



# Características das Linguagens de Programação I

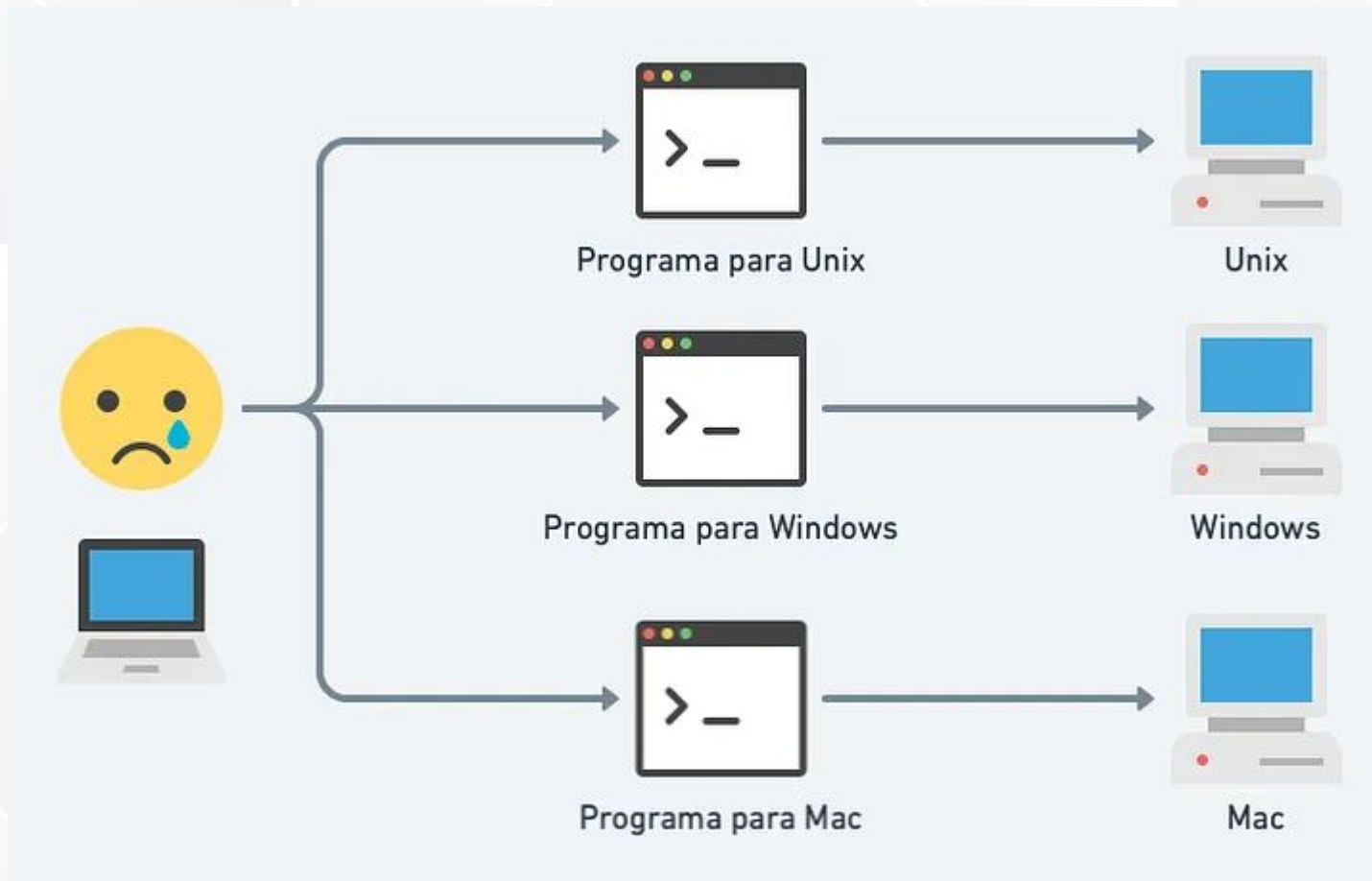
**Conteúdo:** Introdução à Linguagem Java

Prof. Dsc. Giomar Sequeiros  
[giomar@eng.uerj.br](mailto:giomar@eng.uerj.br)

# Introdução a Java

# Introdução

Um programa escrito em em C, C++...



# Java: história

---

- Início em **1991**:
- Pequeno grupo de projeto da **Sun Microsystems**, denominado **Green Team**.
- O projeto visava o desenvolvimento de software para uma ampla variedade de dispositivos de rede e sistemas embutidos.
- **James Gosling**, decide pela criação de uma nova linguagem de programação que fosse simples, portátil e fácil de ser programada.
- Surge a linguagem interpretada **Oak** (carvalho em inglês), mais tarde rebatizada como **Java** devido a problemas de direitos autorais.

# Java: história

---



Starseven criado em oak

# Java: história

---

- Mudança de foco para aplicação na Internet.
  - (**visão**: um meio popular de transmissão de texto, som, vídeo).
- Projetada para transferência de conteúdo de mídia em redes com dispositivos heterogêneos.
- Também possui capacidade de transferir “comportamentos”, junto com o conteúdo. (**HTML por si só não faz isso**)
- Em 1994: **Jonathan Payne** e *Patrick Naughton* desenvolveram o programa navegador **WebRunner**.



# Java: história

## WebRunner

WebRunner is a World Wide Web browser that brings true interactivity to the Internet. WebRunner makes the Internet "come alive". It builds on the network browsing techniques established by Mosaic and expands them by adding dynamic behavior that transforms static documents into dynamic applications capable of real-time interactive response. Using WebRunner you can create applications that range from interactive games to dynamic forms to customized newspapers to interactive shopping ... the possibilities are endless.

WebRunner also provides a new way for users to access these applications. Software transparently migrates across the network. There is no such thing as "installing" software. It just comes when you need it. Content developers can embed new software, media types and protocols in WWW pages, and the extensions reach the user's system automatically.

What makes this dynamic behavior possible is Java (formerly "Oak"), the underlying environment in which WebRunner is built. Java is a simple, dynamic, multithreaded, safe, compact and portable object-oriented programming language and runtime system. Java has an architecture-neutral distribution format, so that Java content runs on anyone's WebRunner home page, regardless of the underlying CPU architecture.

