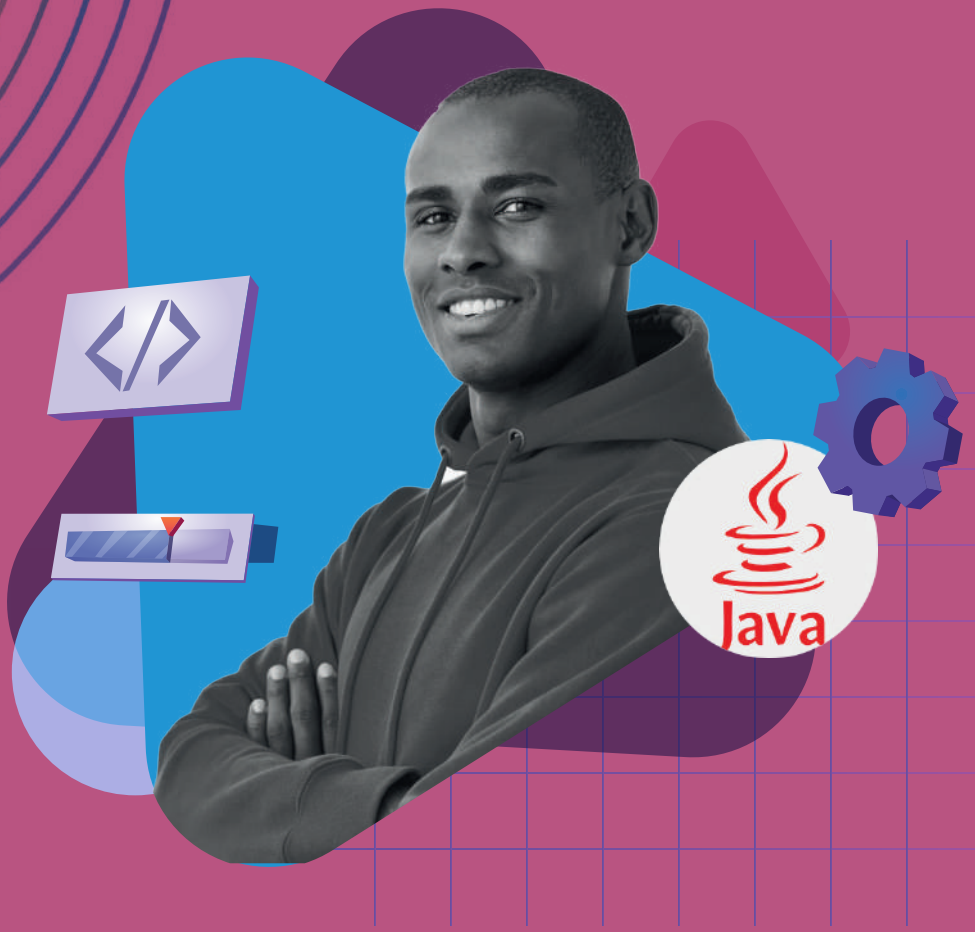
Google olabi

<PLATAFORMA JAVA>



"Write once, run anywhere"

("Escreva uma vez, rode em qualquer lugar")

{ O que é Java?

Java é uma linguagem de programação de alto nível, multiplataforma e orientada para objetos.

{ O que é a Plataforma Java?

A Plataforma Java abriga uma coleção de programas que ajudam os programadores a desenvolverem e executarem aplicativos de programação Java de forma eficiente. Ela inclui um mecanismo de execução, um compilador e um conjunto de bibliotecas.

{ Componentes da linguagem de programação Java

• Kit de desenvolvimento Java (JDK)

JDK é um ambiente de desenvolvimento de *software* utilizado para criar *applets* (pequenos *softwares* que executam uma atividade específica dentro de outro programa maior, como um *plugin*) e aplicativos Java. Os desenvolvedores Java podem usá-lo no Windows, macOS, Solaris ou Linux. O JDK os ajuda a codificar e executar programas Java e possibilita a instalação de mais de uma versão do JDK em um mesmo computador.

Por que usar JDK?

• Contém as ferramentas necessárias para escrever programas Java e JRE que os executem;
• Inclui um compilador e um lançador de aplicativos Java e Appletviewer que converte código escrito em Java para código de byte;
• O lançador de aplicativos Java abre um JRE, carrega a classe necessária e executa seu método principal.

• Ambiente de Tempo de Execução Java (JRE)

JRE é um *software* projetado para executar outro, que contém bibliotecas de classes, classe do carregador e JVM. Caso queira executar um programa Java, você precisará do JRE, e se precisar executar *applets* Java, também precisará instalá-lo em seu sistema.

Por que usar JRE?

• Ele contém bibliotecas de classes, JVM e outros arquivos de suporte (exceto ferramentas de desenvolvimento Java, como depurador, compilador etc.);
• Ele usa classes de pacotes importantes, como bibliotecas Math, Swing, Util, Lang, Awt e Runtime.

