

Introdução à linguagem Java

CST em Análise e Desenvolvimento de Sistemas

Prof. Emerson Ribeiro de Mello

mello@ifsc.edu.br

Licenciamento



Slides licenciados sob [Creative Commons “Atribuição 4.0 Internacional”](https://creativecommons.org/licenses/by/4.0/)

Linguagem Java

- Em 1991 Sun Microsystems acreditava que a nova onda computacional seria a **união dos dispositivos eletrônicos portáteis com os computadores**
- Em 1995 Sun lança oficialmente o ambiente Java e sua incorporação no Netscape Navigator trouxe vida as páginas web, antes estáticas



Figura: Produto inicial chamado StarSeven - *7. Fonte: <https://tech-insider.org/java/research/1998/05-a.html>

A onipresença Java

- Aplicações para computadores de mesma
 - IRPF, Astah, IntelliJ
- Aplicações servidoras
 - Apache Tomcat, JBoss, GlassFish
- Aplicações *web*
 - SIGAA
- Dispositivos móveis
 - Aplicativos Android
- Sistemas embarcados
 - Ginga (SBTVD), SmartTVs, Smartcards

Características da linguagem Java

■ Orientada a objetos

- Paradigma que surgiu na década de 60 que tem como foco dados, ou objetos, e suas interfaces
- Recursos de OO do Java são comparáveis aos recursos do C++

■ Robustez

- Apresenta solução elegante para os principais pontos fracos do C++
 - Alocação dinâmica de memória e ponteiros

■ Neutro em relação à arquitetura

- Compilador gera um código intermediário, chamado de *bytecode*
- *bytecode* é executado pela Máquina virtual Java (JVM)

Características da linguagem Java

■ Independente de plataforma

- Escreva uma única vez e execute em qualquer lugar que tenha uma JVM
- Outras linguagens de programação executadas pela JVM. Ex: Groovy, Scala, Kotlin, Jython, JRuby

■ Desempenho

- Os *bytecode* são interpretados pela JVM resultando em um desempenho inferior quando comparado com códigos compilados para um CPU específico
- Os compiladores de bytecode "*just-in-time*" surgem como uma solução para este problema

Alguns mitos e confusões

- **O Java é interpretado, portanto é muito mais lento**




















- Compiladores *just-in-time* permitem que códigos Java sejam executados com tanta rapidez como códigos C++
- A inicialização da JVM e as interfaces gráficas em Java (GUI) são sim lentas

- **Javascript é uma versão simplificada do Java**

- Javascript foi criada pela Netscape para criação de scripts que podem ser usada em páginas Web

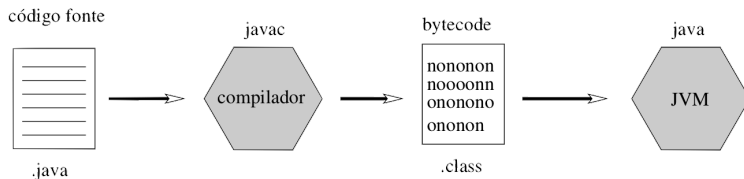
Ranking de linguagens da TIOBE

<https://www.tiobe.com/tiobe-index>

 TIOBE <small>(the software quality company)</small>							
Mar 2025	Mar 2024	Change	Programming Language	Ratings	Change		
1	1		 Python	23.85%	+8.22%		
2	3		 C++	11.08%	+0.37%		
3	4		 Java	10.36%	+1.41%		
4	2		 C	9.53%	-1.64%		
5	5		 C#	4.87%	-2.67%		
6	6		 JavaScript	3.46%	+0.06%		
7	8		 Go	2.78%	+1.22%		
8	7		 SQL	2.57%	+0.65%		
9	10		 Visual Basic	2.52%	+1.09%		
10	15		 Delphi/Object Pascal	2.15%	+0.94%		
11	14		 Fortran	1.70%	+0.48%		
12	9		 Scratch	1.66%	+0.21%		
13	12		 PHP	1.48%	+0.16%		
14	17		 Rust	1.23%	+0.20%		