### WebRunner Alpha2 Features:

- Full-Function WWW Browser
  - HTML-compatible with Mosaic and Netscape
  - supports all standard Internet protocols
  - fast performance
- Enables Interactive Content
- Dynamic Content Loading
  - occurs transparently across the network
  - add new protocols and applications on the fly
  - platform-independent
- WWW Newsreader
- Security and Authentication Support
  - file system protection and code checking
- Includes full Java language and runtime system
- Available on Solaris

### WebRunner Beta Features:

- WYSIWYG HTML Editor
  - no HTML knowledge necessary
  - implements all common HTML extensions
  - drag and drop links, images, audio, applications into browser
- Security and Authentication Support
  - S-HTTP, RSA public key encryption and authentication
- HTTP server with support for real-time data feeds
- Available on Solaris, Windows95, and MacOS

For instructions on installing WebRunner, visit <http://tachyon.eng.soloio/>



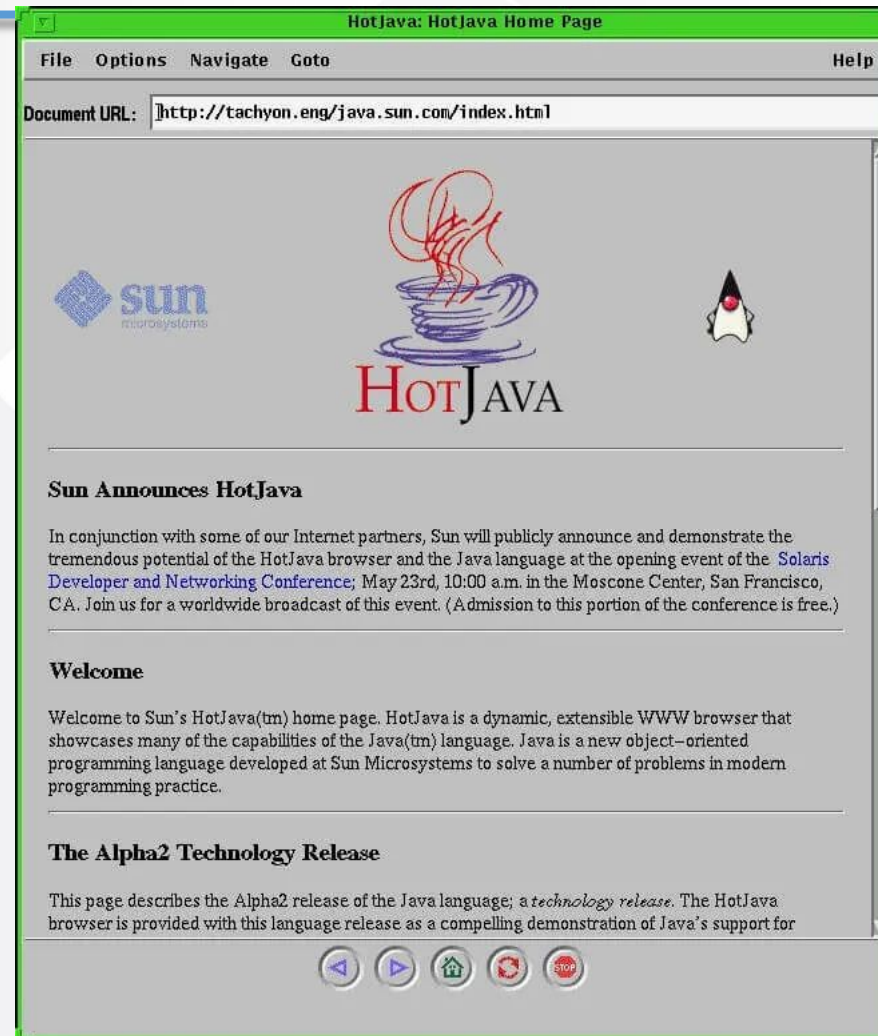
# Java: história

---

- No ***SunWorld'95*** a *Sun* apresenta formalmente o navegador *HotJava* e a linguagem *Java*.
- Poucos meses depois a ***Netscape Corp.*** lança o seu navegador capaz de fazer download e executar pequenos códigos *Java* chamados de **Applets**.
- Imediatamente a *Sun* decide disponibilizar o *Java* gratuitamente para a comunidade de desenvolvimento de softwares e assim surge o *Java Developer's Kit 1.0 (JDK 1.0)*.
- Inicialmente: *Sun Solaris* e *Microsoft Windows 95/NT*.
- Progressivamente surgiram kits para outras plataformas como *Linux* e *Apple Macintosh*.
- Código aberto desde 2006
- Controlada pela Oracle desde 2009



# Java: história



# Java atualmente

---



**The Mars Rover**



**Google**



**eBay**



**Minecraft**



**Netflix**



**Uber**



**Eclipse IDE**



**Twitter**



**Spotify**



**CashApp**



**Signal**



**NASA WorldWind**

# Características de Java

---

- ✓ Simples
- ✓ Portável
- ✓ Orientada a Objetos
- ✓ Interpretada
- ✓ Distribuída
- ✓ Alta Performance
- ✓ Robusta
- ✓ Multitarefa
- ✓ Segura
- ✓ Dinâmica
- ✓ De Arquitetura Neutra



# Java Características: Simples

---

- É de fácil aprendizado.
- Puramente **orientada a objetos**:
- Permite o desenvolvimento de sistemas de uma forma **mais natural**.

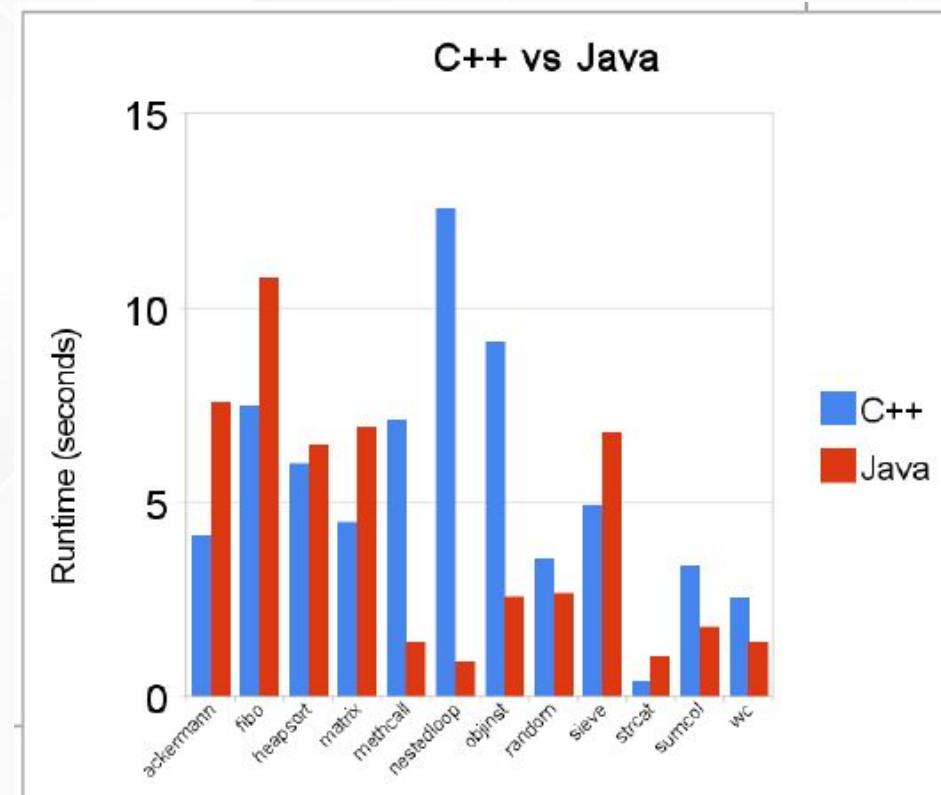
# Java Características: Distribuída

---

- Projetada para trabalhar em ambiente de **redes**.
- **Não** é uma linguagem para **programação distribuída**:
  - Oferece bibliotecas para facilitar o processo de comunicação.
- É uma linguagem interpretada e existe uma grande discussão quanto a sua performance.

