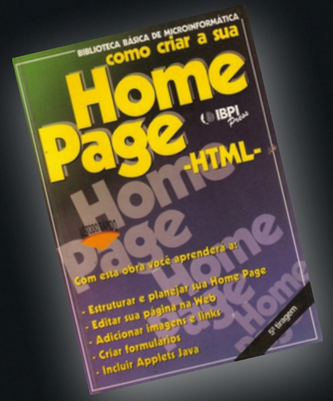
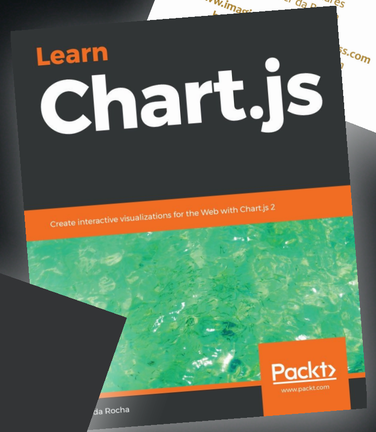
16 de outubro de 2025

Ficou mais fácil programar em

Java 25

Helder da Rocha
www.argonavis.com.br
helder@argonavis.com.br





Helder da Rocha

Quem?

Tecnologia * Ciência * Arte
HTML & tecnologias Web desde 1995
Autor de cursos e livros sobre
Java, XML e tecnologias Web

argonavis.com.br

@helderdarocha

helder@argonavis.com.br

Helder da Rocha

Java ...

... desde 1995

O que vamos discutir **hoje**?

A pergunta de sempre: "Ainda vale a pena investir em **Java**?"

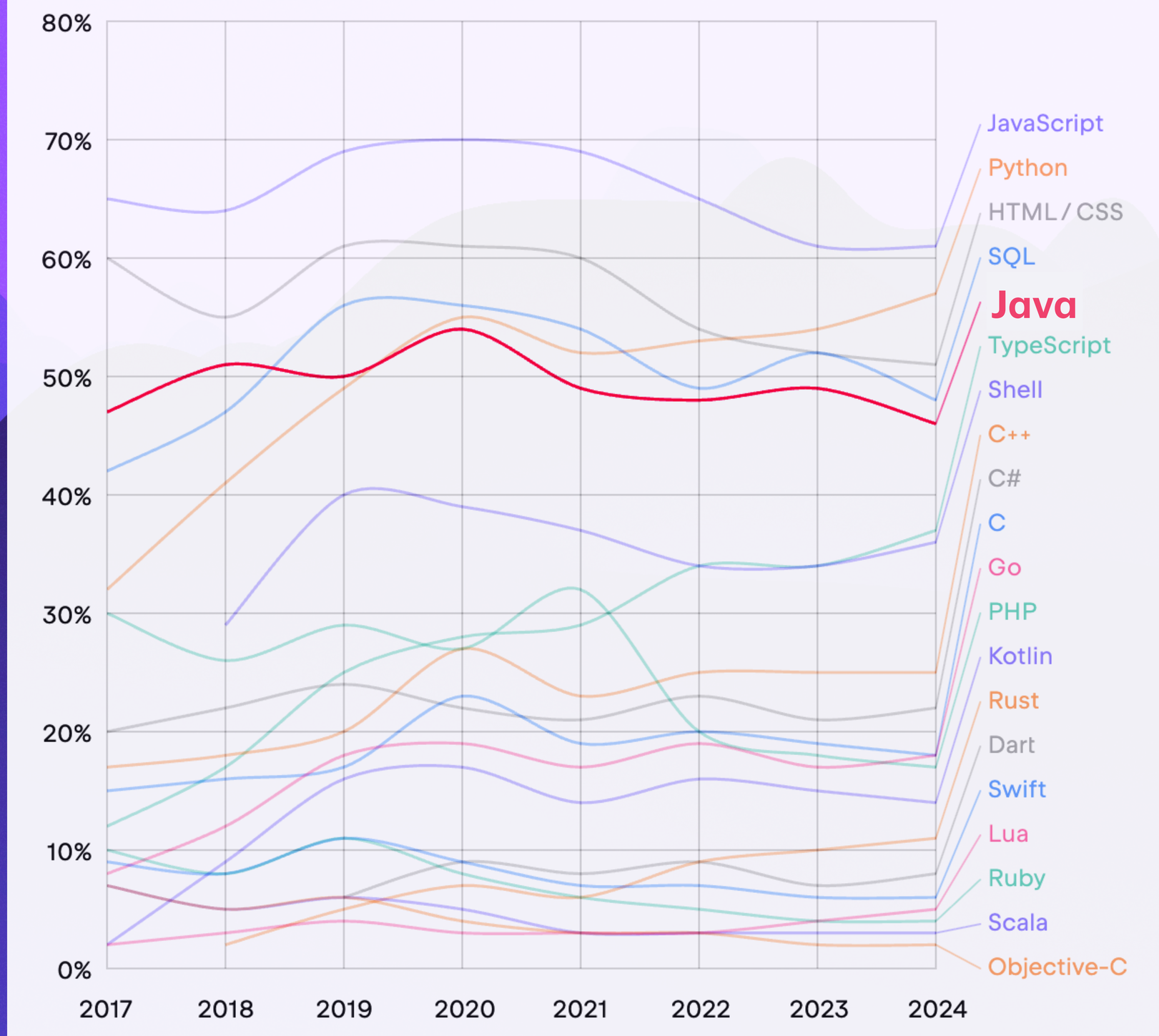
O que mudou nos últimos anos: Java ficou **melhor**?

