

---

# Programação Orientada a Objetos

— Prof. Gabriel M. Alves —  
2023 (IFSP-SJBV)

---

# Introdução a linguagem Java

# Histórico da linguagem Java

- 1991 - Empresa Sun cria o Green Team liderado por James Gosling
  - oak: a idéia era criar interpretador para pequenos dispositivos (tv, rádio, vídeo-cassete, etc)
- 1995 - advento da web, applets
- 1996 - lançamento da primeira versão;
- Inspirações: C e C++;
- “Write once, run anywhere”

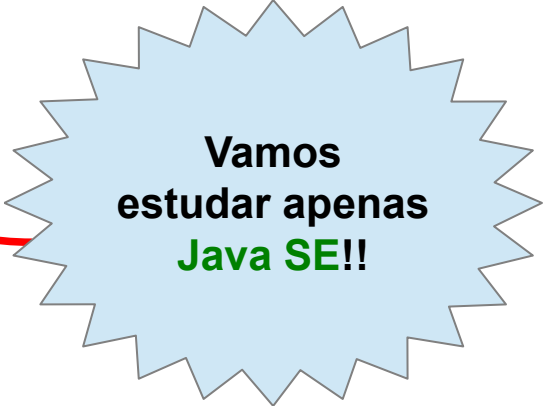


James Gosling

[Fonte da imagem](#)