# Java Características: Alta Performance

- É uma **linguagem interpretada** e existe uma grande discussão quanto a sua performance.
- As melhorias na tecnologia de compilação, tem aproximado o desempenho ao de linguagens como C e C++.
- Ex.: Benchmarks mostram melhor desempenho em alguns casos e pior em outros, caracterizando “empate técnico”.





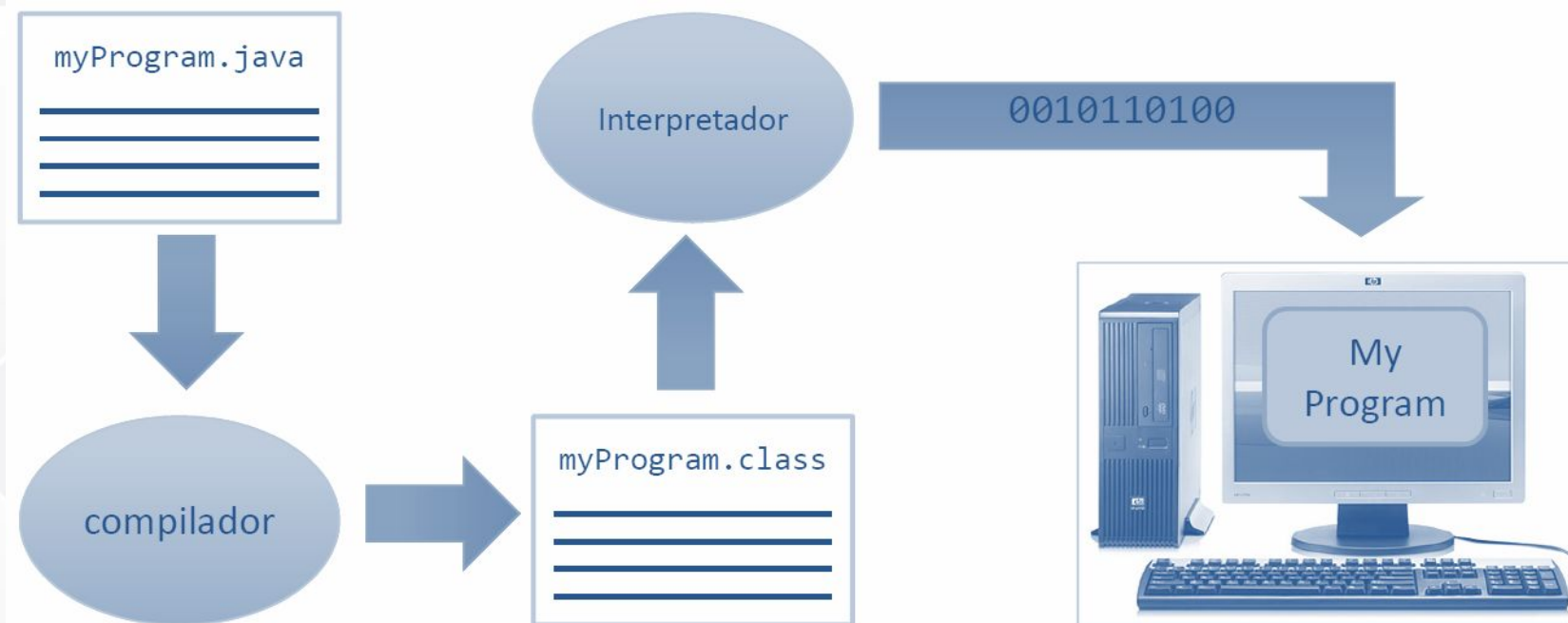
# Java Características: Robusta e Segura

---

- É fortemente tipada;
- Não possui aritmética de ponteiros;
- Possui mecanismo de coleta de lixo;
- Possui verificação rigorosa em tempo de compilação;
- Possui mecanismos para verificação em tempo de execução;
- Possui gerenciador de segurança.
- possui mecanismos de segurança que evitam operações no sistema de arquivos da máquina alvo.

# Java Características: Interpretada, Neutra, Portável

- **Bytecodes** executam em qualquer máquina que possua uma JVM, permitindo que o código em Java possa ser escrito independente da plataforma.
- A característica de ser neutra em relação à arquitetura permite uma grande portabilidade.



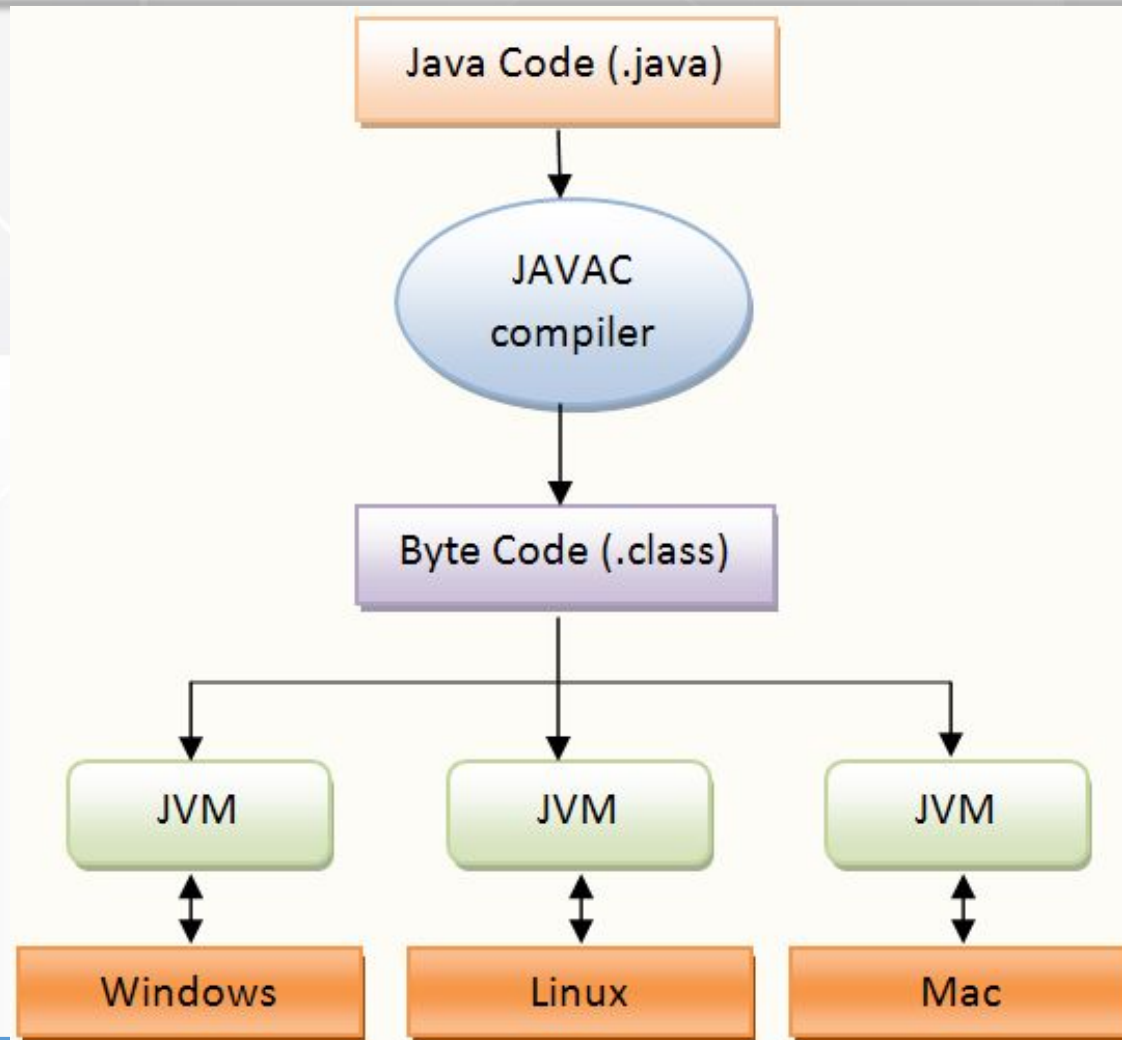
# Java: como funciona?