O **Java 25** foi lançado em 16-09-2025. **Java 8** tem 11 anos.
Qual Java você usa?

Destaques para programadores,
desde o último LTS (Java 21) até o Java 25

Recursos em **prévia** no Java 25

Alguém ainda programa em Java?



https://www.jetbrains.com/lp/devecosystem-2024/

E... é fácil aprender?

Primeiro programa

```
print("Hello, Monty Python!");
```

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

```
helderdarocha — -zsh — 60×14
~% java HelloWorld

Hello Java!

~% py HelloWorld.py

Hello Monty Python!
```

É fácil?

Segundo programa

```python
msg = input("Digite uma mensagem: ")
print(msg)
```


```java
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class MySecondJavaProgram {
    public static void main(String[] args) {
        BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
        System.out.print("Digite uma mensagem: ");
        String message = null;
        try {
            message = reader.readLine();
        } catch (IOException e) {
            throw new RuntimeException(e);
        } finally {
            try {
                reader.close();
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
        }
        System.out.println(message);
    }
}
```

... depois de compilar:

```
helderdarocha — -zsh — 60×14

~% java HelloWorldInput.java

Digite uma mensagem: Здраствуй Землья!
Здраствуй Землья!
```

4 grandes projetos do OpenJDK

Promovendo a evolução da linguagem Java moderna

Panama

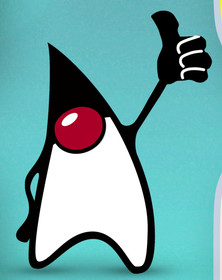
Melhorar a **interoperabilidade** entre o Java e código nativo, com impacto na computação de alto desempenho, ML, IA, jogos, etc.

concluído **70%**

Amber

Tornar mais **produtivo**, **fácil** e **agradável** escrever programas em Java

concluído **95%**



Valhalla

Modernizar o modelo de memória da JVM para aproveitar da melhor forma os recursos de hardware modernos

concluído **45%**

Loom

Simplificar e tornar mais **leves** e **eficientes** programas que usam concorrência e threads

concluído **80%**

Simplificando o Java com JEPs do Projeto Amber

OpenJDK

- Declaração com `var` (**Java 10**)
- `var` em lambdas (**Java 11 LTS**)
- Expressões em `switch` (**Java 14**)
- Blocos de texto `"""` (**Java 15**)
- Records (**Java 16**)
- Pattern matching com `instanceof` (**Java 16**) e `switch` (**Java 19**)
- `sealed` classes (**Java 17 LTS**)
- Record patterns (**Java 19**)
- Coleções sequenciadas (**Java 21 LTS**)
- `_` Variáveis sem nome (**Java 22**)
- `import module` (**Java 25 LTS**)
- Arquivos-fonte compactos; `void main()` (**Java 25 LTS**)
- Flexibilização do corpo de construtores (**Java 25 LTS**)

Java 25 PREVIEW 3

- Tipos primitivos em patterns, `instanceof` e `switch`

Descontinuado

- Template + multiline strings

Design multi-paradigma

Evolução do Java moderno suporta diferentes estratégias de design

Programação orientada a objetos

Polimorfismo
Herança
Encapsulamento
Abstração
Modularidade
Pacotes, módulos, classes e objetos

