

## Capítulo 1 - A história e evolução do Java

A inovação e o desenvolvimento da linguagem de computador ocorrem por duas razões:

- Para se adaptar a ambientes e usos em mudança.
- Implementar refinamentos e melhorias na arte de programação.

### Linhagem de Java

Java é parente de C++ que é descendente de C. Herda de C: sintaxe e de C++ recursos orientados a objetos. A criação de Java está ligada à evolução histórica de linguagens que buscavam superar limitações de antecessoras.

### O nascimento da programação moderna: C

A linguagem C foi resultado direto da necessidade de uma linguagem estruturada, eficiente e de alto nível que substituisse o código assembly no criação de programas de sistemas. Seu desenvolvimento envolveu concessões como:

- Facilidade de uso × potência
- Segurança × eficiência
- Rigidão × extensibilidade.

Antes do C, linguagens como Fortran, Basic, Cobol e Pascal possuíam limitações: ou eram eficientes mas pouco estruturadas, ou vice-versa.

O desenvolvimento de C, ocorreu na década de 1970, inventado e implementado por Dennis Ritchie nos laboratórios Bell para o UNIX.

C é considerado por muitos como o marco inicial da era moderna da linguagem de computador pelos conciliou atributos conflitantes,

tornando-se uma linguagem poderosa, eficiente e estruturada, relativamente fácil de aprender. Projetada para programadores, rapidamente conquistou ampla aceitação.

### C++: o próximo passo

No final dos anos 1970 e início de 80, a crescente complexidade dos programas exigiu novas soluções. O C++ surgiu para atender a essa necessidade, permitindo melhor gerenciamento dessa complexidade, principalmente por meio da programação orientada a objetos (POO), que organiza programas complexos em estruturas classes.

A programação estruturada, surgida no decâo de 1960, já havia melhorado a organização do código. A POO, levou isso além, tornando grandes projetos mais fáceis de desenvolver e manter. Inicialmente chamado C com classes, o C++ incorporou recursos orientados a objetos em 1983, mantendo todos os atributos de C e adicionando ferramentas para construção modular e reutilização de código.

### - A criação de Java

Java foi criado por James Gosling e seu equipe no Sun Microsystems em 1995, inicialmente chamado de "Oak", mas renomeado para "Java".

O impeto original não era a internet, mas sim criar uma linguagem independente de plataforma para softwares a ser embutido em dispositivos eletrônicos de consumo.

No época, o problema de C e C++ é que foram projetados para serem compilados para um alvo específico, sendo esse o desafio para criar Java. Gosling e outros começaram a trabalhar em completo protótipo e independente de plataforma. Com ascensão da Web, Java ganhou destaque por suas capacidades em rede em diversos CPUs. Java adotou non suporta integração multithreading e fornecia biblioteca simplificada.

## A conexão C#

C# criado pela Microsoft para oferecer suporte ao .NET Framework, foi influenciado pelo Java. Ambos compartilham a mesma estrutura geral, suportam programação distribuída e utilizam o mesmo modelo de objetos.

### - Como o Java impactou a Internet

Além de simplificar a programação web em geral, o Java inovou com um novo tipo de programa em rede chamado applet, e também abordou questões mais complexas associadas à internet: portabilidade e segurança.

### Applet Java

É um tipo especial de programa Java projetado para ser executado todo pelo internet e executado automaticamente em navegadores compatíveis. Serve para processar dados, fornecer funções simples e permitir que certas funcionalidades fiquem armazenadas no servidor para o cliente.

Embora seja provável que ainda existam applets em uso hoje, a partir do JDK 9 foram descontinuados e removidos no JDK 11.

### Segurança

Programas dinâmicos em rede apresentam riscos, como vírus ou códigos maliciosos que podem aceder recursos de sistema. O Java resolveu isso usando os aplicativos em um ambiente seguro, impedindo acesso não autorizado a outras partes do computador.

## Portabilidade

A internet conecta diversos sistemas e dispositivos diferentes. Java solucionou o problema de compatibilidade permitindo que o mesmo código fosse executado em múltiplas plataformas, CPUs, sistemas operacionais e navegadores sem precisar de versões específicas para cada um. O mecanismo que garante segurança também contribui para essa portabilidade.