---

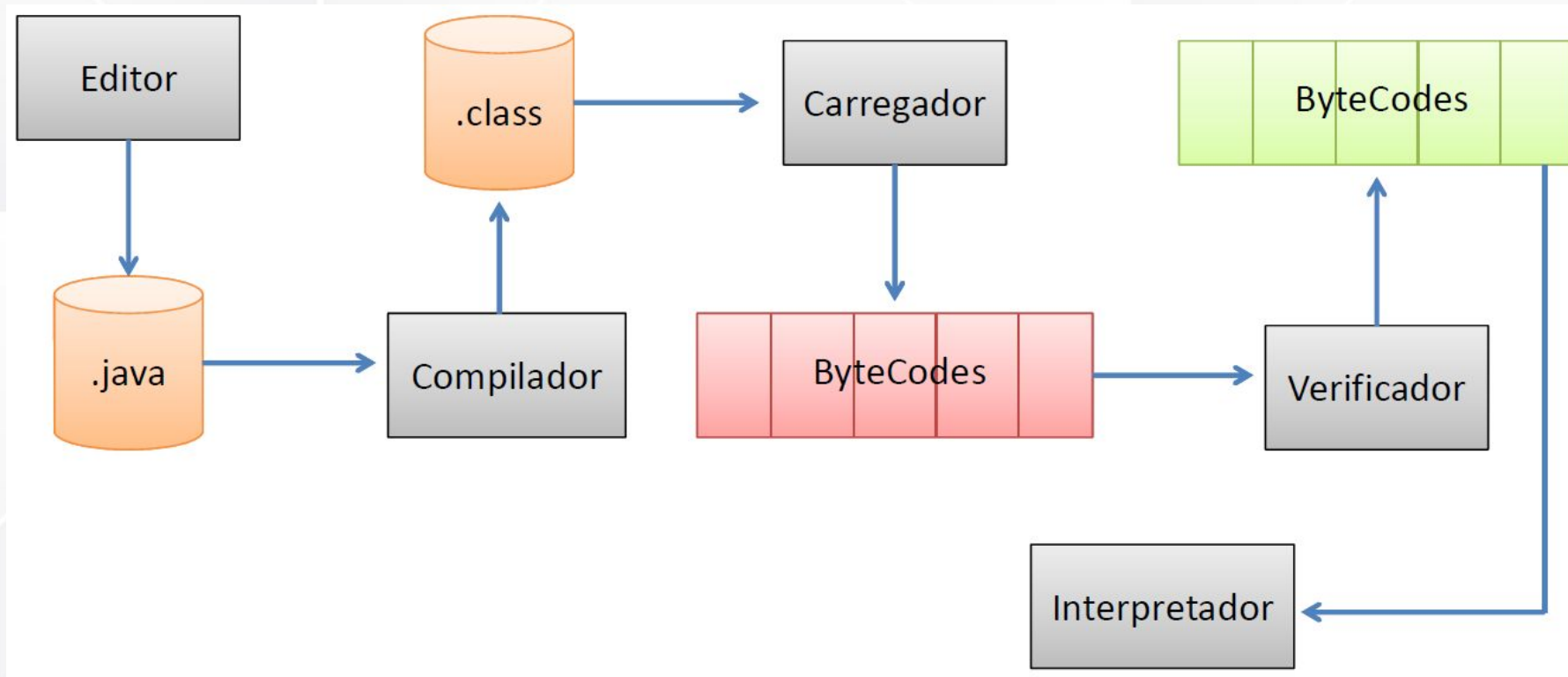
A execução de um programa java se dá em três ações distintas:

1. **Crie um programa fonte em Java.** Um programa fonte, contém texto escrito na linguagem de programação Java. Um programa Java, pode ser entendido por você e outros programadores. Qualquer editor de texto pode ser usado para criar e editar programas fonte.
1. **Compile o programa fonte em um arquivo "*bytecode*".** O compilador Java "*javac*", traduz o seu programa fonte em instruções que **a Máquina Virtual Java (JVM)** pode entender. O compilador armazena estas instruções num arquivo "*bytecode*".
1. **Execute o programa contido no arquivo "*bytecode*".** A Máquina Virtual Java, implementa o **interpretador** Java "*java*", Este interpretador, processa o arquivo "*bytecode*" gerando instruções equivalentes de forma que a plataforma em uso (Sistema Operacional e Hardware) consiga entender e executar estas instruções.

# Java: como funciona?

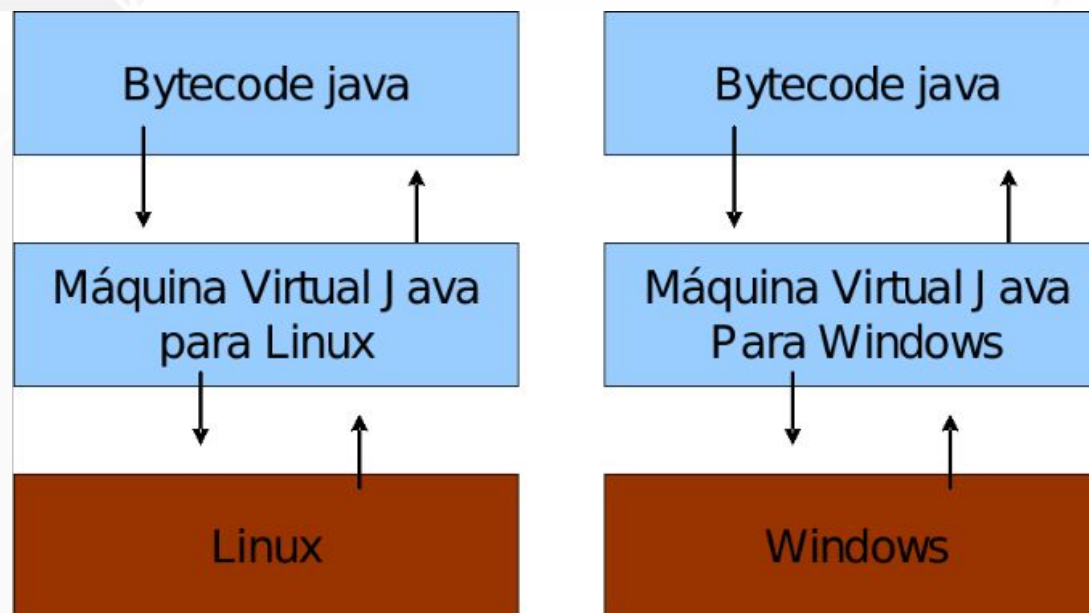


# Java: como funciona?



# Java: Máquina Virtual

- Java utiliza do conceito de **máquina virtual**, uma **camada extra** (entre o sistema operacional e a aplicação) responsável por "**traduzir**" o que uma aplicação deseja fazer para as respectivas chamadas do sistema operacional onde ela está rodando no momento.





