

# **JAVA — PARTE 1**

Profº Wellington Moreira de Oliveira



# JAVA: CRIADORA E ATUAL PROPRIETÁRIA



# JAVA: CRIADORA E ATUAL PROPRIETÁRIA

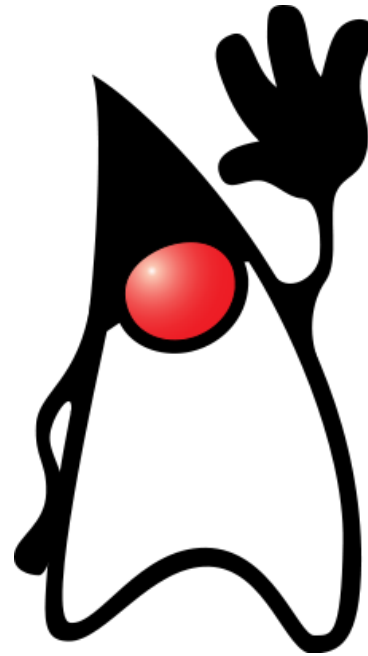
2010



# LOGO E MASCOTE



Java



Duke



# DUKE MINEIRO



# UTILIZAÇÃO JAVA



# EVOLUÇÃO DAS LINGUAGENS

- 1960: Algol 60
- 1967: BCPL (Martin Richards)
- 1970: B (Ken Thompson)
- 1972: C (Dennis Ritchie)
- 1980: C++ (Bjarne Stroustrup)
- 1993: Oak (James Gosling)
- 1995: Java (James Gosling)



# HISTÓRIA DO JAVA

- Projeto Green (1991) – liderada por James Gosling
- Comunicação entre aparelhos eletrônicos
- Perceberam que era impraticável criar um programa para cada SO e criaram o **GreenOS**
- **Oak** (carvalho) – nome que já existia
- **Java** – nome de uma cafeteria frequentada pela equipe do projeto (nome tb de uma terra onde se cultiva café)
- Oportunidade na **Web**: não havia muita **interatividade**
- Primeiro a conectar TVs à dispositivos portáteis





# BENEFÍCIOS JAVA

- Suporte à orientação a objetos
- Portabilidade (“write once run anywhere”)
- Segurança
- Multithreaded
- Alta Desempenho
- Dinâmica



# INDICE TIOBE

Jan 2021	Jan 2020	Change	Programming Language	Ratings	Change
1	2	▲	C	17.38%	+1.61%
2	1	▼	Java	11.96%	-4.93%
3	3		Python	11.72%	+2.01%
4	4		C++	7.56%	+1.99%
5	5		C#	3.95%	-1.40%
6	6		Visual Basic	3.84%	-1.44%
7	7		JavaScript	2.20%	-0.25%
8	8		PHP	1.99%	-0.41%
9	18	▲	R	1.90%	+1.10%
10	23	▲	Groovy	1.84%	+1.23%
11	15	▲	Assembly language	1.64%	+0.76%
12	10	▼	SQL	1.61%	+0.10%
13	9	▼	Swift	1.43%	-0.36%
14	14		Go	1.41%	+0.51%
15	11	▼	Ruby	1.30%	+0.24%
16	20	▲	MATLAB	1.15%	+0.41%
17	19	▲	Perl	1.02%	+0.27%
18	13	▼	Objective-C	1.00%	+0.07%
19	12	▼	Delphi/Object Pascal	0.79%	-0.20%
20	16	▼	Classic Visual Basic	0.79%	-0.04%