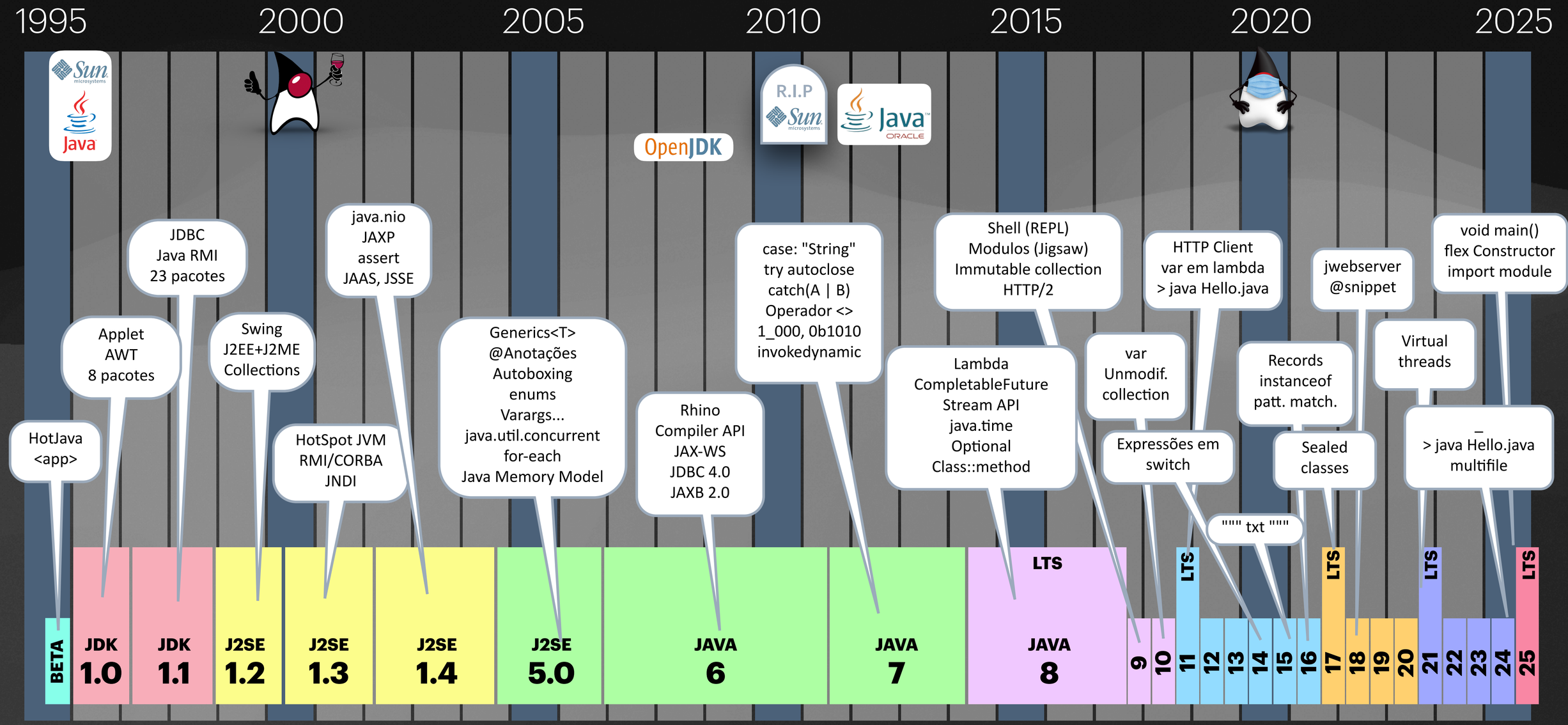
Programação funcional

Imutabilidade e funções puras
Funções de ordem superior
Lambda
Referências de métodos
API de streams