# Java: Máquina Virtual

---

- Java Virtual Machine (**JVM**)
- Independência de sistema operacional
- Independência de plataforma
- Compilador javac
- “Write once, run anywhere”

# JVM, JRE e JDK

---

- **JVM** = apenas a virtual machine (download não existe, ela sempre vem acompanhada).
- **JRE = Java Runtime Environment**, ambiente de execução Java, formado pela JVM e bibliotecas (tudo que você precisa para executar uma aplicação Java).
- **JDK = Java Development Kit**: Ele é formado pela JRE somado a ferramentas, como o compilador.

# Algumas ferramentas do Java JDK

---

- o compilador Java (**javac**)
- o **interpretador** de aplicações Java (java)
- o interpretador de **applets** Java (appletviewer)
- **javadoc** (um gerador de documentação para programas Java)
- **Jar** (o manipulador de arquivos comprimidos no formato Java Archive)
- **jdb** (um depurador de programas Java)
- etc.



# Fundamentos Java

# Packages ou pacotes

---

Os arquivos Java serão **armazenados fisicamente** em uma **pasta**.

Cada arquivo representa uma classe Java.

Com o uso de packages podemos organizar de forma física algo lógico. (um grupo de classes em comum)

Para indicar que as definições de um arquivo fonte Java fazem parte de um determinado pacote, a primeira linha de código deve ser a declaração de pacote:

**package** nomedopacote;

Caso tal declaração não esteja presente, as classes farão parte do “pacote default”, que está mapeado para o diretório corrente

# Packages ou pacotes

---

Referenciando uma classe de um pacote no código fonte:

**import** nomedopacote.Xyz                      ou simplesmente  
**import** nomedopacote.\*

Com isso a classe **Xyz** pode ser referenciada sem o prefixo nome\_do\_pacote no restante do código.

A única exceção refere-se às classes do pacote java.lang.



# Tipos de dados primitivos

Classificação	Tipo	Descrição
Lógico	boolean	Pode possuir os valores true (verdadeiro) ou false (falso)
Inteiro	byte	Abrange de -128 a 127 (8 bits)
	short	Abrange de -32768 a 32767 (16 bits)
	int	Abrange de -2147483648 a 2147483647 (32 bits)
	long	Abrange de $-2^{63}$ a $(2^{63})-1$ (64 bits)
Ponto Flutuante	float	Abrange de $1.40239846^{-45}$ a $3.40282347^{+38}$ com precisão simples (32 bits)
	double	Abrange de $4.94065645841246544^{-324}$ a $1.7976931348623157^{+308}$ com precisão dupla (64 bits)
Caracter	char	Pode armazenar um caracteres unicode (16 bits) ou um inteiro entre 0 e 65535

# Tipos de dados primitivos do Java: Inteiros

---

Tipos de Dados Inteiros	Faixas
byte	-128 a +127
short	-32.768 a +32.767
int	-2.147.483.648 a +2.147.483.647
long	-9.223.372.036.854.775.808 a +9.223.372.036.854.775.807

- O valor default de todos é 0 (zero).

# Tipos de dados primitivos do Java: Ponto Flutuante

Tipos de Dados em Ponto Flutuante	Faixas
float	de $\pm 1.40282347 \times 10^{-45}$ até $\pm 3.40282347 \times 10^{+38}$
double	de $\pm 4.94065645841246544 \times 10^{-324}$ até $\pm 1.79769313486231570 \times 10^{+308}$

- **Exemplos:**
- 1.44E6 é equivalente a  $1.44 \times 10^6 = 1.440.000$ .
- 3.4254e-2 representa  $3.4254 \times 10^{-2} = 0.034254$ .
- O valor default de ambos é 0 (zero).

# Tipos de dados primitivos do Java: Caractere

---

- O tipo **char** permite a representação de **caracteres individuais**.
- Ocupa **16 bits internamente** permitindo até 32.768 caracteres diferentes.
- O valor default é 0 (zero).
- Caracteres de controle e outros caracteres cujo uso é reservado pela linguagem devem ser usados precedidos por **\**.

# Tipos de dados primitivos do Java: Caractere

- 

<code>\b</code>	backspace
<code>\t</code>	Tabulação horizontal
<code>\n</code>	newline
<code>\f</code>	form feed
<code>\r</code>	carriage return
<code>\"</code>	aspas
<code>\'</code>	aspas simples
<code>\\</code>	contrabarra
<code>\xxx</code>	o caracter com código de valor octal xxx, que pode assumir valores entre 000 e 377.
<code>\uxxxx</code>	o caráter com código de valor hexadecimal xxxx, que pode assumir valores entre 0000 e ffff.

# Tipos de dados primitivos do Java: Booleano

---

- É representado pelo tipo lógico **boolean**.
- Assume os valores **false** (falso) ou **true** (verdadeiro).
- O valor default é *false*.
- Ocupa 1 bit.



# Comentários

---

```
1 // comentário de uma linha
2
3 /* comentário de
4 múltiplas linhas */
5
6 /** comentário de documentação
7 * que também pode
8 * possuir múltiplas linhas
9 */
```

# Palavras reservadas

---

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

# Estrutura de um programa em Java

---

A estrutura de um programa em Java segue uma organização específica que inclui a definição de classes, métodos, e a execução de código dentro do método **main**. Elementos:

1. **Pacote (opcional):** Declara o pacote ao qual a classe pertence.
2. **Importações (opcional):** Importa outras classes ou pacotes que são usados no programa.
3. **Definição da Classe:** Declara a classe principal do programa.
4. **Método main:** O ponto de entrada do programa onde a execução começa.

# Estrutura de um programa em Java: Exemplo

```
// 1. Declaração de pacote (opcional)
package com.exemplo.meuprograma;
// 2. Importações (opcional)
import java.util.Scanner;
// 3. Definição da Classe

public class OlaMundo {

    // 4. Método principal
    public static void main(String[] args) {
        // Código a ser executado
        System.out.println("Olá, Mundo!");

        // Exemplo de uso do Scanner para entrada do usuário
        Scanner scanner = new Scanner(System.in);
        System.out.print("Digite seu nome: ");
        String nome = scanner.nextLine();
        System.out.println("Bem-vindo, " + nome + "!");

        // Fechar o Scanner
        scanner.close();
    }
}
```

# Declaração de Variáveis

---

Uma **variável não pode utilizar** como **nome** uma **palavra reservada** da linguagem.