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

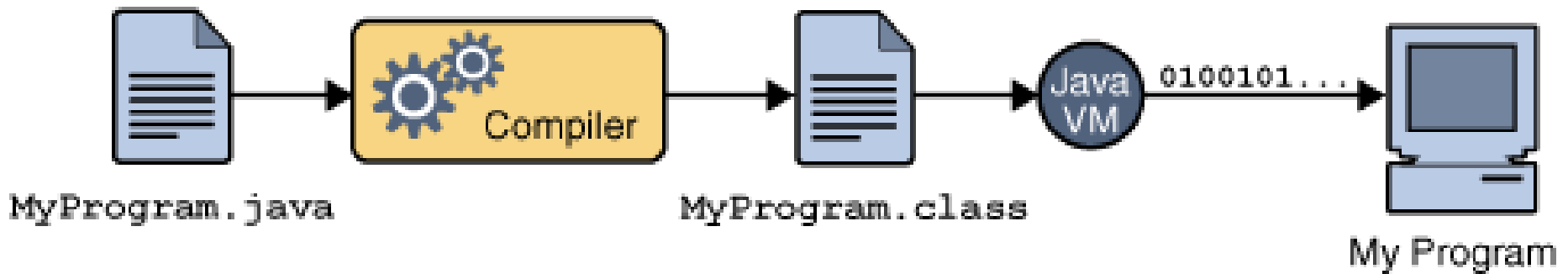


# PRINCIPAIS SIGLAS JAVA

- JDK (Java Development Kit)
- JVM (Java Virtual Machine)
- JRE (Java Runtime Environment)
- J2SE (Java 2 Standard Edition)
- J2ME (Java 2 Micro Edition)
- J2EE (Java 2 Enterprise Edition)
- JDBC (Java Database Connectivity)
- Servlets
- JSP (JavaServer Pages)
- EJB (Enterprise JavaBeans)
- RMI (Remote Method Invocation)



# PROCESSO DE COMPILAÇÃO/INTERPRETAÇÃO



# FASES DOS PROGRAMAS JAVA

- Edição
- Compilação
- Carregamento
- Verificação
- Execução



# CONFIGURAÇÃO DAS VARIÁVEIS DE AMBIENTE

## ➤ java\_home

C:\Program Files\Java\jdk1.8.0\_45

## ➤ classpath

.;%java\_home%

## ➤ Path

;%java\_home%\bin



# CRIANDO, COMPILANDO E EXECUTANDO

- Criar o programa e salvar com a extensão .java
- Abrir o prompt de comando

## **CMD, CD**

- Compilar

**javac meuprograma.java**

- Executar

**java meuprograma**

- Exemplo prático




# PRIMEIRO PROGRAMA JAVA

```
public class OlaMundo {  
    public static void main(String [] args ) {  
        System.out.println("Olá Mundo");  
    }  
}
```





# PRIMEIRO PROGRAMA JAVA

 Command Prompt

```
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\Wellington>cd Desktop

C:\Users\Wellington\Desktop>javac OlaMundo.java

C:\Users\Wellington\Desktop>java OlaMundo
Olá Mundo

C:\Users\Wellington\Desktop>_
```