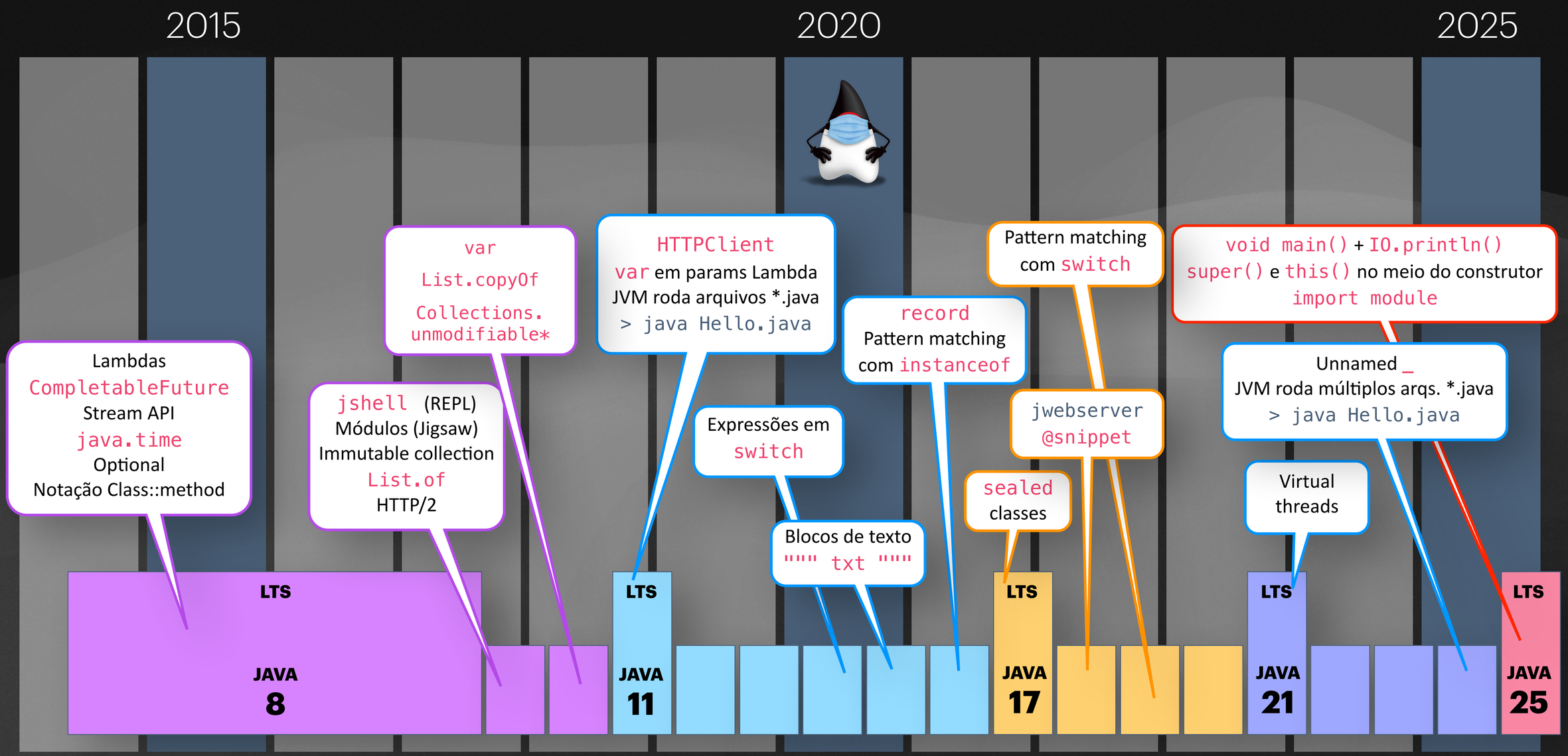
Programação orientada a dados

Dados imutáveis
Pattern matching
Tipos de dados algébricos
Records (**AND** type)
Classes e interfaces seladas (**OR** type)