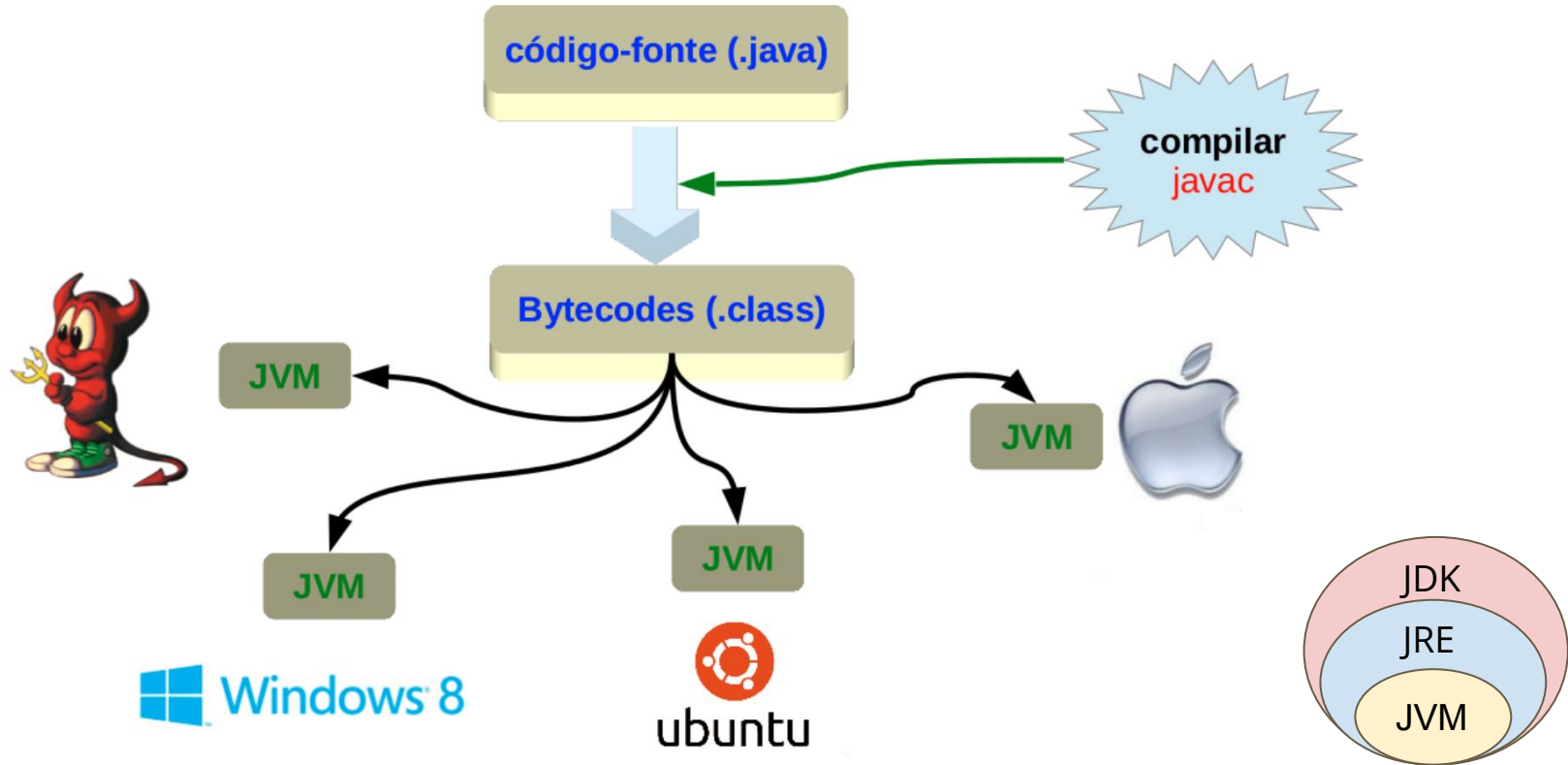
# Histórico da linguagem Java

- Plataforma Java:
  - Java Standard Edititon (Java SE);
  - Java Micro Edition (Java ME);
  - Java Enterprise Edittion (Java EE);
- 2010: Oracle compra a Sun;
- Endereços:
  - [http://en.wikipedia.org/wiki/Java\\_version\\_history](http://en.wikipedia.org/wiki/Java_version_history)
  - <http://www.jcp.org>
  - <http://www.oracle.com>
  - <http://www.java.com>