 TIOBE <small>(the software quality company)</small>											
Programming Language	2025	2020	2015	2010	2005	2000	1995	1990	1985		
Python	1	3	7	7	7	24	23	-	-		
C++	2	4	4	4	3	2	1	2	13		
C	3	2	1	2	1	1	2	1	1		
Java	4	1	2	1	2	3	-	-	-		
C#	5	5	5	6	9	10	-	-	-		
JavaScript	6	7	8	9	10	7	-	-	-		
Go	7	15	36	184	-	-	-	-	-		
Visual Basic	8	18	234	-	-	-	-	-	-		
SQL	9	9	-	-	100	-	-	-	-		
Fortran	10	30	31	24	15	18	5	3	12		
PHP	13	8	6	3	5	27	-	-	-		
Ada	24	36	30	27	16	17	7	6	3		
Lisp	27	31	19	16	14	9	6	5	2		
Objective-C	34	10	3	21	39	-	-	-	-		
(Visual) Basic	-	-	100	5	4	4	3	7	4		

Criando e executando um aplicativo Java



■ Compilando

```
javac Arquivo.java
```

■ Executando

```
java Arquivo
```

Definições iniciais

- Um programa em Java consiste em uma coleção de classes
- Geralmente cada classe possui seu respectivo arquivo `.java`
- O **nome do arquivo** deve ser **idêntico ao nome da classe**
 - Nome do arquivo: `OlaMundo.java`
 - Nome da classe: `OlaMundo`
- O conteúdo do método `main` é a primeira parte de uma classe a ser executada

Primeiro código em Java – OlaMundo.java

```
public class OlaMundo{  
  
    public static void main(String[] args){  
        // imprimindo a mensagem na tela  
        System.out.println("Ola mundo!");  
    }  
}
```

■ Compilando e executando

```
$ javac OlaMundo.java
```

```
$ java OlaMundo
```

Referências sobre a linguagem

Declarando variáveis de tipos primitivos

```
byte b = 65;  
char c = 'A'; // ou c = 65;  
  
int    i = 65;  
long   l = 65L;  
short  s = 65;  
  
double d = 65.1;  
float  f = 65.1f; // ou f = (float) 65.1;  
  
boolean b = true; // ou false  
  
i=i+1;  
i+=10;  
i++;  
++i;
```

[Clique aqui para ver a documentação oficial](#)

Estruturas de decisão

```
if (i > 10){
    System.out.println("É maior");
}else if (i < 10){
    System.out.println("É menor");
}else {
    System.out.println("São iguais");
}

// Use parênteses para melhorar a
// legibilidade e evitar erros
if ((i != 10) && (c == 'a')){
    System.out.println("Operador AND");
}

// Operador ternário
// resultado = (condição) ? valor1 : valor2
String s = (hora > 12) ? "tarde" : "dia";
```

```
switch (i){
    case 1:
        System.out.println("Um");
        break;
    case 2:
        System.out.println("Dois");
        break;
}

switch (i){
    case 1 -> System.out.println("Um");
    case 2 -> System.out.println("Dois");
}

// Switch com expressão
String s = switch(i){
    case 1 -> "Um";
    case 2 -> "Dois";
    default -> "Outro";
};
```

Estruturas de repetição

```
for(int i=0; i<10; i++){  
    System.out.println("i: " + i);  
}  
  
boolean b = true;  
int i = 10;  
  
while((b == true) || (i >= 0 )){  
    System.out.println("Operador OR");  
    b = false;  
}  
  
i = 0;  
do{  
    System.out.println("i: " + i);  
    i++;  
}while(i < 10);
```

Arranjos (vetor e matriz)

```
//vetor de inteiros com 10 posições
int[] vet = new int[10];
vet[0] = 5; // primeira posição
vet[9] = 4; // última posição

// matriz 2x3
int[][] mat = new int[2][3];
// primeira linha, primeira coluna
mat[0][0] = 10;
// última linha, última coluna
mat[1][2] = 3;

// Inicializando vetores com valores
int[] pares = {2, 4, 6};
int[] impares = new int[]{1, 2, 3};
int[][] casas = {{1,2}, {3,4}};

// Percorrendo um vetor
for(int i=0; i<pares.length; i++){
    System.out.println(pares[i]);
}
```

```
// Percorrendo uma matriz
for(int i=0; i<casas.length; i++){
    for(int j=0; j<casas[i].length; j++){
        System.out.println(casas[i][j]);
    }
}

// Percorrendo vetor com for each
for(int valor: pares){
    System.out.println(valor);
}

// Percorrendo matriz com for each
for(int[] linha: casas){
    for(int valor: linha){
        System.out.println(valor);
    }
}
```