Timeline: **30 anos** de Java



Timeline OpenJDK pós-**Java 8 LTS** (2014)

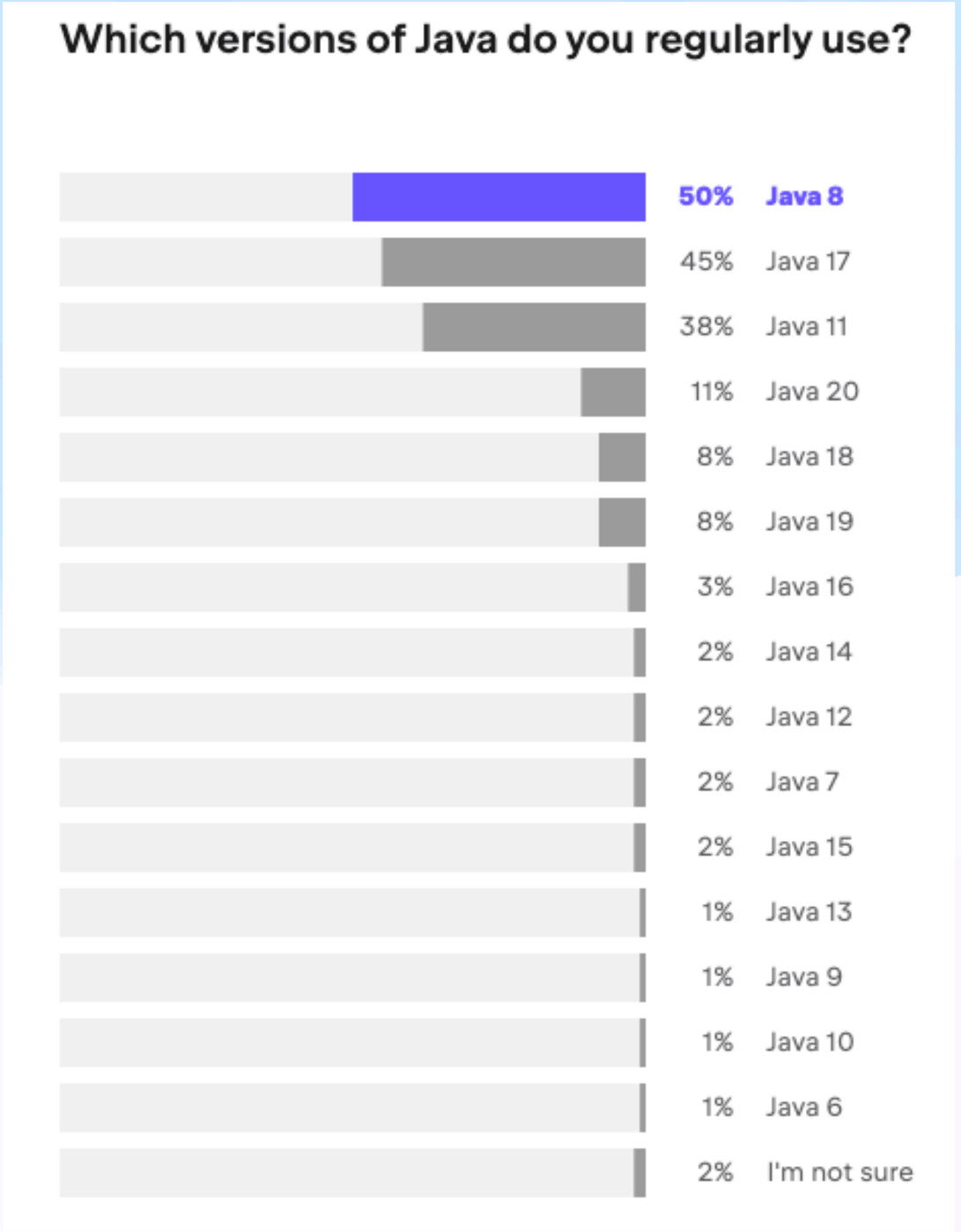


https://www.jetbrains.com/lp/devecosystem-2023/java/

Qual Java você usa?