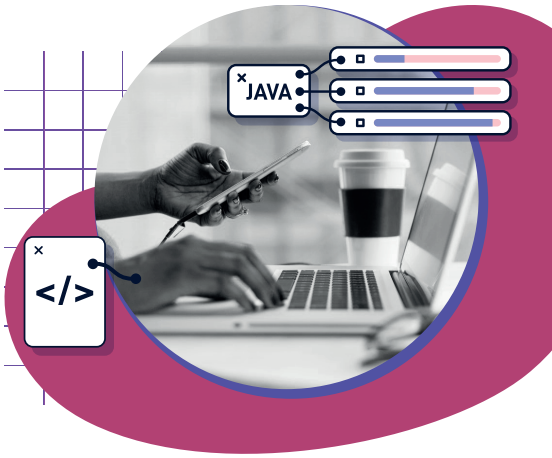
• Máquina Virtual Java (JVM)

JVM é um mecanismo que faz parte do Java Run Environment (JRE) e fornece um ambiente de tempo de execução para conduzir o código Java ou aplicativos. Ele converte *bytecode* Java em linguagem de máquina.

Em outras linguagens de programação, o compilador produz código de máquina para um determinado sistema. Já o compilador Java produz código para uma máquina virtual, conhecida como Java Virtual Machine.

Por que JVM?

• Fornece uma plataforma independente de execução do código-fonte Java;
• Disponibiliza inúmeras bibliotecas, ferramentas e estruturas;
• Permite que, depois de executar um programa Java, você possa executá-lo em qualquer plataforma;
• Vem com o compilador JIT (*Just-In-Time*), que converte o código-fonte Java para linguagem de máquina de baixo nível e o executa mais rapidamente.



Tipos de plataforma Java:

1. Plataforma Java, Standard Edition (Java SE):

A API do Java SE oferece a funcionalidade principal da linguagem de programação Java. Ele define toda a base de tipo e objeto para classes de alto nível. Ele é usado para rede, segurança, acesso a banco de dados, desenvolvimento de interface gráfica do usuário (GUI) e análise XML.

2. Plataforma Java, Enterprise Edition (Java EE):

A plataforma Java EE oferece uma API e ambiente de tempo de execução para desenvolver e executar aplicativos de rede altamente escaláveis, de grande escala, multicamadas, confiáveis e seguros.

3. Plataforma de Linguagem de Programação Java, Micro Edition (Java ME):

A plataforma Java ME oferece uma API e uma máquina virtual de pequeno porte executando aplicativos de linguagem de programação Java em pequenos dispositivos, como telefones celulares.

4. Java FX:

JavaFX é uma plataforma para o desenvolvimento de aplicativos avançados de internet usando uma API de interface de usuário leve. Ele usa gráficos e mecanismos de mídia acelerados por hardware que ajudam o Java a aproveitar os clientes de alto desempenho e uma aparência moderna e APIs de alto nível para conectar-se a fontes de dados em rede.

{ Ambiente de desenvolvimento:

Para escrevermos os códigos, precisaremos de um ambiente de desenvolvimento (IDE). Com isto, utilizaremos o Intellij como IDE.

<Conceitos do que veremos durante a aula: >

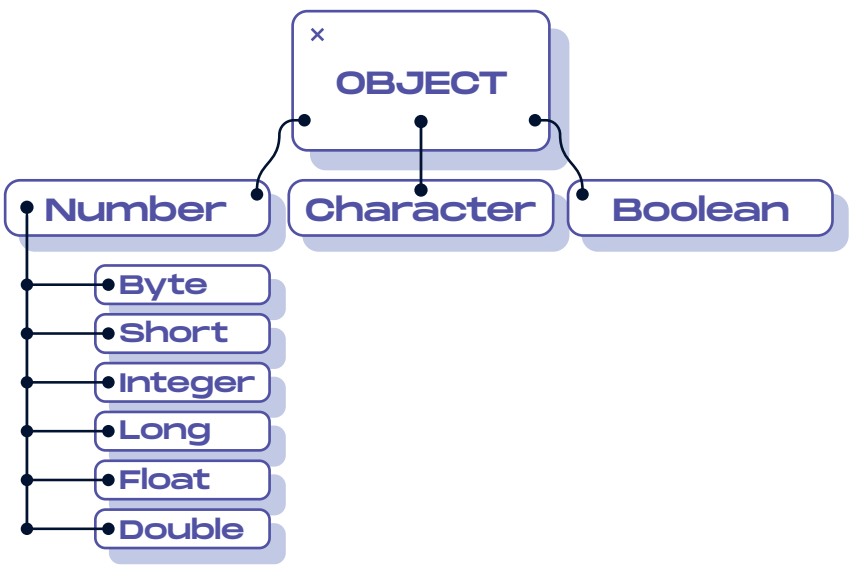
- Numbers:

O tipo de dados mais básico de qualquer linguagem de computador é o número. Geralmente, ao trabalhar com números em Java, usamos tipos de dados primitivos que são byte, short, int, long, float e double.