### - A magia do Java: o bytecode

O bytecode é a chave para que o Java alcance portabilidade e segurança. Em vez de gerar código executável diretamente, o compilador Java produz um conjunto de instruções padronizadas executadas pelo Maquinário Virtual Java (JVM), parte da JRE. A JVM é o projeto do Java interpretar o bytecode, permitindo que o mesmo programa funcione em diferentes plataformas, bastando que cada uma tenha seu próprio JVM.

Esse modelo elimina a necessidade de criar versões específicas do programa para cada tipo de hardware ou sistema operacional.

A execução dentro do JVM também garante segurança, pois o programa rodeia em um ambiente controlado (sandbox) que restringe o acesso a recursos do sistema, prevenindo danos e reforçando a proteção por meio de restrições adicionais. Embora a interpretação seja, em geral, mais lento que o código nativo, o bytecode do Java é altamente otimizado e conta com o uso de compiladores Just-in-time (JIT), como o HotSpot, que convertem partes do bytecode em código nativo durante a execução, aumentando significativamente o desempenho.

Outro ponto: háve experimentos com compilação ahead-of-time, que transforma o bytecode em código nativo antes da execução pelo JVM. Embora útil em casos específicos, essa abordagem não substitui o método tradicional com JIT, sendo considerado uma solução especializada.

## - Indo além dos applets

Os applets eram uma parte crucial do programação Java, pois eles dependem de um plug-in de navegador no qual vivem dominando. dessa forma, sem suporte ao navegador, os applets não saem mais. Por isso, a partir do JDK 9, a eliminação gradual foi iniciada e concluída com o lançamento do JDK 11. A partir do JDK 17, toda a API do applet foi descontinuada para remoção.

Após a criação do Java, alternaram para o Java Web Start, que permite o download dinâmico de um app a partir de uma página web. Era útil para apps maiores que não eram adequados para applets e também era executado de forma independente, fora do navegador.

Nosso parecendo num app normal, requer um JRE independente com suporte ao Web Start estabelecendo o sistema host. A partir do SDK 11 ele foi removido. Depois foi criado outros mecanismos para implementar o Java, como: jlink (SDK 9) e o jpackage (SDK 16).

## Um cronograma de lançamento mais rápido.

No passado, os principais lançamentos Java eram normalmente separados por dois ou mais anos. Após o lançamento do JDK 9, o intervalo foi reduzido de apenas 6 meses entre as atualizações.

O lançamento semestral, chamado de lançamento de recurso, inclui os recursos prontos no momento e permite que o Java acompanhe rapidamente as mudanças no ambiente de programação, beneficiando o desenvolvimento.

A cada três anos é lançada uma versão de suporte de longo prazo (LTS), que permanece viável por um período superior a seis meses. A primeira versão LTS foi o JDK 11, seguido pelo 17. A estabilidade dessas versões, serve como base de estabilidade por vários anos.

Os lançamentos de recursos estão programados para Março e Setembro. JDK 19 foi lançado em Março de 2018, JDK 11 setembro 2018 e assim por diante.

## Servlets: Java no lado do servidor

Após o lançamento unificado do Java, percebe-se que ele pode ser usado no lado do servidor. Surgiu o servlet, um pequeno programa executado no servidor. Eles são usados para criar conteúdo dinamicamente e enviado ao cliente. Por exemplo, numa loja online pode usar o servlet para consultar o preço de um item em um banco de dados e gerar uma página web com as informações.

Embora mecanismos como o CGI já permitissem conteúdo dinâmico, o servlet oferece vantagens como maior desempenho. Por serem compiladores em bytecode e executado pelo JVM, os servlets são altamente portáteis e podem ser usados em diversos ambientes de servidor, geralmente em containers de servlet.

### As palavras do modus em Java.

Tangões de linguagem: Simples, seguro, portável, orientado à objetos, robusto, multithread, arquitetura neutra, interpretado, alto desempenho, distribuído e dinâmico. Além de portabilidade e segurança.

#### Simples

Java foi criado para ser fácil de aprender e usar. Quem já conhece programação orientada a objetos, C ou C++, aprende Java sem esforço.

#### Orientado a objetos.

Java não foi projetado para ter código-fonte compatível com nenhum outro linguagem, permitindo a liberdade de projeto do zero para ter uma abordagem clara e prática no POC, equilibrando simplicidade e desempenho, mantendo tipos primitivos Java dos objetos.