Sintaxe:

Tipo nome1 [, nome2 [, nome3 [..., nomeN]]];

Exemplos:

**int** i;

**float** total, preco;

**byte** mascara;

**double** valormedio;

# Declaração de Variáveis

---

Embora não seja de uso obrigatório, existe a convenção padrão para atribuir nomes em Java, como:

- Nomes de **classes** são iniciados por letras **maiúsculas**;
- Nomes de **métodos**, **atributos** e **variáveis** são iniciados por letras **minúsculas**;
- Em nomes compostos, cada palavra do nome é iniciada por letra maiúscula, as palavras não são separadas por nenhum símbolo.

# Declaração de Variáveis: Exemplos

- Execute o programa a seguir:

```
public class ExemploVariaveis {  
    public static void main(String[] args) {  
        // Declarando variáveis inteiras  
        int idade = 25;  
        int anoAtual = 2024;  
  
        // Declarando variáveis de ponto flutuante  
        double altura = 1.75;  
        double precoProduto = 59.99;  
  
        // Declarando variáveis de caracteres  
        char inicial = 'J';  
  
        // Declarando variáveis booleanas  
        boolean isEstudante = true;  
        boolean isVerificado = false;  
  
        // Declarando variáveis de texto  
        String nomeCompleto = "João Silva";  
        String mensagem = "Olá, bem-vindo ao Java!"  
    }  
}
```

```
        // Imprimindo valores das variáveis  
        System.out.println("Idade: " + idade);  
        System.out.println("Ano Atual: " + anoAtual);  
        System.out.println("Altura: " + altura);  
        System.out.println("Preço do Produto: " + precoProduto);  
        System.out.println("Inicial: " + inicial);  
        System.out.println("É Estudante: " + isEstudante);  
        System.out.println("Está Verificado: " + isVerificado);  
        System.out.println("Nome Completo: " + nomeCompleto);  
        System.out.println("Mensagem: " + mensagem);  
    }  
}
```



# Escopo das Variáveis

---

É possível declarar variáveis a qualquer momento. Porém, dependendo de onde foi declarado, ela vai valer de um determinado ponto a outro.

```
// aqui a variável i não existe
int i = 5;
// a partir daqui ela existe
while (condicao) {
    // o i ainda vale aqui
    int j = 7;
    // o j passa a existir
}
// aqui o j não existe mais, mas o i continua dentro do escopo
```

# Operadores Aritméticos

+	Adição	$a+b$
-	Subtração	$a-b$
*	Multiplicação	$a*b$
/	Divisão	$a/b$
%	Resto da divisão inteira	$a\%b$
-	- Unário	$-a$
+	+ Unário	$+a$
++	Incremento unitário	$++a$ ou $a++$
--	Decremento unitário	$--a$ ou $a--$

# Operadores Aritméticos

---

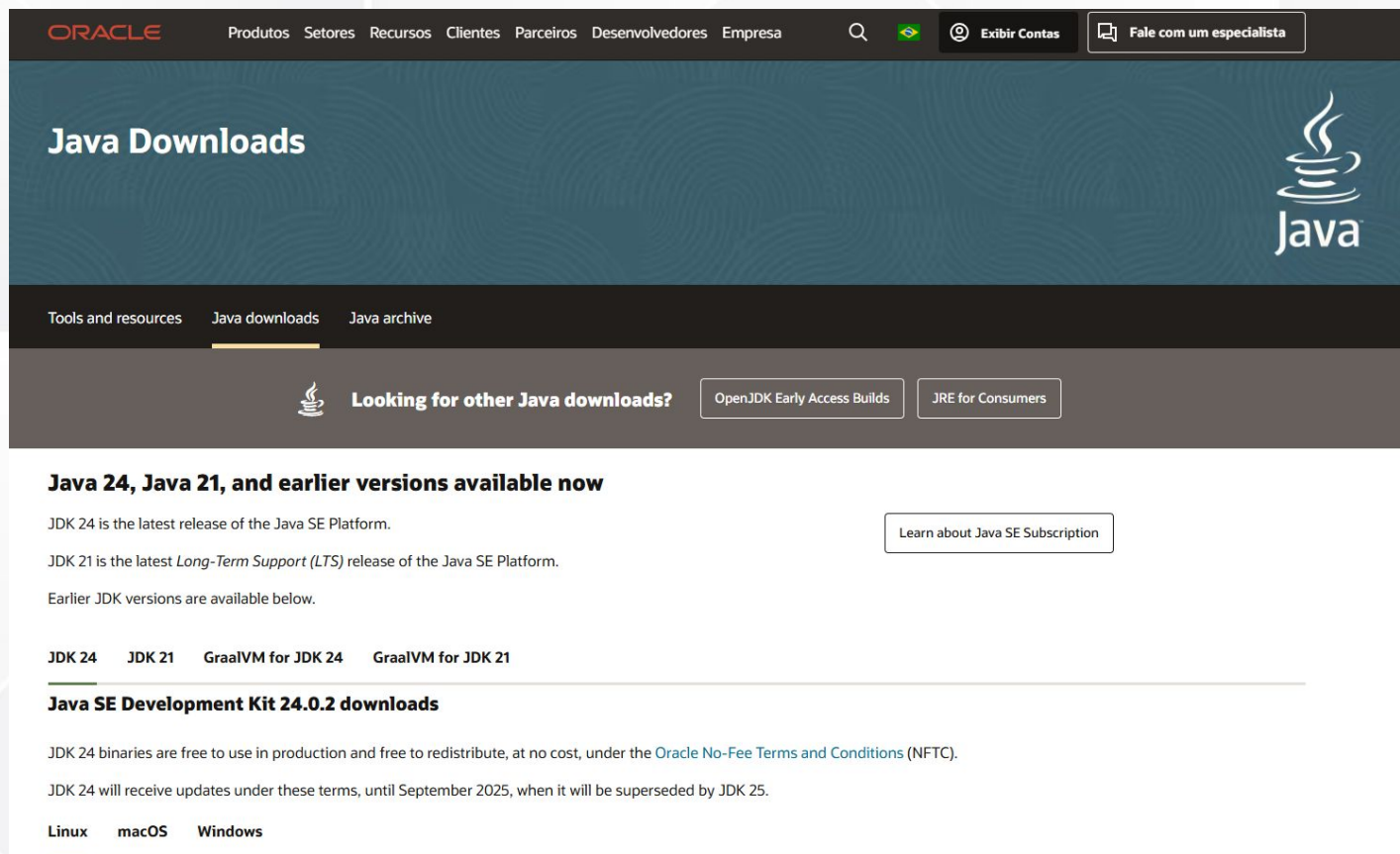
==	Igual	a==b
!=	Diferente	a!=b
>	Maior que	a>b
>=	Maior ou igual a	a>=b
<	Menor que	a<b
<=	Menor ou igual a	a<=b

