# JAVA

Edson Pimenta Almeida
Luís Augusto Starling Toledo

# TÓPICOS DE ABORDAGEM

- História
- Linha do tempo e genealogia
- Curisosidades
- Características gerais
- Linguagens relacionadas
- Aplicação
- Consideracões finais
- Apêndices

# JAMES GOSLING

James Gosling, criador da linguagem Java, desenvolveu-a para permitir compatibilidade entre plataformas. Com formação em ciência da computação, ele lançou Java na Sun Microsystems, tornando-a uma das linguagens mais populares da atualidade.

# CONTEXTO HISTÓRICO

- Projeto "Green": Desenvolvido por James Gosling e equipe para criar um sistema distribuído para o mercado de eletrônicos de consumo.
- Objetivo: Levar tecnologia de software moderna para fabricantes de eletrônicos de consumo.

Prioridades dos Consumidores:

- Produtos confiáveis, simples e de baixo custo.
- Sem bugs e fáceis de usar

# HISTÓRIA DE CRIAÇÃO

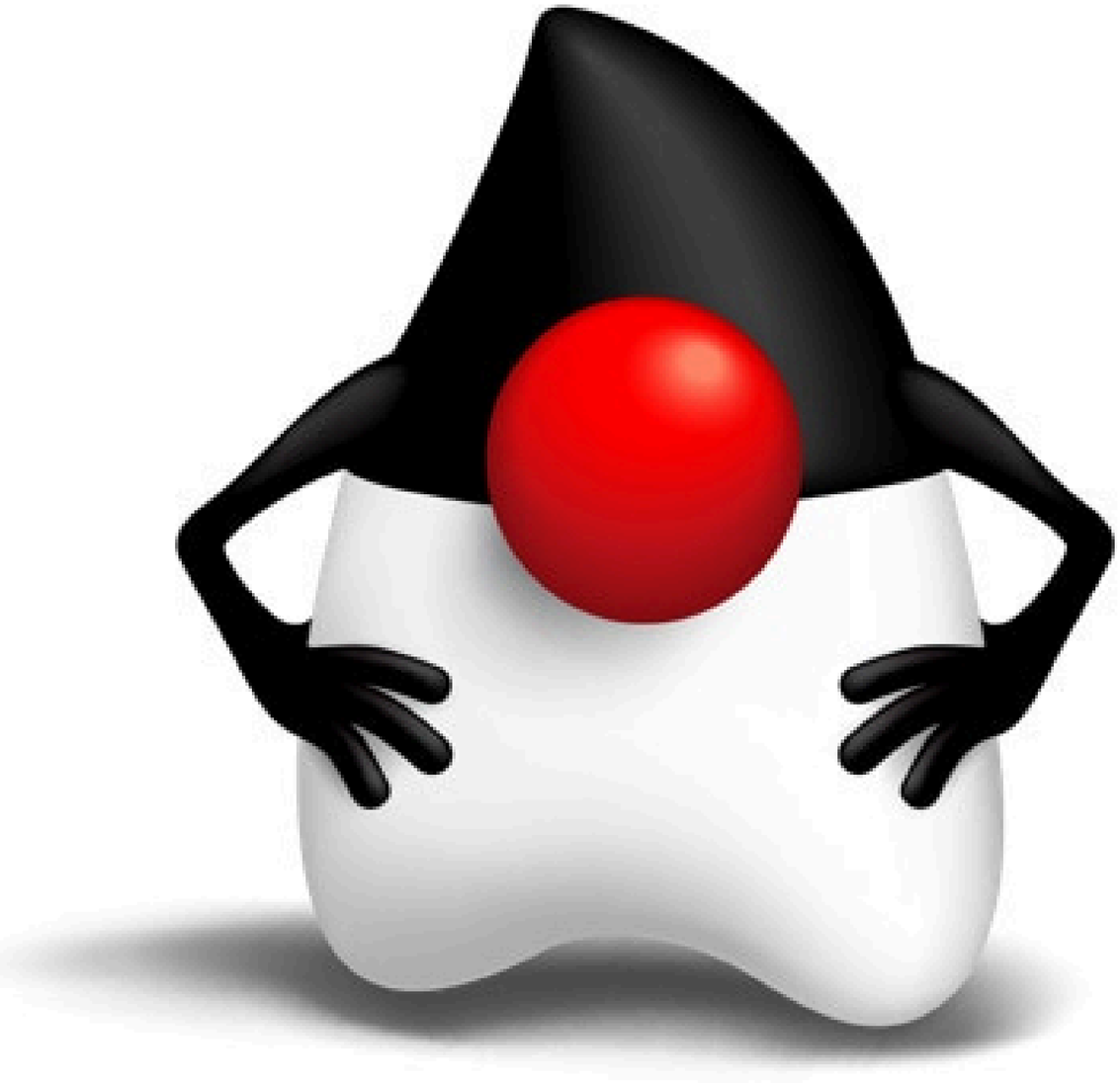
- Inicio com C++.
- Concepção de Oak em meados de 1991.
- Novo nome: Java
- O objetivo nunca foi 'Vamos encarar o C++'.
- Falha no processo de Tv Iterativa.
- Construção de um navegador HotJava.