Classe utilitária Math¹ e classe Random²

```
// obtém a raiz quadrada
double d = Math.sqrt(25);

// 4 elevado a 2
d = Math.pow(4,2);

// Trigonométricas.  Math.cos(45), Math.tan(45), Math.PI, ...
d = Math.sin(45);

// Arredondamento
long n = Math.round(4.5632); // resultado será 5

// obtendo números pseudo-aleatórios de 0 a 9
Random r = new Random();
int i = r.nextInt(10);
```

¹ <https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html>

² <https://docs.oracle.com/javase/8/docs/api/java/util/Random.html>

Classe String

Métodos `empty`, `equals`, `length` e `repeat`

```
// Criando uma String com o valor "Engenharia"
String s = "Engenharia";

if (s.isEmpty()){ // verifica se está vazia
    System.out.println("Vazia");
}else if (s.equals("Tele")){ // para comparar Strings
    System.out.println("Iguais");
}

// Obtém o tamanho da String
int tamanho = s.length();

// Concatenando Strings
String nova = s + " de Telecomunicações";

// Método repeat para repetir uma String
String repetida = "Java".repeat(3); // resultado:  JavaJavaJava
```

Classe String

Métodos charAt, substring e split

```
String s = "Engenharia";  
// Obtém o caractere na posição 1  
char c = s.charAt(1);  
  
// Obtém uma substring com os 4 primeiros caracteres de s  
String sub = s.substring(0,4);  
  
// Uma String com nomes de alunos separados por ":"  
String alunos = "Joao:Pedro:Ana";  
  
// Dividindo a String em um vetor de Strings  
// O método split recebe um caractere ou uma expressão regular  
// Neste caso, o caractere ":" é usado para dividir a String e o resultado é armazenado  
// em vetAlunos  
String[] vetAlunos = alunos.split(":");  
  
// Acessando o primeiro elemento do vetor, que é "Joao"  
System.out.println(vetAlunos[0]);
```

Formatando Strings

<https://docs.oracle.com/javase/8/docs/api/java/util/Formatter.html>

■ Pequenos exemplos de formatação de Strings

```
String s;  
  
// Olá Juca, aula de P00  
s = String.format("Olá %s, aula de %s", "Juca", "P00");  
  
// Largura de campo de 8 caracteres e precisão de 2 caracteres  
s = String.format("PI: %8.2f, sem máscara: %f", Math.PI, Math.PI);  
  
s = String.format("%05d", 123); // preenche com zeros. Saída: 00123  
  
s = String.format("%5d", 123); // largura de 10 caracteres. Saída: " 123"  
s = String.format("%-5d", 123); // alinhado à esquerda. Saída: "123 "  
  
s = String.format("%o", 123); // inteiro em octal. Saída: 173  
s = String.format("%x", 123); // inteiro em hexadecimal. Saída: 7b
```

Conversões

```
// Convertendo de String para int
String idade = "20";
int i = Integer.parseInt(idade);

// Convertendo de int para String
String a = Integer.toString(i);
String b = String.valueOf(i);
String c = String.format("%d", i);
String d = ""+i;

// divisão de inteiros sempre gera inteiros
double res = 1 / 2; // será igual a 0

// Coerção de tipos (typecasting)
double r = (double) 1 / 2; // será igual a 0.5

int j = (int) Math.round(4.5632); // método round retorna um long
```

Concatenação de Strings com a classe StringBuilder

- A classe String é imutável. Assim, a cada concatenação um novo objeto é criado na memória

```
// Em Java objeto da classe String imutável
String s = "Engenharia";

// JVM cria um novo objeto na memória
s += " de Telecomunicações";
```