## robusto

Tarvo foi projetado para criar programas confiáveis, detectando erros logo no inicio do desenvolvimento. Ele verifica o código tanto em tempo de compilação quanto em tempo de execução, dificultando o surgimento de bugs.

A linguagem evita dois problemas comuns: erros de gerenciamento de memória e falhas em condições excepcionais. Com colete de lixo e tratamento pressível de erros, Tarvo reduz a complexidade e aumenta a segurança dos programas.

## Multithread

Tarvo permite criar programas que executam várias tarefas ao mesmo tempo. Isso possibilita o desenvolvimento de sistemas interativos mais eficientes e com melhor desempenho, sem prejudicar a execução de programas.

## Arquitetura neutra

Tarvo foi projetado para ser portátil e durável, com o objetivo de "escrever uma vez, executar em qualquer lugar, a qualquer hora, para sempre". Garantindo o funcionamento em diferentes máquinas e sistemas, mesmo após atualizações.

## Interpretado e de alto desempenho

O código Tarvo é compilado em bytecode, que pode ser executado em qualquer sistema com a JVM. O bytecode foi projetado para ser facilmente traduzido em código nativo, permitindo alto desempenho com o uso de compiladores just-in-time.

## Distribuído

Foi projetado para o ambiente distribuído da internet, pous lid com protocolos TCP/IP. Java também suporta Invocação de Método Remoto (RMI) que permite invocar métodos em uma rede.

## Dinâmico

Java armazena informações detalhadas de tipo em tempo de execução, permitindo verificar e acessar objetos de forma segura e rápida. Isso possibilita vinculação dinâmica de código e a atualização de bytecode enquanto o sistema estiver em funcionamento, aumentando a robustez e flexibilidade da linguagem.

### - A Evolução do Java

Ao contrário do maior dos outros sistemas de software, que geralmente se acomodam em um padrão de pequenos melhorias incrementais, o Java continuou a evoluir em um ritmo explosivo.

O Java surgiu com a versão 1.0, revolucionando a programação para a internet. Logo criou 1.1 e veio com novos elementos de biblioteca e remoção de recursos obsoletos da versão antiga.

A criação de Java 2 (J2SE 1.2), introduziu recursos como Swing, Collections Framework, melhorias no JVM e alterações na classe Thread, na qual os métodos suspend(), resume(), e stop() foram descontinuados.

O J2SE 1.3 e 1.4 continuaram essa evolução com aprimoramento na biblioteca, novo palavrão-chave assert, exceções encadeadas, e um subsistema de E/S baseado em corrd. A versão 1.4 manteve quase 100% de compatibilidade com codiça.

O Java J2SE 5 foi revolucionário. Entre seus recursos estão genericas, laço for-each, varargs, importação estática, auto boxing/unboxing, enumerações, anotações e utilitários de simultaneidade. O Java 5 e 6 consolidaram essas melhorias, trazendo novos pacotes e otimizações na execução.

Com o Java SE 7, vieram mudanças de sintaxe pelo Projeto Lasso, como o operador `lambda` para inferência de tipos, multi-catch, `try-with-resources`, `String` em `switch` e literais binários também trouxe melhorias no NIO e a introdução do Fork/Join Framework para programação paralela. No Java SE 8, as expressões `lambda` transformaram a linguagem, acompanhadas da API de Streams, métodos padrão em interfaces, novo API de `date/time` e suporte a processamento paralelo otimizado.

O Java SE 9 trouxe o sistema de módulos, o ferramente `glink` para criar runtimes personalizados no `JSshell` para experimentação interativa. Applets foram o Java SE 10 adicionou a inferência de tipo de variáveis locais com `var`, e o Java SE 11, versão LTS, expandiu o uso de `var` para expressões `lambda`, adicionou a `HTTP Client API` e removeu definitivamente applets e Java Web Start.

Entre o JDK 12 e 16, surgiram melhorias no `switch` (incluindo expressões), blocos de texto, pattern matching para `instanceof` e a introdução de records. No Java 17, LTS, chegou o conceito de classes e interfaces seladas, controlando a herança de forma mais restrita, e a API de applet foi marcada para remoção.

Ao longo do histórico, Java manteve-se em constante evolução, impulsionado pelo Java Community Process e pelo modelo open source que iniciou em 2006, consolidando-se como uma linguagem versátil, robusta e adaptado às demandas modernas de programação.