# JAWA

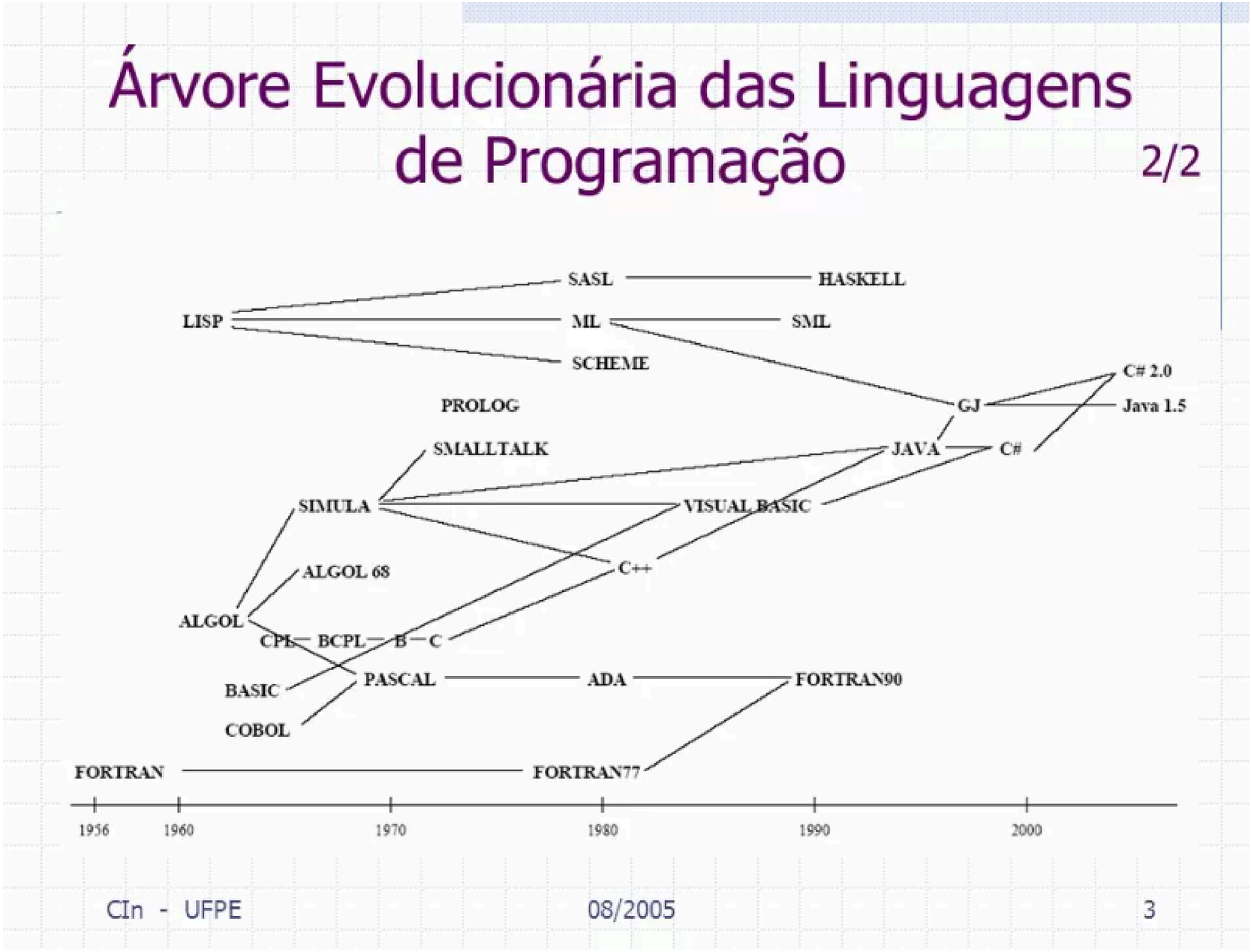
O nome da linguagem Java, foi inspirado na Ilha de Java. Uma ilha caracterizada pela origem de um dos cafés mais exóticos do mundo: o Java

# MASCOTE

Duke, mascote do Java, foi criado pela equipe do Green Project da Sun Microsystems como um "agente de software" interativo.

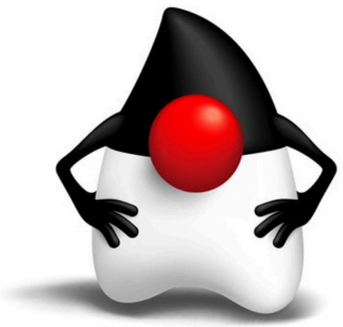


# LINHA GENEALÓGICA

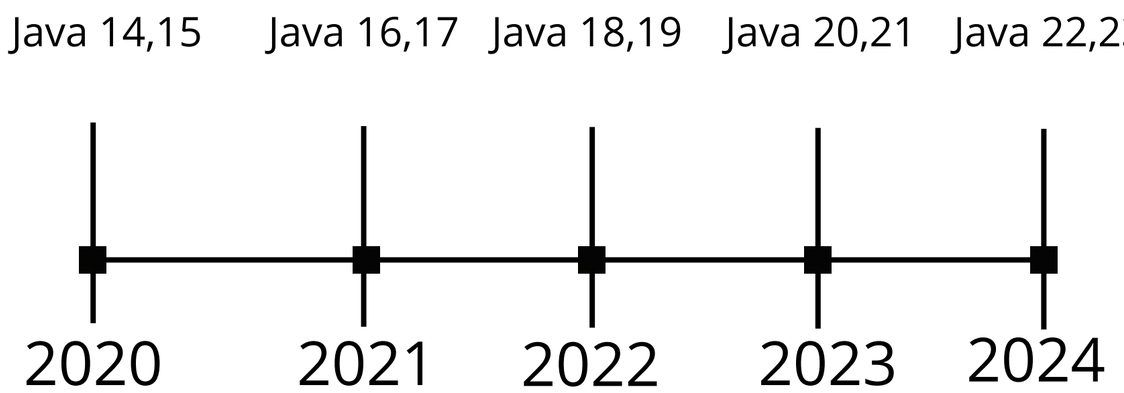


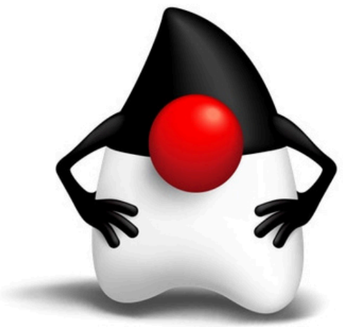
# LINHA DO TEMPO

| The green project | Java 1.0 | Java 1.2 | Java 1.3 | Java 1.4 | Java 1.5 | Java 1.6 | Java 8 | Java 9 | Java 11 | Java 12,13,14,15 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1991 | 1995 | 1998 | 2000 | 2002 | 2004 | 2006 | 2014 | 2017 | 2018 | 2019 |