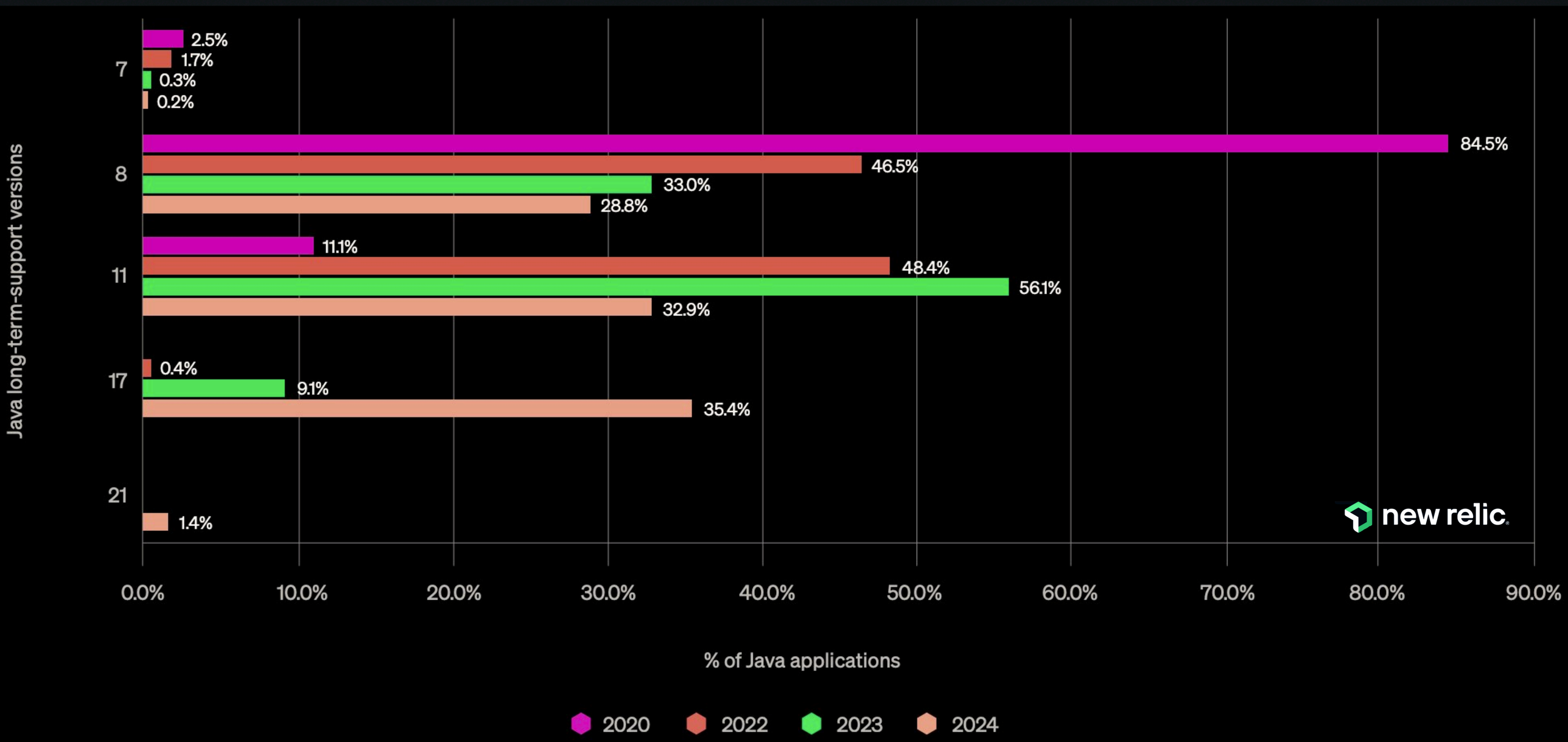
Java Developer Ecosystem 2023

JETBRAINS



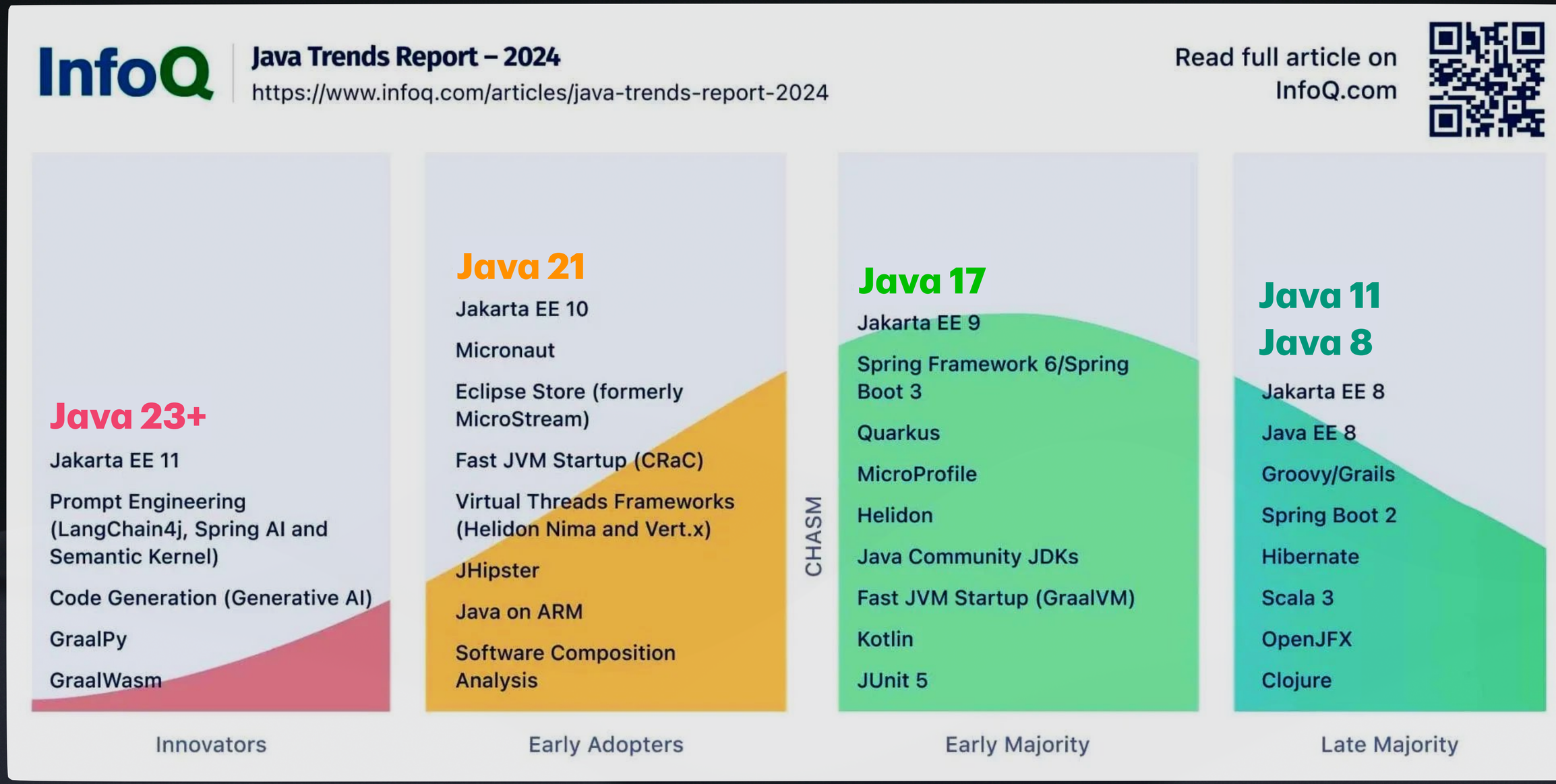
Qual **Java** você usa?

(% de aplicações Java em 2024)



Qual **Java** você usa?

2024 (baseado nas pesquisas da NewRelic e JetBrains)



Estado atual (**2025**) - estimativa (MUNDO)