# Operadores Lógicos



---

&&	E lógico ( <i>and</i> )	a&&b
	Ou lógico ( <i>or</i> )	a  b
!	Negação ( <i>not</i> )	!a


# Instalando Java




The screenshot shows the Oracle Java Downloads page. At the top is the Oracle logo and a navigation bar with links: Produtos, Setores, Recursos, Clientes, Parceiros, Desenvolvedores, Empresa, a search icon, a Brazilian flag, and buttons for 'Exibir Contas' and 'Fale com um especialista'. The main header is 'Java Downloads' with the Java logo on the right. Below this is a sub-navigation bar with 'Tools and resources', 'Java downloads' (highlighted), and 'Java archive'. A section titled 'Looking for other Java downloads?' includes links for 'OpenJDK Early Access Builds' and 'JRE for Consumers'. The main content area features the heading 'Java 24, Java 21, and earlier versions available now', followed by text about JDK 24 being the latest release, JDK 21 being the latest LTS release, and a link to 'Learn about Java SE Subscription'. It also lists links for 'JDK 24', 'JDK 21', 'GraalVM for JDK 24', and 'GraalVM for JDK 21'. A section titled 'Java SE Development Kit 24.0.2 downloads' explains that JDK 24 binaries are free to use and redistribute under the Oracle No-Fee Terms and Conditions (NFTC), and that updates will continue until September 2025. At the bottom, there are links for 'Linux', 'macOS', and 'Windows'.

ORACLE Produtos Setores Recursos Clientes Parceiros Desenvolvedores Empresa   Exibir Contas Fale com um especialista

## Java Downloads



Tools and resources Java downloads Java archive

 Looking for other Java downloads? OpenJDK Early Access Builds JRE for Consumers

### Java 24, Java 21, and earlier versions available now

JDK 24 is the latest release of the Java SE Platform. [Learn about Java SE Subscription](#)

JDK 21 is the latest *Long-Term Support (LTS)* release of the Java SE Platform.

Earlier JDK versions are available below.

[JDK 24](#) [JDK 21](#) [GraalVM for JDK 24](#) [GraalVM for JDK 21](#)

### Java SE Development Kit 24.0.2 downloads

JDK 24 binaries are free to use in production and free to redistribute, at no cost, under the [Oracle No-Fee Terms and Conditions](#) (NFTC).

JDK 24 will receive updates under these terms, until September 2025, when it will be superseded by JDK 25.

[Linux](#) [macOS](#) [Windows](#)