# CONTIUAÇÃO...

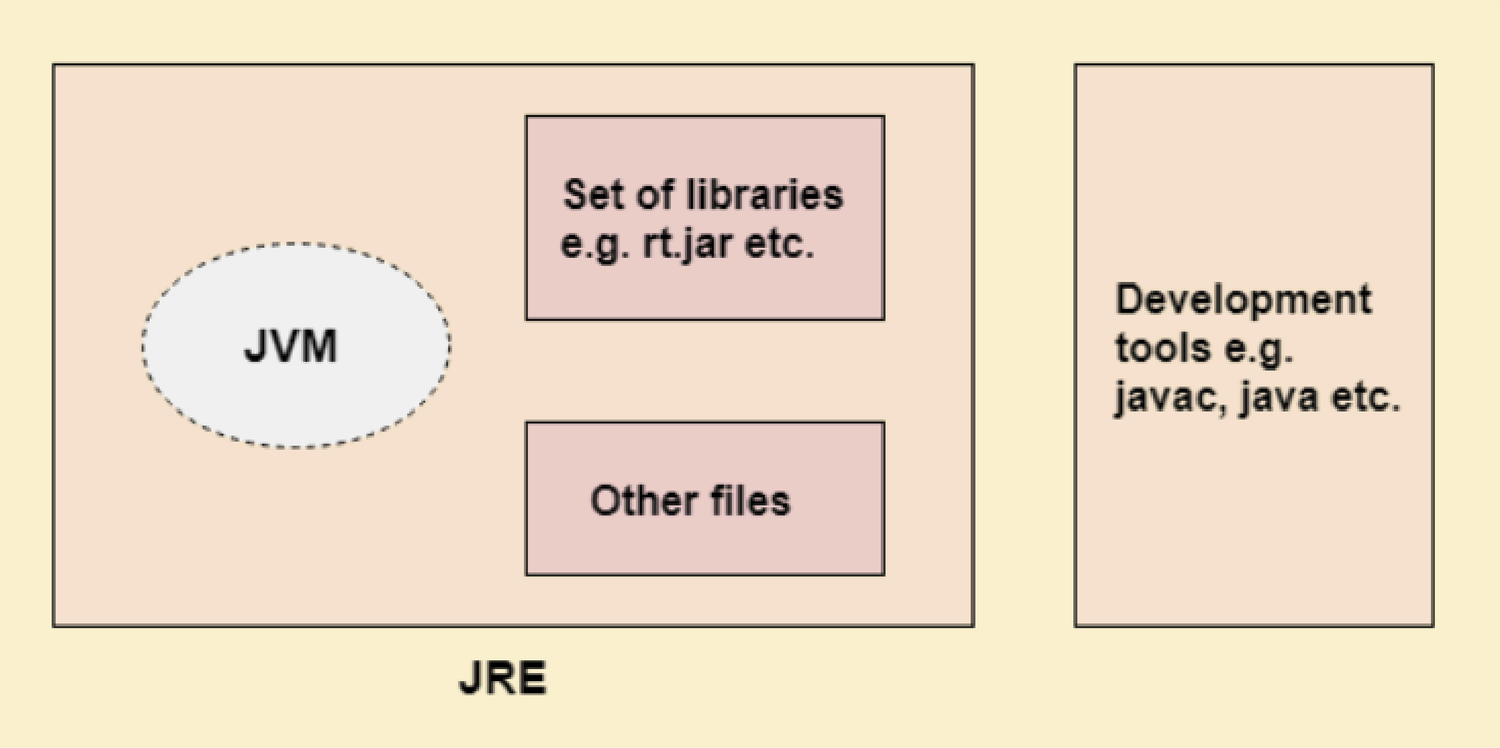

Java 14,15
Java 16,17
Java 18,19
Java 20,21
Java 22,23
2020
2021
2022
2023
2024



# JDK (Java Development Kit)
# JRE (Java Runtime Environment)

O JDK é o kit completo para desenvolvimento.
Serve para escrever, compilar e empacotar aplicativos Java.

O JRE é o ambiente necessário para executar aplicativos Java.
Contém a JVM e bibliotecas padrão para rodar o código.