- A classe StringBuilder é mais eficiente para concatenar Strings. Opte por ela quando for necessário concatenar várias Strings

```
// Classe mais adequada quando se deseja concatenar strings
StringBuilder sb = new StringBuilder("Engenharia");

// concatenando
sb.append(" de Telecomunicações");

// convertendo para objeto da classe String
String res = sb.toString();
```

Lendo informações pelo teclado

```
import java.util.Scanner;

public class Segundo{

    public static void main(String[] args){

        Scanner teclado = new Scanner(System.in);

        System.out.print("Entre com seu nome: ");
        String s = teclado.nextLine(); // lendo cadeia de caracteres
        System.out.println("Nome:   " + s);

        System.out.print("Informe um número inteiro: ");
        int i = teclado.nextInt(); // lendo inteiro

        System.out.print("Informe um número real: ");
        double r = teclado.nextDouble(); // lendo real

        System.out.println("inteiro: " + i + ", real: " + r);

    }
}
```

Problema com a classe Scanner

- Se o método `nextLine()` for chamado depois dos métodos `nextInt()`, `nextDouble()`, `nextFloat()`, `nextByte()`, `nextShort()`, `nextLong()` ou `next()`, então ele não irá ler os valores
- Os métodos `nextXXXX()` ignoram o caractere de nova linha (NL) e assim esse caractere é consumido pelo `nextLine()` subsequente
- **Solução:** Adicionar um chamada extra do método `nextLine()`

```
int i = teclado.nextInt();  
teclado.nextLine(); // chamada extra para consumir NL  
String s = teclado.nextLine(); // lendo cadeia de caracteres
```


Argumentos de linha de comando

```
public class Argumentos{

    public static void main(String[] args){

        System.out.println("Forneceu " + args.length() + "argumentos");
        System.out.println("Argumentos fornecidos: ");

        for(int i = 0; i < args.length(); i++){
            System.out.println(args[i]);
        }

        // Usando o Foreach para percorrer uma coleção (arranjo, listas etc)
        for(String argumento: args){
            System.out.println(argumento);
        }

    }
}
```

```
javac Argumentos.java
java Argumentos Ola mundo
```

Redirecionamento de entrada

```
import java.util.Scanner;

public class ConverteString{

    public static void main(String[] args){
        Scanner entrada = new Scanner(System.in);

        // Enquanto houver nova linha
        while(entrada.hasNext()){
            String linha = entrada.nextLine();
            System.out.println(linha.toUpperCase());
        }
    }
}
```

```
java ConverteString < entrada.txt
```

Usando interface gráfica para interagir com o usuário

```
import javax.swing.JOptionPane;

public class Terceiro{

    public static void main(String[] args){
        String s = JOptionPane.showInputDialog("Entre com um numero");

        //convertendo String para int
        int numero = Integer.parseInt(s);

        JOptionPane.showMessageDialog(null, numero);
    }
}
```

Classes `LocalDate`³ e `DateTimeFormatter`⁴

Para converter String para formato de data

```
Scanner teclado = new Scanner(System.in);

// O usuário informa a data no formato dd/mm/aaaa
System.out.print("Entre com a data (dd/mm/aaaa): ");
String dataString = teclado.nextLine();

// Formato da data - d representa dia, M representa mês e y representa ano
DateTimeFormatter formato = DateTimeFormatter.ofPattern("dd/MM/yyyy");

// Converte a String para LocalDate, considerando o formato informado na String
LocalDate data = LocalDate.parse(dataString, formato);

// Exibe a data no formato padrão que é yyyy-MM-dd
System.out.println("Data: " + data);

// Exibe a data no formato informado pelo objeto formato
System.out.println("Data: " + data.format(formato));
```

³<https://docs.oracle.com/javase/tutorial/datetime/iso/index.html>

⁴<https://docs.oracle.com/javase/8/docs/api/java/time/format/DateTimeFormatter.html>

Leitura obrigatória



Caelum

Apostila Caelum FJ-11 Java e Orientação a Objetos

<https://www.caelum.com.br/apostila/apostila-java-orientacao-objetos.pdf>

■ Ler capítulos 2 e 3



Google

Google Java Style Guide

<https://google.github.io/styleguide/javaguide.html>