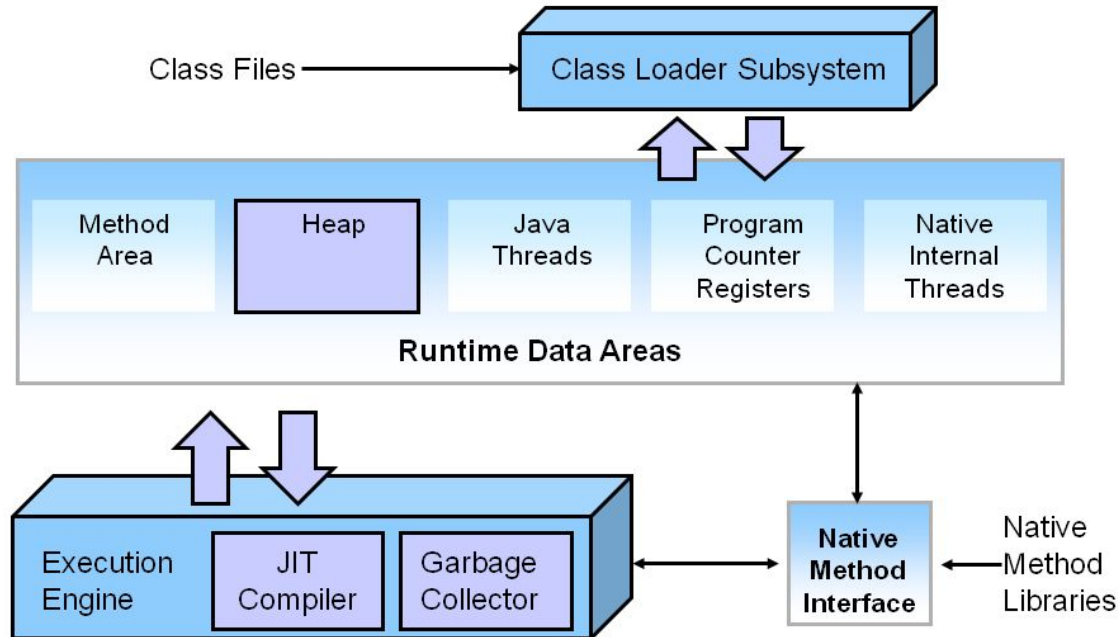
**Vamos  
estudar apenas  
Java SE!!**

# Write once, run anywhere



# Java Virtual Machine (JVM)

## Key HotSpot JVM Components



# Java Virtual Machine (JVM)

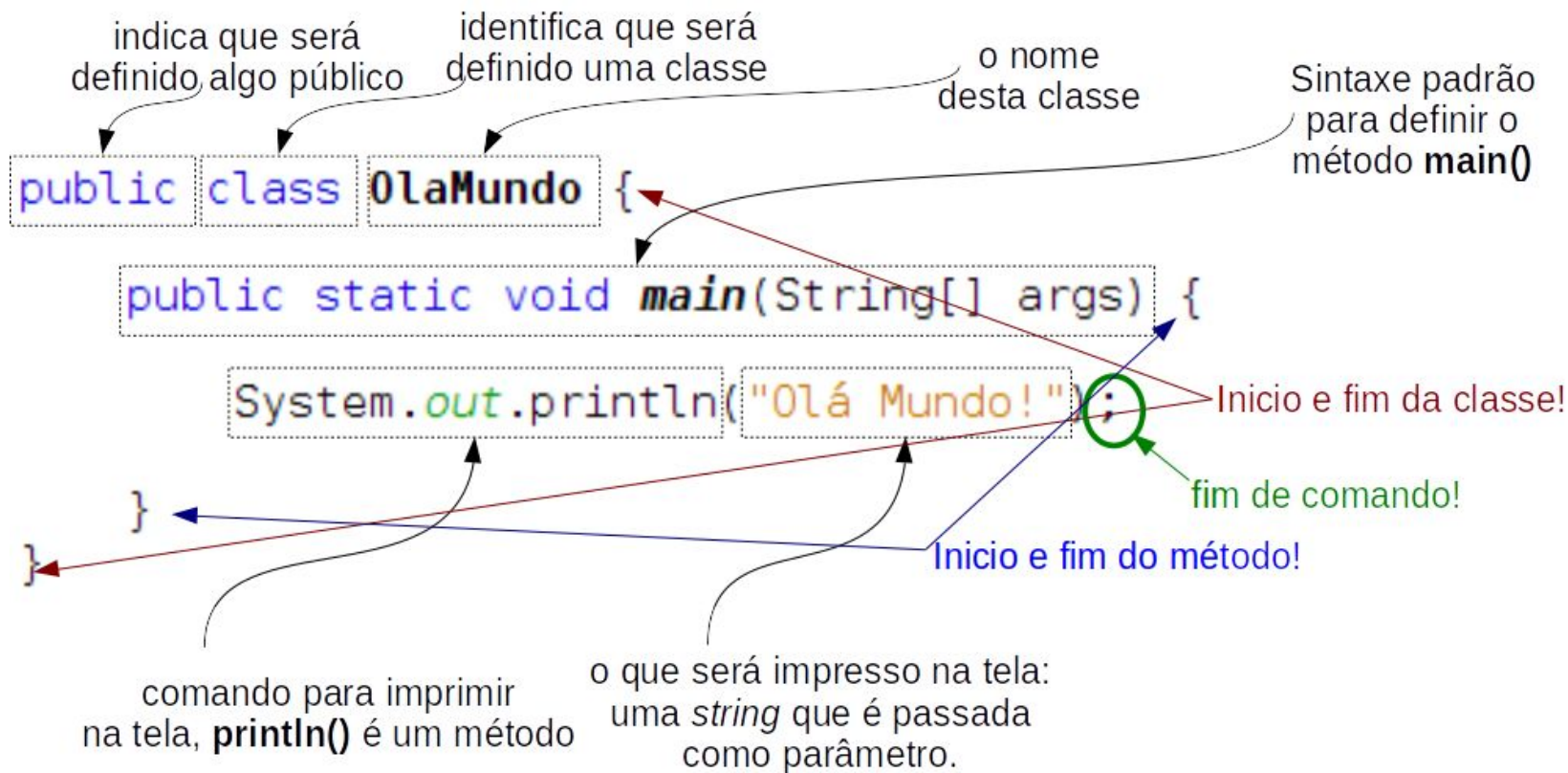
- Isola totalmente a aplicação do sistema operacional
- É uma especificação e não uma implementação
- Implementações da Máquina Virtual:
  - Java HotSpot (Oracle)
  - OpenJDK (Open-source)
  - MJVM (Microsoft)

# JRE e JDK

- JRE – Java Runtime Enviroment
  - ambiente de execução Java
  - VM e bibliotecas padrões
  - **utilizado para executar aplicações** (java)
- JDK – Java Development Kit
  - ambiente de desenvolvimento java
  - javac, jar, javadoc