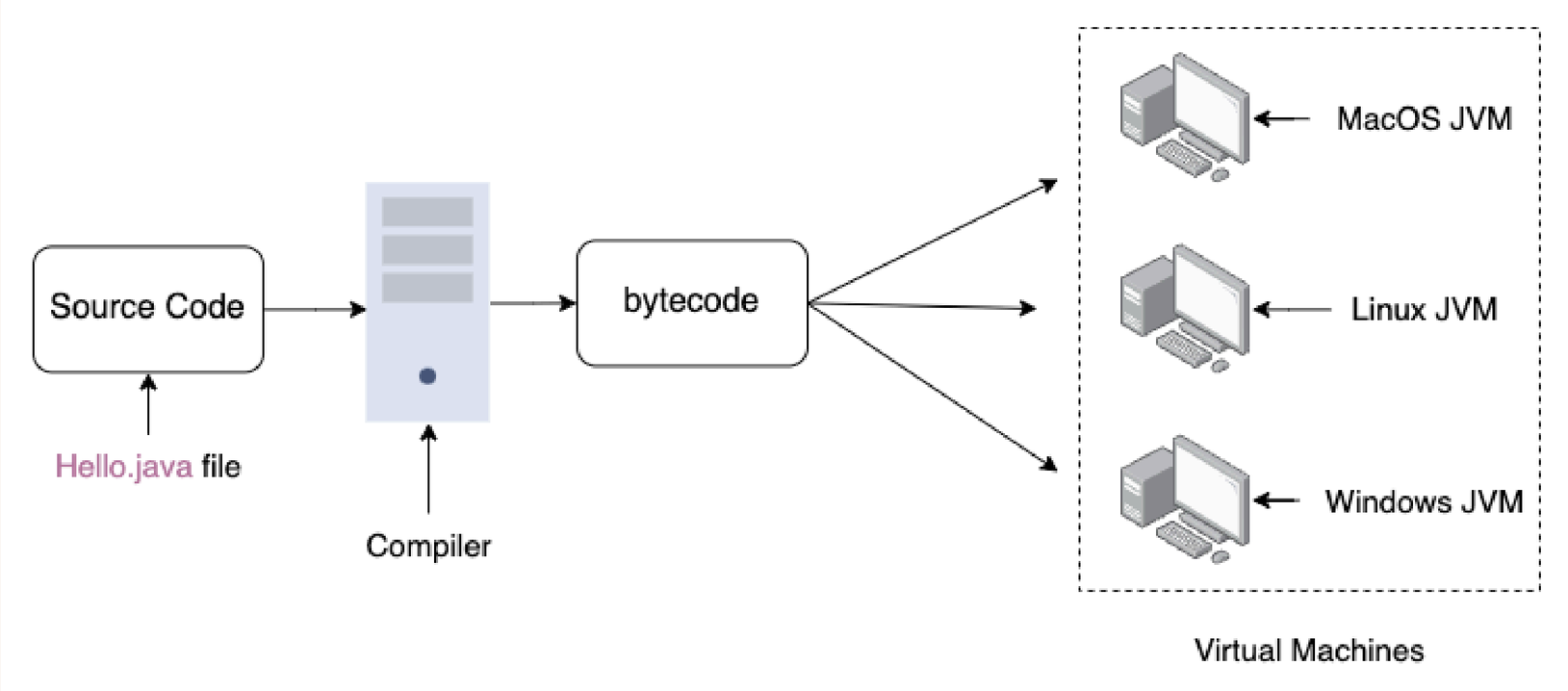


# JAVA VIRTUAL MACHINE

## JVM

É uma máquina virtual que executa bytecode Java, fornece um ambiente de execução para Java e outras linguagens compiladas em bytecode.

Compilação Just-In-Time (JIT):

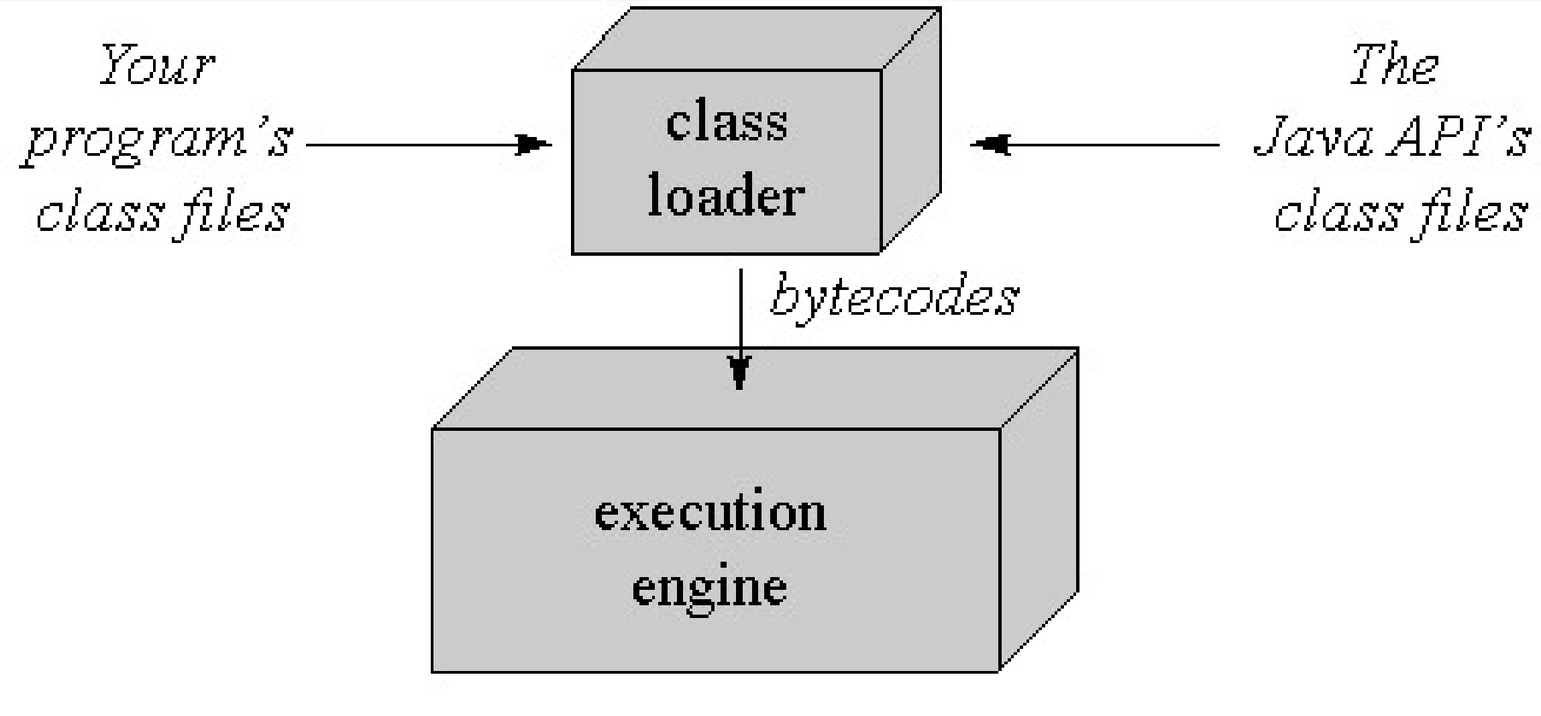


```java
public class IO{
    Run | Debug
    public static void main(String[] args){
        System.out.println(x:"Ola, Mundo!");
    }
}
```

```
PS C:\Users\PICHAU\exemploSeminario> javac IO.java
PS C:\Users\PICHAU\exemploSeminario> javap -c IO.class
Compiled from "IO.java"
public class IO {
  public IO();
    Code:
       0: aload_0
       1: invokespecial #1                  // Method java/lang/Object."<init>":()V
       4: return

  public static void main(java.lang.String[]);
    Code:
       0: getstatic     #7                  // Field java/lang/System.out:Ljava/io/PrintStream;
       3: ldc           #13                 // String Ola, Mundo!
       5: invokevirtual #15                 // Method java/io/PrintStream.println:(Ljava/lang/String;)V
       8: return
}
```