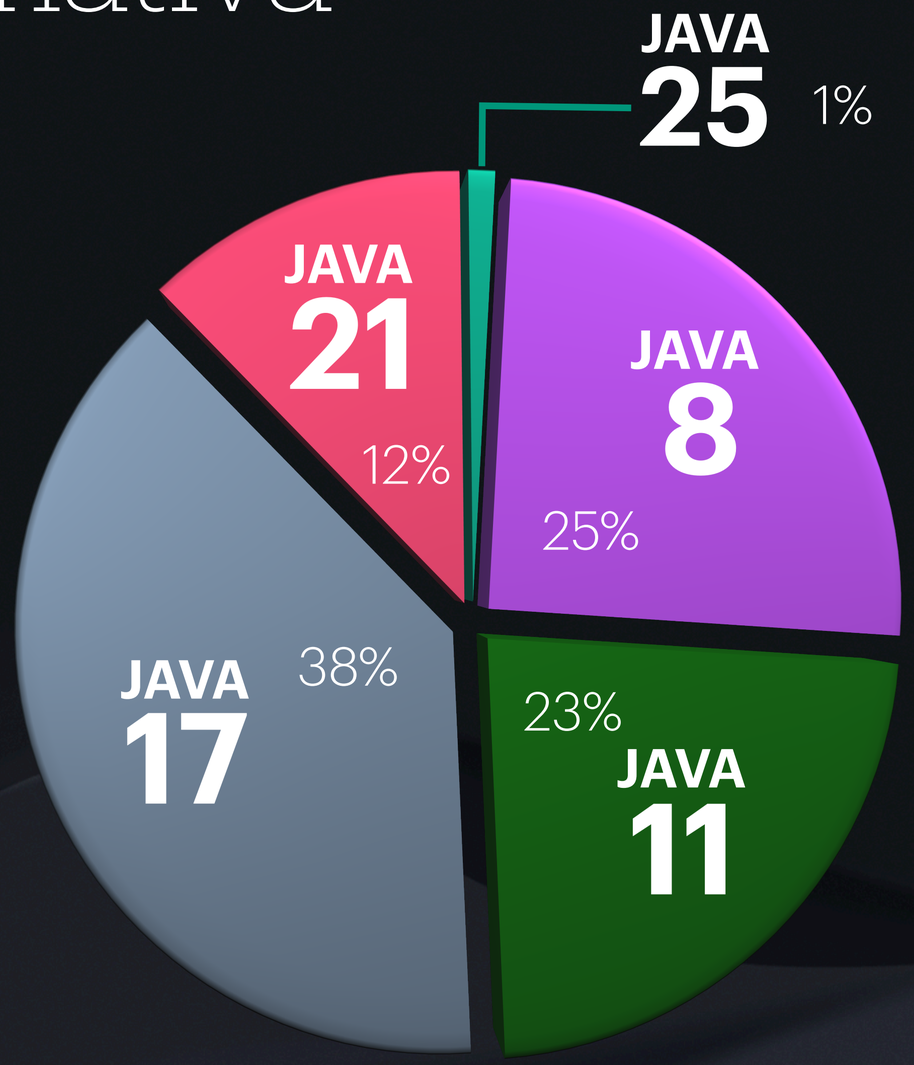
Considerando as linguagens Java **LTS**

~100% dos programadores Java sabem **Java 8**

~75% sabem **Java 11**

~50% sabem **Java 17**

~10% sabem **Java 21**



Estimativa de adoção em **2025**

Estimativas obtidas a partir de dados da New Relic (aplicações), JetBrains (usuários) + ajuda do Anthropic Claude

Java 22 a **Java 25 LTS**

Destaques desde o **Java 21 LTS** (2023)

- **Java 22**: Unnamed variables and patterns (Project **Amber**)
- **Java 25 LTS**:
  - Module import declarations (Project **Amber**)
  - Flexible Constructor Bodies (Project **Amber**)
  - Scoped values (Project **Loom**)
  - Compact source files and instance main methods (Project **Amber**)
  - **Preview 3**: Primitive types in patterns, instanceof, and switch) (Project **Amber**)
  - **Prevew 5**: Structured concurrency (Project **Loom**)

Unnamed variables (**Java 22**)

JEP 456

- Permite usar o sublinhado (_) para representar variáveis locais, parâmetros de exceções, parâmetros de lambdas e componentes de padrões que são obrigatórios mas não são usados no código.

```java
public void validar (String s){
    try {
        int i = Integer.parseInt(s);
        System.out.println(i + " é um número");
    } catch (NumberFormatException _) {
        System.out.println(s + " não é um número");
    }
}
```

Objeto da exceção nunca é usada no bloco

```java
button.addActionListener( _ -> {
    System.out.println("Ativado!");
    update();
});
```

Evento não é usado no bloco

```java
public void processar(Contrato c, Writer w) {
    switch (c) {
        case Recusado rec    -> w.println("Motivo: " + rec.recusado());
        case Finalizado _    -> w.println("Finalizado");
        case Iniciado _      -> w.println("Iniciado");
    }
}
```

Variáveis de padrão não são usadas

Module import declarations

JEP 511

- Permite importar todos os pacotes exportados por um módulo.
- Por exemplo:

```
import module java.compiler;
```

Equivalente a escrever:

```
import javax.annotation.processing.*;
import javax.lang.model.*;
import javax.lang.model.element.*;
import javax.lang.model.type.*;
import javax.lang.model.util.*;
import javax.tools.*;
```

Flexible constructor bodies

JEP 513

- Passa a permitir código antes de **`this()`** e **`super()`** (com restrições).
- Pode-se inicializar membros de instância, chamar métodos estáticos (mas não pode ler variáveis ou chamar métodos de instância)
- Útil para fazer validação ou correção de valores antes de inicializar
- Por exemplo:

```
public class DBConnection extends Conection {
    public DBConection(String url) {
        String finalUrl = url;
        if (!url.startsWith("jdbc:")) {
            finalUrl = "jdbc:" + url;
        }
        super(finalUrl);
        this.initializePool();
    }
}
```

Antes era necessário repassar chamada a método estático local ao construtor, que era chamado na superclasse.

Corrigindo a URL antes de construir o objeto

Scoped values

JEP 506

Virtual threads (Java 21/Loom) são threads gerenciados pela JVM (leves, super escaláveis, fáceis de usar e debugar e compatíveis com Threads da CPU). Aprenda a usar!

- Uma alternativa mais eficiente e segura a **`ThreadLocal`** e compatível com **virtual threads**!

Muitos virtual threads + ThreadLocal = baixa performance

Mutável

```
class Antes {
    private static final ThreadLocal<User>
        CURRENT_USER = new ThreadLocal<>();

    public void handleRequest(User user) {
        try {
            CURRENT_USER.set(user);
            processRequest();
        } finally {
            CURRENT_USER.remove();
        }
    }
}
```

Se você esquecer, vaza memória com virtual threads!

- Imutável e eficiente com virtual threads

```
class Depois {
    private static final ScopedValue<User>
        CURRENT_USER = ScopedValue.newInstance();

    public void handleRequest(User user) {
        ScopedValue.where(CURRENT_USER, user)
                   .run(() -> processRequest());
    }
}
```

Fechamento automático (não precisa de try-finally)

Compact source files and instance main methods

JEP 512

- Método **`main()`** de instância + pacote **`java.lang.IO`** + unnamed classes
- Muito mais fácil escrever os primeiros programas em Java!