# Primeiro programa Java



# Considerações gerais

- o arquivo deve conter o nome da classe pública;
- todos os métodos são declarados dentro de uma classe;
- métodos estáticos não são chamados pelos objetos;
- o programa inicia-se pelo método estático main;
- possui 8 tipos primitivos;
- operadores e estruturas de controles são similares a C/C++;
- System.out é um objeto que apresenta informações no terminal;
- Scanner com System.in permite leitura do terminal;

# Atividades práticas

1. Estrutura básica de um programa Java;
2. Compilar um programa (javac)
3. Executar um programa (java)
4. Usar package e salvar .class em outra pasta (javac **-d**)
5. Executar um programa cujo .class está em outra pasta (java **-cp**)

# Atividades práticas

1. Estrutura básica de um programa Java;
2. Verificar os bytecodes de um programa (**javap -c**);
3. Observar os tipos de comentários (javadoc).

## Links úteis:

<https://www.oracle.com/technetwork/articles/java/index-137868.html>

<https://docs.oracle.com/en/java/javase/13/>

<https://www.oracle.com/webfolder/technetwork/tutorials/obe/java/gc01/index.html>

<https://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html>

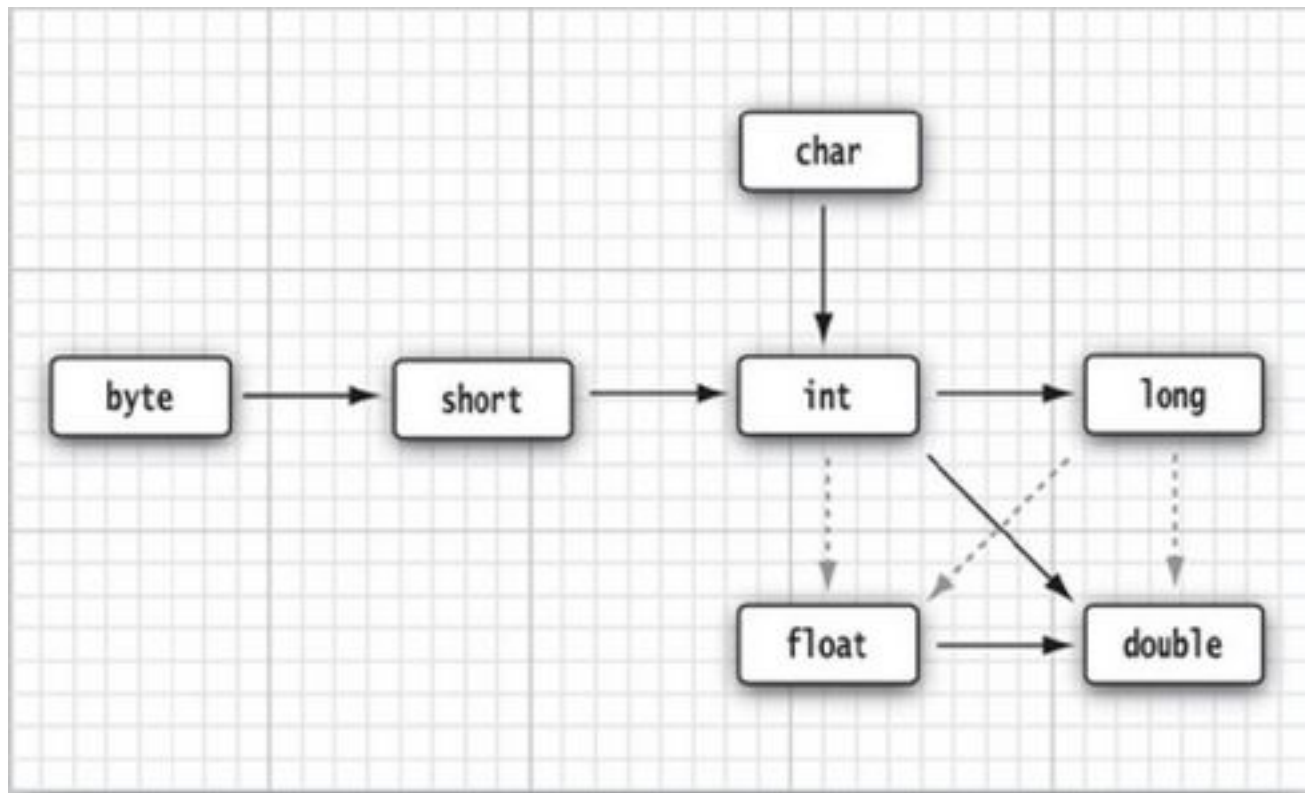
# Estrutura básica - considerações

- objeto pertence a uma classe específica (instância);
- é possível definir níveis de visibilidade (public, private, protected, default);
- tudo é declarado dentro de uma classe;
- pacote é um conjunto de classes relacionadas;
- Tipos de comentários:
  - `//` uma linha
  - `/*` um bloco `*/`
  - `/**` documentação `*/`

# Tipos primitivos de dados

Tipo	Armazenamento	Intervalo
<b>int</b>	4 bytes	-2.147.483.648 a 2.147.483.647
<b>short</b>	2 bytes	-32.768 a 32.767
<b>long</b>	8 bytes	-9.223.372.036.854.775.808 a 9.223.372.036.854.775.807
<b>byte</b>	1 byte	-128 a 127
<b>float</b>	4 bytes	$\pm 3,40282347e+38f$
<b>double</b>	8 bytes	$\pm 1,79769313486231570e+308$ ( <b>IEE 754</b> )
<b>char</b>	2 bytes	Suporta Unicode (65.536 caracteres)
<b>boolean</b>	VM dependent	1 bit ( <b>true</b> ou <b>false</b> )

# Conversões entre tipos (casting)



# Exemplos

1. `double` varDouble = 20; // double = int (casting **implícito**, sem perda)