# JVM – CLASS LOADER

- Responsável por carregar classes na JVM durante a execução.
- Separa o carregamento de classes em três etapas:
  - Loading
  - Linking
  - Initialization




```
javac -cp /caminho/para/minhaBiblioteca.jar MeuPrograma.java
```

# GARBAGE COLLECTOR

## JAVA GC

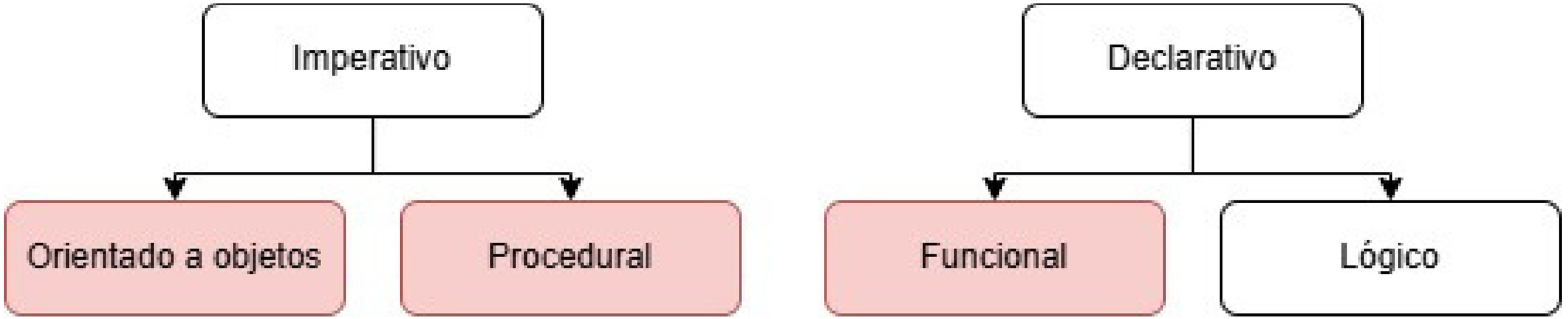
É um processo que roda na JVM e é responsável por liberar automaticamente a memória ocupada por objetos que não são mais utilizados pelo programa.

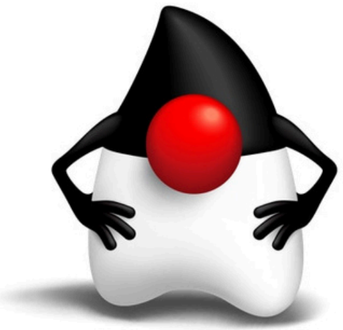
Generational Garbage Collection
Young Generation
Old Generation

java -Xms512m -Xmx2g -XX:+UseG1GC -classpath "meu/caminho/de/classes" MinhaAplicacao

-Xms512m define a memória inicial do heap para 512 MB.
-Xmx2g define a memória máxima do heap para 2 GB.
-XX:+UseG1GC instrui a JVM a usar o Garbage Collector G1.
-classpath especifica o caminho das classes usadas pelo aplicativo.

# PARADIGMAS


Imperativo
Orientado a objetos
Procedural
Declarativo
Funcional
Lógico



# PROGRAMAÇÃO FUNCIONAL

Pode ter funções puras, ou seja:

- Não possuem efeitos colaterais.
- Dependem apenas de seus parâmetros.
- Garantem resultados previsíveis para as mesmas entradas.

```java
import java.util.Random;

public class IO {

    // Função pura: sempre retorna o mesmo resultado para os mesmos argumentos
    // e não depende de variáveis externas nem altera o estado do programa.
    public static int soma(int a, int b) {
        return a + b;
    }

    // Função impura: esta função não é pura porque usa uma variável externa (Random)
    // para gerar um valor aleatório, então o resultado pode variar para a mesma entrada.
    public static int subtrai(int a, int b) {
        return new Random().nextInt() - a - b;
    }

    // Outra função impura: mesmo que multiplique corretamente os valores,
    // pode causar efeitos colaterais (como salvar o resultado externamente).
    public static int multiplica(int a, int b) {
        int resultado = a * b;

        // Imaginando que este código salva o resultado externamente,
        // ele causaria um efeito colateral, então a função não é pura.
        // OutroObjeto.salvar(resultado);

        return resultado;
    }
```

# EXPRESSÃO LAMBDA

As expressões lambda fornecem uma maneira concisa de expressar um método que pode ser passado como um argumento para outro método

```
interface SumCalculator {
    int calculate(int a, int b);
}
public class LambdaExample {
    public static void main(String[] args) {
        SumCalculator calculator = (a, b) -> a + b;
        System.out.println(calculator.calculate(5, 7)); // Output: 12
    }
}
```

**Output:**

```
12
```

# PROGRAMAÇÃO PROCEDURAL

Foco em Funções:

- O código é composto por funções que realizam tarefas específicas.

```java
public class IO {

    // Função para calcular a área de um retângulo
    public static float calcularArea(float largura, float altura) {
        return largura * altura; // Multiplica a largura pela altura para calcular a área
    }

    Run | Debug
    public static void main(String[] args) {
        // Definindo as variáveis de largura e altura
        float largura = 5.0f;
        float altura = 3.0f;

        // Chamando a função calcularArea e armazenando o resultado em 'area'
        float area = calcularArea(largura, altura);

        // Exibindo o resultado da área do retângulo
        System.out.printf(format:"Área do retângulo: %.2f\n", area);
    }
}
```

# PROGRAMAÇÃO ORIENTADA A OBJETOS

**Herança:** Um objeto adquire propriedades e comportamentos de um objeto pai

**Polimorfismo:** Execução de uma tarefa de diferentes maneiras

**Encapsulamento:** Agrupamento de código e dados em uma unidade única, protegendo os dados por meio de modificadores de acesso.

**Acoplamento:** Dependência entre classes

**Coesão:** Nível de um componente em realizar uma única tarefa bem definida

**Associação:** Relação entre objetos, inclui relações 1:1, N:1, 1:N, N:N

**Agregação:** Forma de associação onde um objeto contém outros

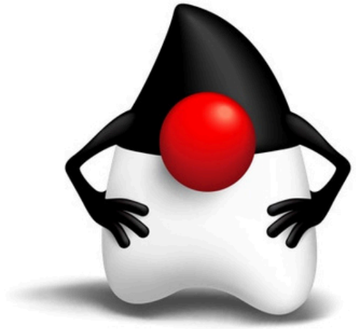
**Composição:** Forma de associação onde um objeto contém outros que não podem existir independentemente

```java
// 1. Herança
interface SomAnimal { void fazerSom(); }

// 2. Polimorfismo
class Animal implements SomAnimal {
    // 6. Encapsulamento
    private String nome;

    public Animal(String nome) { this.nome = nome; }
    public String getNome() { return nome; }

    // Implementando o método da interface
    @Override
    public void fazerSom() { System.out.println(x:"O animal faz um som."); }
}

class Cachorro extends Animal {
    public Cachorro(String nome) { super(nome); }

    @Override
    public void fazerSom() { System.out.println(getNome() + " diz: Au Au!"); }
}

class Gato extends Animal {
    public Gato(String nome) { super(nome); }

    @Override
    public void fazerSom() { System.out.println(getNome() + " diz: Miau!"); }
}
```

```java
// 3. Agregação e 4. Coesão
class Dono {
    private String nome;
    // 5. Associação
    private SomAnimal animal;

    public Dono(String nome) { this.nome = nome; }
    public void adotarAnimal(SomAnimal animal) { this.animal = animal; }

    public void mostrarAnimal() {
        if (animal != null) {
            System.out.println(nome + " tem um animal chamado " + ((Animal) animal).getNome());
            animal.fazerSom(); // Demonstra polimorfismo
        } else {
            System.out.println(nome + " não tem um animal.");
        }
    }
}
```

```java
// 8. Composição
class Zoo {
    private Dono dono;

    public Zoo(Dono dono) { this.dono = dono; }
    public void mostrarDono() { dono.mostrarAnimal(); }
}

// Classe principal para executar o código
public class IO {
    Run | Debug
    public static void main(String[] args) {
        // Criando um dono
        Dono dono = new Dono(nome:"Carlos");

        // Criando animais
        SomAnimal cachorro = new Cachorro(nome:"Rex");
        SomAnimal gato = new Gato(nome:"Mia");

        // 7. Agregação
        dono.adotarAnimal(cachorro); // Adotando um cachorro
        dono.mostrarAnimal(); // Mostra o animal e o som que ele faz

        dono.adotarAnimal(gato); // Trocar o animal (agregação)
        dono.mostrarAnimal(); // Mostra o animal e o som que ele faz

        // Criando um zoológico
        Zoo zoo = new Zoo(dono);
        zoo.mostrarDono(); // Mostra informações do dono e do animal
    }
}
```