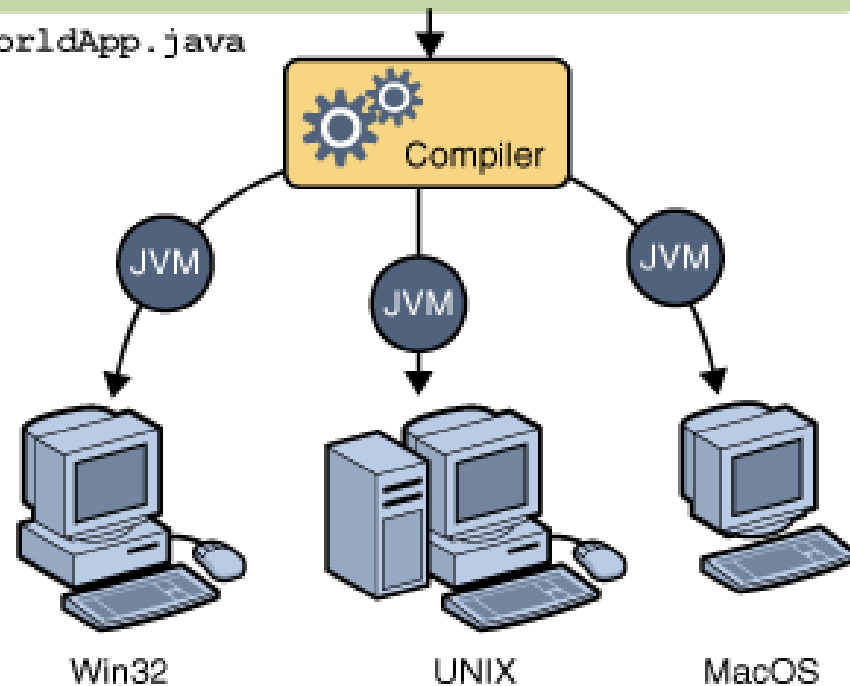


# MULTIPLATAFORMA

Java Program

```
class HelloWorldApp {  
    public static void main(String[] args) {  
        System.out.println("Hello World!");  
    }  
}
```

HelloWorldApp.java



# ERRO DE TEMPO DE EXECUÇÃO (RUNTIME)

## ➤ Fatais

- Para de executar

- Ex.: divisão por zero

## ➤ Não Fatais

- Continua executando



# MAL USO DO JAVA

- Código exótico, distorcido e complexo
- Prática pobre
- Difíceis de ler
- Propenso a comportamentos estranhos
- Difíceis de testar e depurar
- Difíceis de adaptar
- Java não garante portabilidade



# TIPOS PRIMITIVOS

- **boolean:** true, false
- **char:** Usa o código UNICODE e ocupa cada caractere 16 bits.
- **Inteiros:** Diferem nas precisões e podem ser positivos ou negativos.
  - o **byte:** 1 byte.
  - o **short:** 2 bytes.
  - o **int:** 4 bytes.
  - o **long:** 8 bytes.
- **Reais em ponto flutuante:** igual que os inteiros também diferem nas precisões e podem ser positivos ou negativos.
  - o **float:** 4 bytes.
  - o **double:** 8 bytes.



# TIPOS POR REFERÊNCIA

- Não-primitivos
- Referenciam Objetos
- Ex.: String nome;
- Possuem métodos próprios para alteração de tipo, comparação, atribuição de valores etc.



# DELIMITADORES

- //
- /\*
- \*/
- /\*\*
- \*/
- {
- }
- ;



# CARACTER DE ESCAPE

- `\n`

- `\t`

- `\r`

- `\\`

- `\"`





# OPERADORES ARITMÉTICOS

■ +

■ -

■ \*

■ /

■ %

■ Precedência aritmética



# OPERADORES LÓGICOS

■ **&&**

■ **||**

■ **!**



# OPERADORES RELACIONAIS

■ ==

■ !=

■ >

■ <

■ >=

■ <=



# OPERADORES DE ATRIBUIÇÃO COMPOSTOS

■ +=

■ -=

■ /=

■ \*=

■ %=



# OPERADORES UNÁRIOS

■ `a++`

■ `++a`

■ `b--`

■ `--b`



# NOMENCLATURA

- **UPPERCASE**
  - Constantes
- **PascalCase**
  - Classes
- **camelCase**
  - Atributos, métodos
- **Obs.:** Todas as palavras reservadas do java são grafadas em minúsculo



# PALAVRAS-CHAVE E PALAVRAS RESERVADAS

abstract	boolean	break	byte	case	catch
char	class	const	continue	default	do
double	else	extends	final	finally	float
for	goto	if	implements	import	instanceof
int	interface	long	native	new	package
private	protected	public	return	short	static
strictfp	super	switch	synchronized	this	throw
throws	transient	try	void	volatile	while
assert					

- enum
- true - false - null



# MÉTODO BÁSICO DE ENTRADA E SAÍDA

- Saída

```
System.out.println("Minha mensagem");
```

- Entrada

```
import java.util.Scanner;  
  
Scanner scanner = new Scanner(System.in);  
  
scanner.nextFloat();  
scanner.nextLine();  
scanner.nextInt();
```