Como sabemos que esses tipos de dados são apenas valores de dados e não objetos de classe, às vezes precisamos de valores numéricos na forma de objetos. Java resolve este problema fornecendo classes wrapper.

As classes wrapper fazem parte da biblioteca padrão do Java java.lang , que converte os tipos de dados primitivos em um objeto.

O diagrama a seguir mostra uma visão hierárquica dessas classes de wrapper:



A tabela a seguir mostra todos os tipos de dados numéricos com sua classe wrapper correspondente:

Tipos de dados numéricos	Wrapper class
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double

{ Métodos

A classe Number vem com vários métodos que são úteis para realizar operações em números.

- **typeValue:** Converte o valor do objeto da classe Number para o tipo de dados de número primitivo especificado e o retorna. Aqui o type representa tipos de dados de números primitivos: byte, short, int, long, float, double.
- **compareTo:** Compara o argumento de entrada com o objeto Number. Retorna 1 (número positivo) se o valor do objeto Number for maior que o argumento, -1 (número negativo) se for menor que o valor do argumento e 0 se for igual ao argumento. Mas tanto o argumento quanto o número devem ser do mesmo tipo, então somente eles podem ser comparados. O tipo de referência pode ser um byte, double, float, long ou short.
- **equals:** Verifica se o objeto Number é igual ao argumento (também de Number Type). Tanto o objeto Number quanto o argumento podem ser de tipos diferentes. Retorna verdadeiro se os valores forem iguais, caso contrário, retorna falso.
- **parseInt:** Ao trabalhar com strings, às vezes precisamos converter um número representado na forma de string em um tipo inteiro. O método parseInt() é geralmente usado para converter String para Integer em Java. Também podemos passar o argumento radix neste método que representa o tipo decimal, octal ou hexadecimal como saída. Este método retorna o tipo de dados primitivo de uma String.

• **toString:** É usado para obter a representação do objeto String de qualquer objeto Number. Esse método recebe um tipo de dados primitivo como argumento e retorna um objeto String que representa o valor do tipo de dados primitivo.

Existem três variantes deste método;

toBinaryString(int i)
toHexString(int i)
toOctalString(int i)

• **IntergerValueOf:** Converte o valor de um argumento no objeto Number. O argumento pode ser qualquer tipo de dado numérico primitivo ou String.
É um método estático que pode ser chamado diretamente através da classe Integer. O método pode receber dois argumentos, onde um é uma String ou um tipo de dado primitivo e o outro é uma raiz.

{ Strings

Uma string armazena uma sequência de caracteres. O objeto da string é imutável, o texto que ele carrega nunca é alterado. Sempre que o texto precisa ser modificado é utilizado mais espaço em memória

<Como criar uma string?>

Existem diversos métodos de se trabalhar com uma string:

O primeiro deles é o **length**, seu uso é para descobrir o tamanho de uma string. Podemos também criar uma string sem inicializar o seu valor:

Ex: **String** texto;

Temos a possibilidade de comparar string, e caso elas sejam iguais, o valor retorna verdadeiro (true) com o método **equals.**
Também existe o método **equalsIgnoreCase**, com a finalidade de comparar a string ignorando se há letras maiúsculas e minúsculas.

Outro método muito utilizado é o **compareTo**, ele retorna em forma de número inteiro as respostas, por exemplo: Se as strings forem iguais, retorna 0. Caso a comparação seja menor, ele retorna um número negativo e caso for maior, retorna um número positivo.

<Principais métodos de uma classe String:>

- **Concat:** Existem duas formas de unir dois ou mais textos, um deles é usando o operador de concatenação representado por +, e o outro é utilizando o método concat.
- **Length:** Retorna o tamanho do texto.
- **CharAt:** Retorna a localização de um caractere específico. O parâmetro do método é um inteiro, que é usado como índice, e o caractere que será retornado é o que consta nessa posição.
- **IndexOf:** Localiza a primeira ocorrência de um caractere na string.
- **LastOf:** Localiza a última ocorrência de um caractere.
- **Substring:** Permite copiar uma parte da string original, transformando em uma nova string.
- **Replace:** Faz a substituição de algum caractere na string por outro.
- **ToUpperCase:** Converte todos os caracteres para letras maiúsculas.
- **ToLowerCase:** Converte todos os caracteres para letras minúsculas.
- **Trim:** Remove os espaços em brancos existentes no início e final da string.

Referências:

Site oficial: https://www.java.com/en/download/help/whatis_java.html
Java tutorial: https://www.baeldung.com/java-tutorial

Google olabi

<PLATAFORMA JAVA>

/exercícios/

{ Exercício coletivo I

Escreva um programa que pegue as letras iniciais a partir de seu nome completo e as exiba.

{ Exercício coletivo II:

Construa um programa que exiba todos os números primo de 1 a N utilizando o tempo de complexidade O(N^2) para a construção do algoritmo.

{ Exercício de reforço:

Escreva um método que calcule o maior fator primo de um determinado número. Por exemplo: Os fatores primos de 455 são 5, 7 e 13.