2. `int` varInt = (`int`) 20.4; // int = double (casting **explícito**, perda)
3. `byte` varByte = (`byte`) 150; // casting é sempre necessário. O que ocorre?
4. `float` varFloat = 10.3**f**;
5. `long` varLong = 2147483648**L**;
6. `int` varMilhar = 1\_000\_000; // usar `_` como separador



# Variáveis

- fortemente tipada;
- não pode utilizar keywords (if, else, while, etc);
- **precisa ser inicializada antes de ser utilizada;**
- boa prática declarar variáveis próximo da primeira utilização;
- escopo (não é permitido sobreposição);
- **final** denota uma CONSTANTE;
- um conjunto de constantes pode ser organizado em uma enumeração;
- 0x (hex); 0b (binário); 0 (octal).

# Operadores

Operadores	Associatividade
[ ] . ( )	Esquerda
! ~ ++ + - (cast) new	Direita
* / %	Esquerda
+ -	Esquerda
<< >> >>> (shift aritmético)	Esquerda
< > <= >= instanceof	Esquerda
== !=	Esquerda

Operadores (cont.)	Associatividade
& (bitwise and)	Esquerda
^ (bitwise exclusive or)	Esquerda
(bitwise or)	Esquerda
&& (logical and)	Esquerda
(logical or)	Esquerda
? : (condicional)	Esquerda
= += -= *= /= %= <<= >>= >>>= &= ^=  =	Direita

# Estruturas de controle

```
if (condição) { }
```

```
if (condição) {  
} else {  
}
```

```
if (condição) {  
} else if (condição) {  
} else {  
}
```

```
switch (caso) {  
    case 0:  
        break;  
    case 1:  
        break;  
    default:  
}
```

-Xlint:fallthrough

- char, byte, short, int
- Wrappers
- string literal
- enumeração

# Estruturas de controle

```
while (condição) { }
```

```
do {  
} while (condição);
```

```
for (inicialização; condição; incremento) {  
}
```

- **break** e **continue**;

# Leitura do teclado - Scanner

```
import java.util.Scanner;
```

Método	Descrição
Scanner(String), Scanner(File), Scanner(InputStream)	Construtor ex: Scanner( <b>System.in</b> ); //InputStream
next()	Lê uma <b>palavra</b> (string)
nextLine()	Lê uma <b>linha</b> (string)
nextInt(), nextDouble()	Lê um <b>inteiro</b> ou <b>ponto-flutuante</b>
hasNext(), hasNextLine(), hasNextInt(), hasNextDouble()	Verifica se há mais elementos a serem lidos

**Obrigado!**

*This work is licensed under Attribution-NonCommercial-ShareAlike 4.0 International. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-sa/4.0/>*