HelloWorld.java

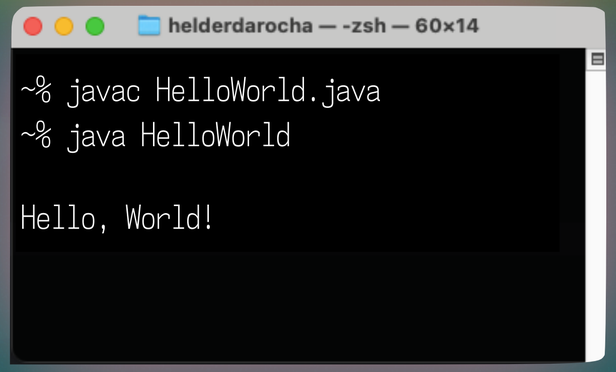
```java
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

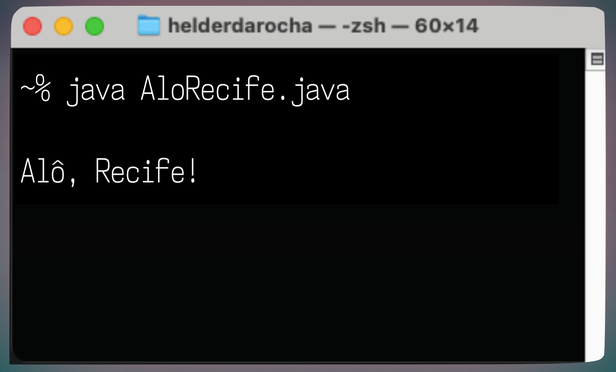
Antes

Depois

AloRecife.java

```java
void main() {
   IO.println("Alô, Recife!");
}
```





Entrada de dados em linha de comando

Finalmente (quase) duas linhas

Antes do **Java 5**

```java
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class HelloWorldInput {
    public static void main(String[] args) {
        BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
        System.out.print("Digite uma mensagem: ");
        String message = null;
        try {
            message = reader.readLine();
        } catch (IOException e) {
            throw new RuntimeException(e);
        } finally {
            try {
                reader.close();
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
        }
        System.out.println(message);
    }
}
```

Antes do **Java 25**

```java
import java.util.Scanner;

public class HelloWorldInput {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Digite uma mensagem: ");
        String message = scanner.nextLine();
        System.out.println(message);
        scanner.close();
    }
}
```

```java
void main() {
    String msg = IO.readln("Digite uma mensagem: ");
    IO.println(msg);
}
```

Java 25

```
helderdarocha — -zsh — 60×14
~% java HelloWorldInput.java

Digite uma mensagem: Alô Recife!
Alô Recife!
```

Hmmm...

Preview 3: pattern matching com tipos primitivos

JEP 507

`java --enable-preview`

- Suporte a **pattern matching** com tipos primitivos em todos os contextos
- `instanceof` e `switch` agora aceitam todos os tipos primitivos

```
Object obj = -33.5;

if (obj instanceof int n) {
    System.out.println("Inteiro: " + n);
}

switch (obj) {
    case int n when n >= 0 -> System.out.println("Inteiro não negativo: " + i);
    case int n when n < 0 -> System.out.println("Inteiro negativo: " + i);
    case double d -> System.out.println("Ponto-flutuante: " + d);
    default -> System.out.println("Outra coisa");
}
```

Em `instanceof`

Em `switch`

Preview 5: concorrência estruturada

JEP 505

`java --enable-preview`

- **Simplifica a programação concorrente**: remove do programador a responsabilidade de ter que lidar com falhas em threads separados em tarefas relacionadas

- Exemplo (da página do JEP 505): método para achar um usuário e um pedido. Este recurso evita que elas falhem ou tenham sucesso independentemente

```
Response handle() throws InterruptedException {
    try (var scope = StructuredTaskScope.open()) {

        Subtask<String> user = scope.fork(() -> findUser());
        Subtask<Integer> order = scope.fork(() -> fetchOrder());

        scope.join();

        return new Response(user.get(), order.get());
    }
}
```

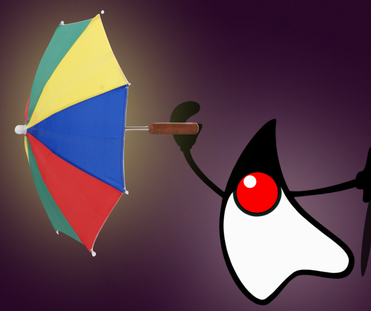
Cria o escopo para tarefa estruturada

Executa subtarefas (threads separados)

Junta as subtarefas, propagando suas exceções

Devolve resultado combinado das duas tarefas

Fonte do exemplo: https://openjdk.org/jeps/505



Conclusões

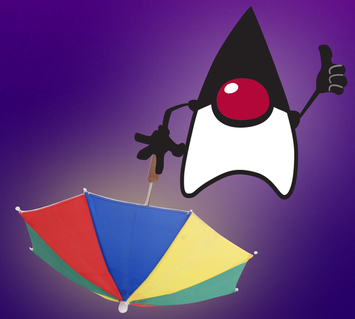
O novo Java é uma linguagem de propósito geral mais **moderna**, mais **fácil de usar**, mais **eficiente** e **multiparadigma.**

Esse é o objetivo dos grandes projetos do **OpenJDK**, quase finalizados.

O foco desta apresentação foram as novidades para programadores.Há vários outros novos recursos no Java 25.

E então, você acha que

ficou mais fácil programar em Java 25?



www.linkedin.com/in/helderdarocha/
www.argonavis.com.br
helder@argonavis.com.br