# LINGUAGENS RELACIONADAS



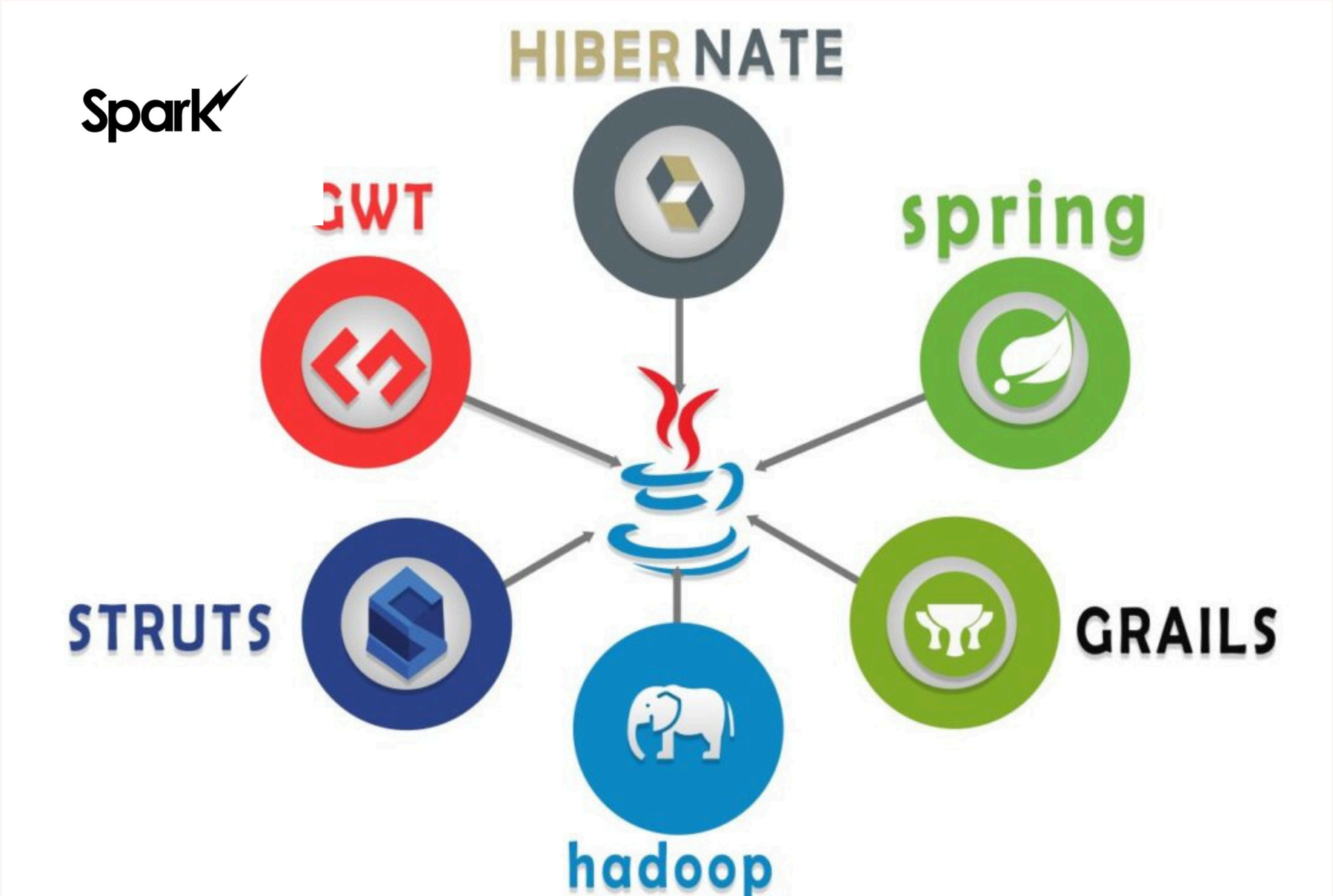
**C++:**

- Java foi criado como uma evolução da linguagem C++.
- Objetivo de Aperfeiçoamento:
- O foco foi eliminar desvantagens do C++.
- Java é projetada para atender a necessidades específicas de aplicações que operam em diferentes plataformas e redes.
- Apesar das melhorias, Java e C++ compartilham muitas características, especialmente na sintaxe.

Simula:

- Uma das primeiras linguagens com classes, objetos, pesquisa dinâmica, subtipagem e herança

# APLICAÇÃO



# CONSIDERAÇÕES FINAIS

Java se tornou uma das linguagens de programação mais influentes desde sua criação na década de 1990, destacando-se pela portabilidade e pela filosofia "Write Once, Run Anywhere" (WORA). Com inovações como a programação orientada a objetos e recursos modernos, como expressões lambda e a API de streams, Java facilita o desenvolvimento de software. Apesar de suas complexidades, sua forte comunidade e o suporte da Oracle garantem sua relevância. Assim, Java permanece uma escolha robusta para diversas aplicações, consolidando-se como uma ferramenta essencial no desenvolvimento de software.

# APENDICES

Edson:
Tive muita aprendizagem ao desenvolver esse trabalho. Aprendi sobre a evolução de Java e suas principais características, além de ver como a linguagem se adaptou ao mercado desde o inicio do projeto “Green”.

Luís:
A experiência de desenvolver este trabalho foi positiva. A pesquisa me fez entender melhor a importância de Java na programação moderna e as inovações que surgiram nas últimas versões.

# BIBLIOGRAFIA


GOSLING, James. Feel of Java. 2016. Disponível em: https://evink.win.tue.nl/education/avp/pdf/feel-of-java.pdf. Acesso em: 01 nov. 2024.

Javatpoint. Difference Between Abstract Class and Interface. 2024. Disponível em: https://www.javatpoint.com/difference-between-abstract-class-and-interface. Acesso em: 30 out. 2024.

Thiago. Java: história e principais conceitos. 2021. Disponível em: https://www.devmedia.com.br/java-historia-e-principais-conceitos/25178. Acesso em: 30 out. 2024.

VIEIRA, Rafael. Programação funcional no Java. 2020. Disponível em: https://medium.com/@nvieirarafael/programação-funcional-no-java-2a005964cb20. Acesso em: 29 out. 2024.

CARVALHO, Pedro. Linguagens de programação. 2022. Disponível em: https://arvoregenealogica.online/blog/linguagens-de-programacao/. Acesso em: 03 out. 2024.

O'CONNELL, Michael. Java: a retrospectiva. 1995. Disponível em: http://sunsite.uakom.sk/sunworldonline/swol-07-1995/swol-07-java.html. Acesso em: 01 nov. 2024.

EVANS, Ben. Java: The Legend. 2016. Disponível em: https://pepa.holla.cz/wp-content/uploads/2016/10/java-the-legend.pdf. Acesso em: 03 nov. 2024.

BYOUS, JON. A história do Java. 1998. Disponível em: https://web.archive.org/web/20080530073139/http://java.sun.com/features/1998/05/birthday.html. Acesso em: 29 out. 2024.

