

Introdução à linguagem Java

CST em Análise e Desenvolvimento de Sistemas

Professores: Emerson Mello, Arliones Hoeller Jr.

mello@ifsc.edu.br, arliones.hoeller@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

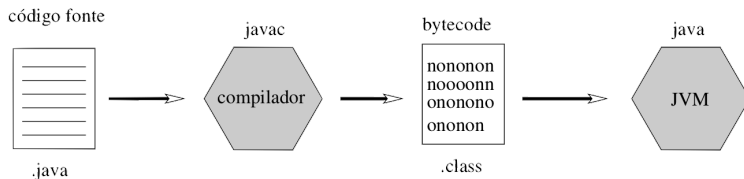
Ranking de linguagens da TIOBE

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

Dec 2023	Dec 2022	Change	Programming Language	Ratings	Change
1	1		 Python	13.86%	-2.80%
2	2		 C	11.44%	-5.12%
3	3		 C++	10.01%	-1.92%
4	4		 Java	7.99%	-3.83%
5	5		 C#	7.30%	+2.38%
6	7	▲	 JavaScript	2.90%	-0.30%
7	10	▲	 PHP	2.01%	+0.39%
8	6	▼	 Visual Basic	1.82%	-2.12%
9	8	▼	 SQL	1.61%	-0.61%
10	9	▼	 Assembly language	1.11%	-0.76%

Programming Language	2023	2018	2013	2008	2003	1998	1993	1988
Python	1	4	8	6	11	24	22	-
C	2	2	1	2	2	1	1	1
C++	3	3	4	3	3	2	2	4
Java	4	1	2	1	1	18	-	-
C#	5	5	5	8	9	-	-	-
JavaScript	6	8	9	9	8	21	-	-
Visual Basic	7	18	-	-	-	-	-	-
PHP	8	7	6	5	6	-	-	-
SQL	9	88	-	-	7	-	-	-
Assembly language	10	13	-	-	-	-	-	-
Ada	24	31	22	20	16	16	7	3
Objective-C	27	12	3	41	48	-	-	-
Lisp	30	29	14	17	15	10	8	2
(Visual) Basic	-	-	7	4	5	3	3	7

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 e repetição

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

if ((i != 10) && (c == 'a')){
    System.out.println("Operador AND");
}

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

```
for(int i=0; i<10; i++){
    System.out.println("i: " + i);
}

while((b == true) || (i >= 0 )){
    System.out.println("Operador OR");
}

do{
    i++;
}while(i<10);

// Operador ternário
// (teste) ? true : false
int resultado = (10 > 20) ? 1 : 2;
```

Arranjos (vetor e matriz)

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

//vetor com 2 dimensões (matriz)
int[][] mat = new int[2][2];
mat[0][0] = 10;
mat[1][0] = 5;

// Declarando e iniciando um vetor com valores
int[] pares = {2, 4, 6};
double[][] casas = {{1,2}, {3,4}};

// Iniciando um vetor com valores
pares = new int[]{10, 8};
```

Classe utilitária Math e classe Random

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

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

// Trigonômétricas.  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);
```


Classe String

```
String s = "Engenharia";  
String sub = null; // valor nulo  
  
if (s.empty()){ // verifica se está vazia  
    System.out.println("Vazia");  
}else if (s.equals("Tele")){ // para comparar Strings  
    System.out.println("Iguais");  
}  
  
sub = s.substring(0,4); // a partir da posição 0 pegue 4 caracteres  
  
int tamanho = sub.length(); //obtendo o tamanho  
char c = sub.charAt(1); // obtendo caractere na posição 1  
  
String alunos = "Joao:Pedro:Ana";  
  
// criará vetor de Strings com 3 elementos  
String[] vetAlunos = alunos.split(":");  
System.out.println(vetAlunos[0]); // Joao
```

Classe String - formatando a saída

```
// Olá Juca, aula de P00
String 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("%10d", 290); // largura de 10 caracteres
s = String.format("%-10d", 290); // alinhado à esquerda
s = String.format("%010d", 290); // preenche com zeros
s = String.format("%o", 290); // inteiro em octal
s = String.format("%x", 290); // inteiro em hexadecimal
```

Veja as máscara de conversão em

<https://docs.oracle.com/javase/7/docs/api/java/util/Formatter.html>

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
```

Classe StringBuilder

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

// JVM cria novos objetos na memória
s += " de Telecomunicações";

// 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);
    }
}
```

Exercícios

Exercícios

- Todos os exercícios deverão ser publicados na sua conta GitHub
- O repositório deverá ter um arquivo `README.md` com as instruções para compilar e executar os programas
- Na raiz do repositório crie o arquivo `.gitignore`. O conteúdo desse arquivo deverá ser gerado no site <https://gitignore.io> com as seguintes *tags*: `java linux windows intellij gradle vscode`
- Para cada exercício você deverá criar um arquivo `.java` separado

Sugestão

- Crie um repositório chamado `poo`
- Crie um diretório chamado `revisao-java` e coloque todos os exercícios dentro desse diretório

Exercício 1

- Faça um programa que leia do teclado seu nome, o ano que você nasceu, o ano atual e imprima na tela seu nome e sua idade.

```
Entre com o seu nome: João  
Entre com o ano que nasceu: 2000  
Entre com o ano atual: 2020
```

Saída:

```
João, possui 20 anos.
```

Exercício 2

- Na disciplina de Programação Orientada a Objetos o aluno será avaliado por meio de 2 projetos práticos (p) e pela participação nas aulas (a). A nota para os projetos práticos é calculada por meio de uma média ponderada, com os seguintes pesos $W = \{w_1, w_2\} = \{2, 3\}$. O Conceito Final (CF) é calculado por meio de uma média ponderada, os projetos com peso 0,9 e a participação nas aulas com peso 0,1. Sendo assim, o Conceito Final (CF) se dará por meio da Equação 1:

$$CF = \left\lfloor \left(\frac{\sum_{i=1}^2 p_i \times w_i}{\sum_{i=1}^2 w_i} \right) \times 0,9 + a \times 0,1 \right\rfloor, \quad CF \in \mathbb{N}. \quad (1)$$

- Desenvolva um aplicativo em Java que permita receber, como argumentos de linha de comando, as notas dos 2 projetos práticos e da participação na aula e depois imprima na tela o conceito final e a palavra APROVADO, caso o $CF \geq 6$ ou REPROVADO, caso contrário.

Exercício 3

Tente adivinhar o número sorteado

- O programa deverá sortear um número de 1 a 100 e deverá questionar o jogador para fornecer um número. O programa só deverá ser encerrado se o jogador acertar o número sorteado. Porém, o programa também deverá dar dicas ao jogador, indicando a quantidade de vezes que ele já optou por um número.
- **Exemplo:** O número sorteado foi 10. O jogador escolhe 5 e o computador exibe “tente outra vez”; o jogador escolhe 20 e o computador exibe “tente outra vez”; o jogador escolhe 5 novamente (ele esqueceu que já tinha tentado com 5) e o computador exibe “você já escolheu 2 vezes o número 5, tente outro número”.

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>