<https://www.oracle.com/br/java/technologies/downloads/>

# IDEs de desenvolvimento

---



<https://eclipse.org/downloads/>



<https://netbeans.org/downloads/>



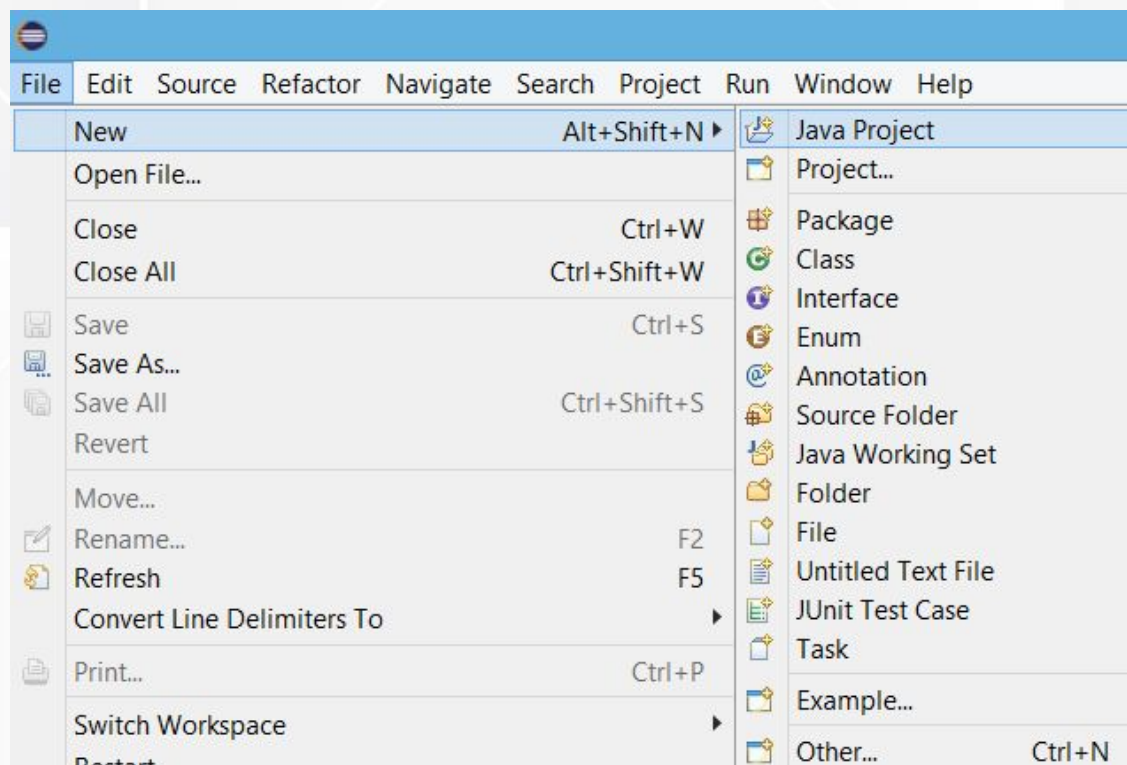
<https://www.jetbrains.com/pt-br/idea/>



<http://www.bluej.org/>

# Primeiro programa em Java(1/3)

Criando um novo projeto Java no Eclipse

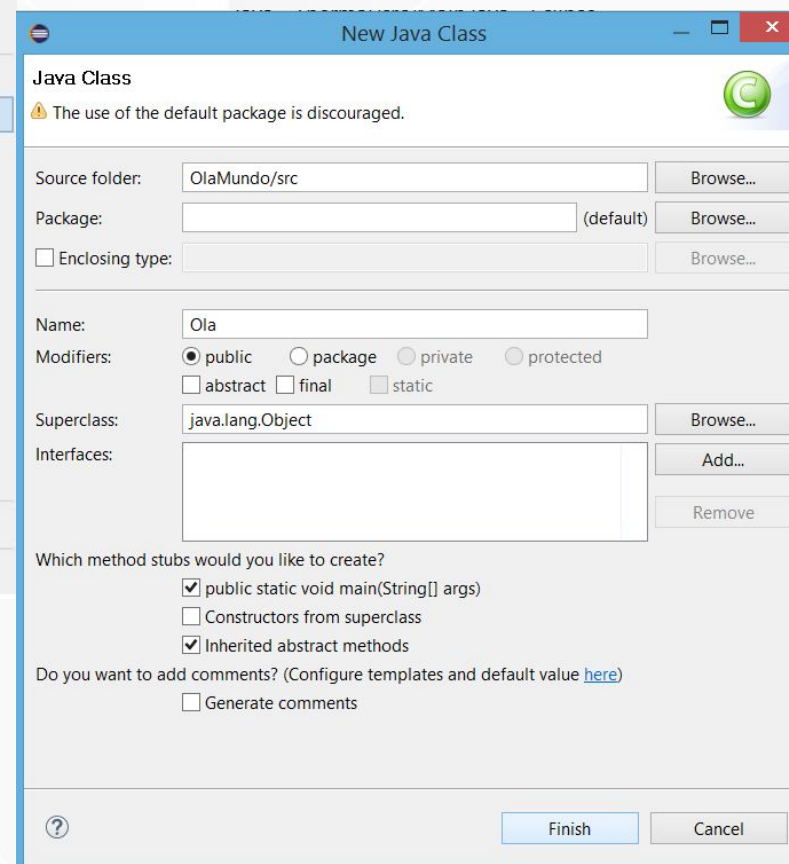
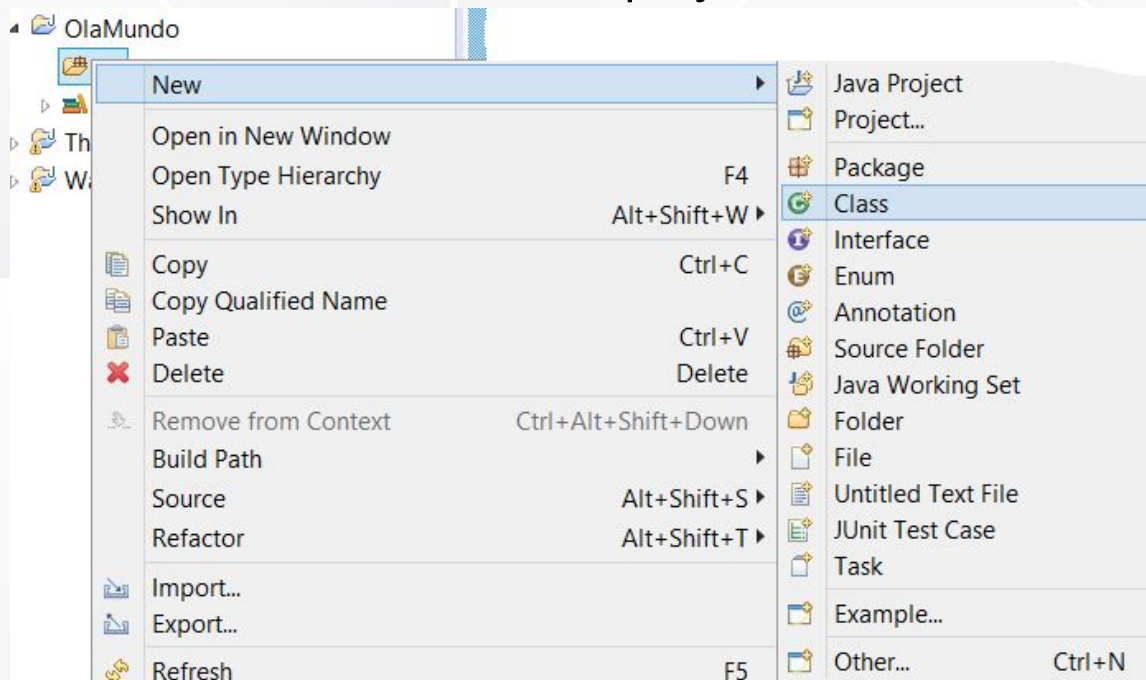


Na próxima janela adicione um nome ao projeto: “OlaMundo”



# Primeiro programa em Java(2/3)

## Adicionando uma classe ao projeto



# Primeiro programa em Java(3/3)

Adicione o código a seguir que imprime uma mensagem e mostra a data atual.

```
1 import java.util.*;
2
3 public class Ola {
4
5     public static void main(String[] args) {
6         System.out.println("Ola, Java");
7         Date d = new Date();
8         System.out.println("Date: "+d.toString());
9     }
10 }
```

Para compilar e executar o programa, selecione o projeto OlaMundo e clique no botão Run



# Exemplo 1: Somar Dois Números

```
import java.util.Scanner;

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

        System.out.print("Digite o primeiro número: ");
        int num1 = scanner.nextInt();

        System.out.print("Digite o segundo número: ");
        int num2 = scanner.nextInt();

        int soma = num1 + num2;

        System.out.println("A soma é: " + soma);
        scanner.close();
    }
}
```

## Exemplo 2: Calcular a Média de Três Números

```
import java.util.Scanner;

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

        System.out.print("Digite o primeiro número: ");
        double num1 = scanner.nextDouble();

        System.out.print("Digite o segundo número: ");
        double num2 = scanner.nextDouble();

        System.out.print("Digite o terceiro número: ");
        double num3 = scanner.nextDouble();

        double media = (num1 + num2 + num3) / 3;

        System.out.println("A média é: " + media);

        scanner.close();
    }
}
```

# Exemplo 3: Calcular o Perímetro de um Círculo

```
import java.util.Scanner;

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

        System.out.print("Digite o raio do círculo: ");
        double raio = scanner.nextDouble();

        double perimetro = 2 * Math.PI * raio;

        System.out.println("O perímetro do círculo é: " + perimetro);

        scanner.close();
    }
}
```

# Exemplo 3: Calcular o Perímetro de um Círculo

```
import java.util.Scanner;

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

        System.out.print("Digite o raio do círculo: ");
        double raio = scanner.nextDouble();

        double perimetro = 2 * Math.PI * raio;

        System.out.println("O perímetro do círculo é: " + perimetro);

        scanner.close();
    }
}
```

# Exercícios

---

Para cada um desses exercícios, siga estas etapas básicas:

- Solicite a entrada do usuário: Use a classe **Scanner** para ler os valores de entrada fornecidos pelo usuário.
  - Realize os cálculos necessários: Utilize as fórmulas matemáticas apropriadas para realizar os cálculos.
  - Exiba o resultado: Use **System.out.println** para exibir os resultados no console.
- 
1. Calcular a Área de um Triângulo
  2. Calcular o Volume de um Cilindro
  3. Dada uma temperatura em graus Fahrenheit, informe o valor correspondente em graus Celsius. [Dica:  $C = (F - 32) * (5 / 9)$ ]
  4. Dadas as medidas dos catetos de um triângulo retângulo, informe a medida da hipotenusa. [Dica: para calcular a raiz quadrada use a função `sqrt()`].



# Referências

## BIBLIOGRAFIA BÁSICA:

- DEITEL, Harvery M.. Java : como programar. 10ª ed. São Paulo: Pearson - Prentice Hall, 2017.
- BORATTI, Isaías Camilo. Programação Orientada a Objetos em Java : Conceitos Fundamentais de Programação Orientada a Objetos. 1ª ed. Florianópolis: VisualBooks, 2007.
- SIERRA, Kathy; BATES, Bert. Use a Cabeça! Java. 2ª ed. Rio de Janeiro: Alta Books, 2007.

