>
<i>char</i>	Permite armazenar um caractere. Exemplo: 'a'.	16
<i>byte</i>	Permite armazenar um número inteiro de -128 até +127. Exemplo: 10.	8
<i>short</i>	Permite armazenar um número inteiro de -32.768 até +32.767. Exemplo: 10.	16
<i>int</i>	Permite armazenar um número inteiro de -2.147.483.648 até +2.147.483.647. Exemplo: 10.	32
<i>long</i>	Permite armazenar um número inteiro de -9.223.372.036.854.775.808 até +9.223.372.036.854.775.807. Exemplo: 10.	64
<i>float</i>	Permite armazenar um ponto flutuante de -3,40292347E+38 até +3,40292347E+38. Exemplo: 3.1416.	32
<i>double</i>	Permite armazenar um ponto flutuante de -1,79769313486231570E+308 até +1,79769313486231570E+308. Exemplo: 3.1416.	64
<i>boolean</i>	Permite armazenar apenas o valor <i>true</i> ou o valor <i>false</i> .	8



Conversões entre tipos primitivos

A conversão de tipos consiste em utilizar uma variável ou valor de um tipo para gerar um novo valor de outro tipo.

Em alguns casos essa conversão é feita de forma direta, como, por exemplo, quando atribuímos um valor **inteiro** a uma variável do tipo **double**.

Nesse caso a conversão é **direta** (também chamada de conversão **implícita**)






Conversões entre tipos primitivos

Mas, alguns tipos de variáveis apresentam valores incompatíveis entre si. Assim, nesses casos não é possível fazer uma atribuição direta.

Para que atribuições como esta sejam possíveis, precisamos solicitar explicitamente que o número real seja moldado (**casted**) como um número inteiro.

Essa operação de conversão entre tipos é também chamada de **casting**.



Casting

Por exemplo, se tentarmos atribuir a uma variável inteira o valor de uma variável double, teremos um erro de compilação:

Para realizar a conversão explícita, coloca-se, antes da expressão a ser convertida, o tipo destino da transformação entre parênteses.

```
7 double q = 24.1;
8 int l;
9 // Casting INCORRETO
10 l=q;
11 l=25.1;
12
13 double p = 9.9;
14 int i;
15 // Casting correto
16 i = (int)p;
17 i = (int)1.1;
18
```

Tipo de Origem	Tipo de Destino	Exemplo
<i>int</i>	<i>double</i>	<i>int a = 20;</i> <i>double b = a; //nesse caso a conversão é implícita</i>
<i>double</i>	<i>int</i>	<i>double a = 20.1;</i> <i>int b = (int) a; //conversão explícita</i>
<i>double</i>	<i>float</i>	<i>double a = 20.1;</i> <i>float b = (float) a; //conversão explícita</i>
<i>long</i>	<i>int</i>	<i>long a = 20;</i> <i>int b = (long) a; //conversão explícita</i>

Operadores

Os operadores aritméticos são símbolos que representam operações aritméticas, ou seja, as operações matemáticas básicas.

Operador	Descrição da operação matemática
+	Soma (inteira e ponto flutuante)
-	Subtração (inteira e ponto flutuante)
*	Multiplicação (inteira e ponto flutuante)
/	Divisão (inteira e ponto flutuante)
--	Decremento unário (inteiro e ponto flutuante)
++	Incremento unário (inteiro e ponto flutuante)
%	Resto da divisão de inteiros

Operadores

Os operadores relacionais são utilizados para realizar comparações entre dois valores de um mesmo tipo, retornando como resultado sempre um valor lógico, ou seja, verdadeiro ou falso.

Descrição	Operador
igual a	== (dois sinais de igual)
maior que	>
menor que	<
maior ou igual a	>=
menor ou igual a	<=
diferente de	!=

Operadores


Os operadores lógicos são utilizados para formar novas proposições lógicas a partir da junção de duas outras.

Descrição	Operador
E	&&
OU	(duas barras verticais)
NÃO	! (exclamação)

Funções e constantes matemáticas

A classe Math possui métodos e atributos static para auxiliar cálculos matemáticos corriqueiros.

- Math.sqrt(x); //raiz quadrada
- Math.sin(x); //seno(x)
- Math.cos(x); //cosseno(x)
- Math.tan(x); //tangente(x)
- Math.exp(x); //ex
- Math.log(x); //ln(x)
- Math.PI; // constante pi




Comandos de decisão ou seleção

O Java fornece três tipos de instruções de seleção. A instrução **if** realiza uma ação se uma condição for verdadeira ou pula a ação se ela for falsa.

A instrução **if...else** realiza uma ação se uma condição for verdadeira ou realiza uma ação diferente se a condição for falsa.

A instrução **switch** realiza uma de muitas ações, dependendo do valor de uma expressão.





Comandos de repetição

O Java fornece três instruções de repetição que permitem que programas executam instruções repetidamente, contanto que uma condição permaneça verdadeira.

As instruções de repetição são **while**, **do...while** e **for**.



Escopo de variáveis

Escopo da variável é o nome dado ao trecho de código em que a variável existe e no qual é possível acessá-la.

Quando abrimos um novo bloco com as chaves, as variáveis declaradas ali dentro só **existirão** até o fim daquele bloco, ou seja, até o ponto onde se fecham as chaves do bloco.

```
public static void main(String [] args){  
    // Aqui a variável x ainda não existe  
    int x = 1;  
    // Aqui a variável x já existe  
    while ( <condição> ) {  
        // Aqui a variável x continua existindo e y ainda não existe  
        int y = 0;  
        // Aqui a variável y já existe  
    }  
    // Aqui a variável já não existe mais. O x continua existindo.  
}
```

Vetores e matrizes

Como vocês já devem ter aprendido nas disciplinas anteriores, vetor (array) é uma estrutura de dados utilizada para representar certa quantidade de variáveis de valores homogêneos, ou seja, um conjunto de variáveis, todas do mesmo tipo.

Em Java podemos criar vetores de tipos primitivos ou de objetos.

```
i.nt numeros[] = new int[3];  
numeros[0] = 57;  
numeros[1] = 51;  
numeros[3] = 37; // Esta linha gera um erro de execução!
```

Vetores e matrizes

Vetores podem ter mais de uma dimensão, sendo conhecidos como vetores multidimensionais ou matrizes. As sintaxes para declaração e uso de matrizes são idênticas às de vetores acrescentando valores para cada uma de suas dimensões.

```
int matriz[][] = new int[10][10];  
matriz[9][9] = -3;  
matriz[2][10]=5;//Erro de compilação!! A segunda dimensão está acima do máximo!
```


Momento Questionário



Atividades

1. Faça um programa que dada a idade de uma pessoa verifique sua classe eleitoral:
 - a. menor que 16 anos não pode votar;
 - b. com 16 ou 17 anos ou mais que 65 anos, votar é facultativo;
 - c. entre 18 e 65 anos (inclusive), votar é obrigatório
2. Faça um programa que imprima os trinta primeiros elementos da série de Fibonacci. A série é a seguinte: 1, 1, 2, 3, 5, 8,13 etc. Para calculá-la, o primeiro e segundo elementos valem 1; daí por diante, cada elemento vale a soma dos dois elementos anteriores.



Atividades - Para os corajosos

3. Crie um programa que armazene um vetor com as notas de dez alunos, calcule e imprima a média dessas notas e depois informe a quantidade de notas acima e abaixo da média calculada.
 4. Faça um programa que fique em laço solicitando a digitação de números inteiros maiores ou iguais a zero. Quando o usuário digitar um número menor que zero, o programa deve exibir a quantidade de números digitados e a média desses números.
- 