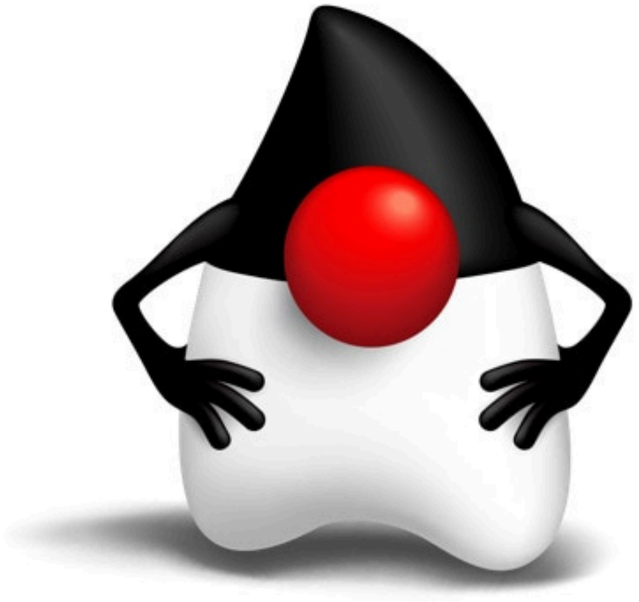
ORACLE. Duke: O mascote do Java. 2024. Disponível em: https://www.oracle.com/br/java/duke/. Acesso em: 29 out. 2024.

ORACLE. Breve história do Java. 2021. Disponível em: https://www.oracle.com/a/ocom/docs/dc/ww-brief-history-java-infographic.pdf. Acesso em: 28 out. 2024.

CANALLE, Gabrielle e Araújo, Everton. Do C++ para o Java: conheça as diferenças e principais características. 2021. Disponível em: https://www.devmedia.com.br/do-c-c-para-o-java-conheca-as-diferencas-e-principais-caracteristicas/26773. Acesso em: 1 out. 2024.

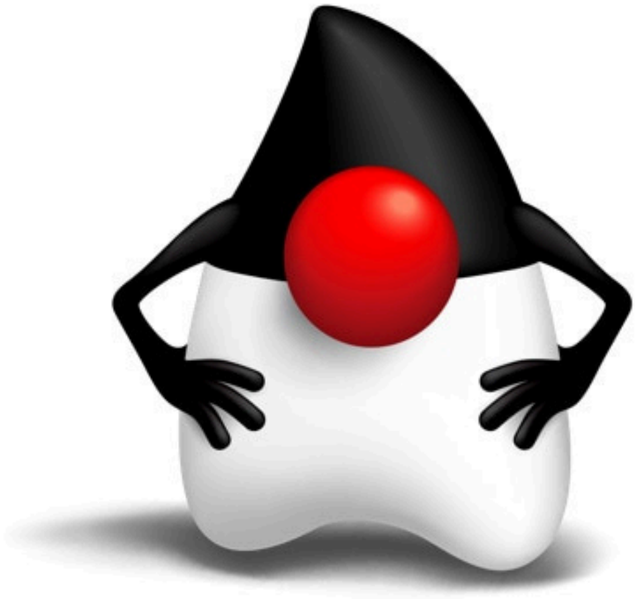
# PRÁTICA

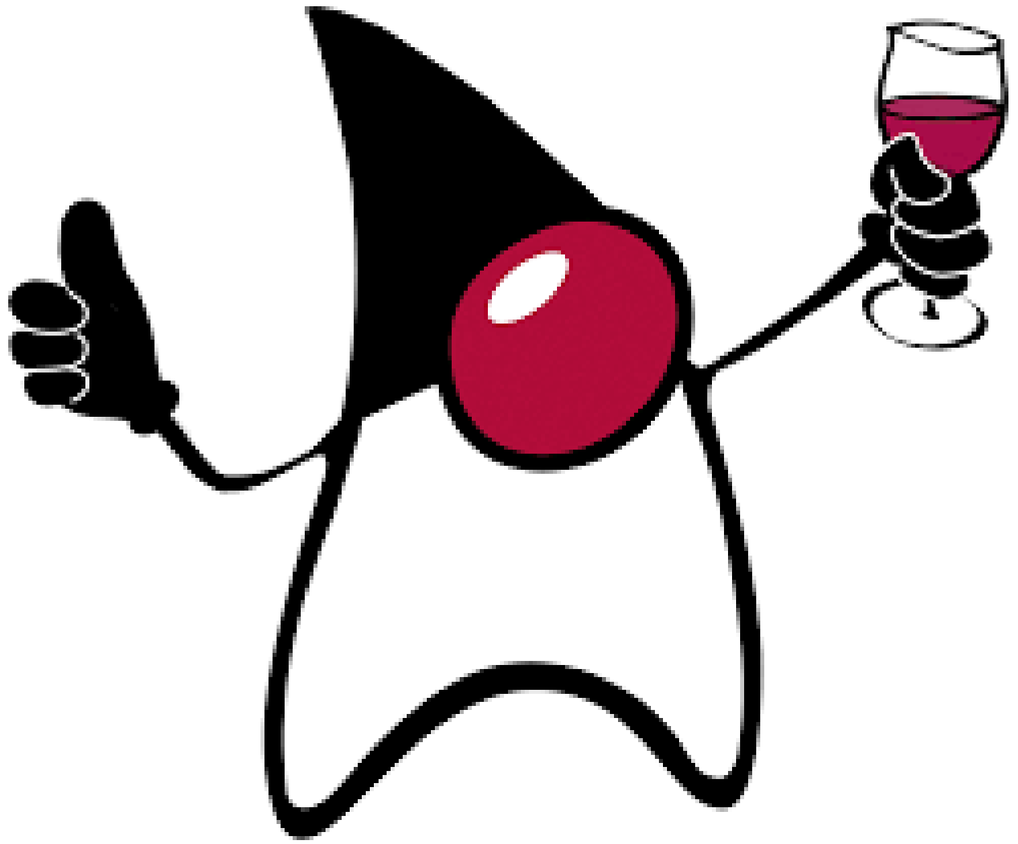
- Java em desenvolvimento web.
- Aplicação com interface gráfica
- Java Multithreading



# TUTORIAL

- https://www.youtube.com/watch?v=o91uQ7nMHxU&ab_channel=DoBackaoFront-Programa%C3%A7%C3%A3oFullStack





# FIM!