



①

②

## The history and evolution of Java

### Java's lineage :

- ⇒ Java é um "descendente" de C e C++
  - ⇒ Onde, a sua sintaxe é derivada do C, e os recursos de orientação a objeto do C++
- ⇒ Suas criações decorrem de um processo de adaptação e refinamento de linguagens de programação que vem ocorrendo a duas décadas

### The birth of modern programming : C

- ⇒ Linguagem com MUITO impacto mundial, mudou a forma de programar
  - ⇒ Veio do necessidade de uma linguagem estruturada, eficiente e high-level
  - ⇒ Veio para substituir o Assembly !!!

- ⇒ Trocas de linguagem de programação ocorrem de vez em quando
  - Facilidade de uso versus potência
  - Segurança versus eficiência
  - Portabilidade versus extensibilidade

(2)

→ Antes do C as linguagens, normalmente, tinham um conjunto de características → Muitas linguagens tinham que ser utilizadas em questões disso → C "quebrou" essa forma de programar

↳ Outro problema era que essas linguagens não eram fundamentadas em princípios estruturais (- organizadas), dependiam de instruções GOTO para fazer o controle dos programas

↳ Por isso, muitas dessas linguagens mais antigas acabavam gerando "Spaghetti code" (códigos mal organizados → e muito difíceis de serem lidos)

→ Inventado e implementado por Dennis Ritchie nos anos 70

↳ BCPL → B → C

→ C marcou o começo da era moderna das linguagens de programação (era lisa, rendeu muitos problemas de época e era relativamente fácil de aprender)

↳ Foi a primeira linguagem desenvolvida por, de fato, "programador el", antes as linguagens eram feitas em exercícios acadêmicos ou por comitês burocráticos

### C++: The next step

→ Surgiu devido ao aumento da complexidade dos programas da época

↳ Os métodos de programação estruturada estavam chegando no seu limite

↳ Dessa limitação surgiu a Orientação a Objetos !!

**spiral**

(3)

→ A programação orientada a objeto (POO) é uma metodologia de programação que ajuda a organizar problemas complexos através do uso de herança, encapsulamento e polimorfismo

→ Programação estruturada tem dificuldade com problemas/projetos muito grandes

↳ C++ aplica POO para resolver essa questão

→ Foi inventado por Bjarne Stroustrup em 1979, tem todos os recursos, atributos e benefícios do C + POO!

### The stage is set for Java

→ C++ dominou durante o final dos anos 80 e inicio dos anos 90

↳ Uma nova linguagem (interpretável) estava por vir (Java)

### The creation of Java

→ Foi criado por James Gosling, Patrick Naughton, Chris Warth, Ed Frank e Mike Sheridan em 1991

→ Foi originalmente chamada de Oak, depois a mudança para Java ocorreu em 1995

→ Java originalmente não foi criado para a Internet, mas sim para ser uma linguagem de plataforma independente (conquistando mundo) que fosse como função criar software para dispositivos eletrônicos, como micro-ondas e controles remotos

(4)

→ Um problema com C e C++, elas eram feitas para rodar em um tipo de CPU específico

↳ Ainda é possível rodar um programa em C++ em qualquer tipo de CPU, mas para isso é necessário um compilador específico para aquela CPU (e compiladores costumam diminuir!!!)

↳ Java resolve esse problema, pois seu código pode rodar em uma ampla gama de CPU's e em diversos ambientes (programas portáteis)

→ A World Wide Web (WWW) estava emergindo na mesma época e influenciou muito na criação do Java (ela, a Web, demandava programar portáteis)

→ Java é a solução para o quanto da portabilidade dos programas

→ Java não veio para substituir o C++, mas sim para resolver outros problemas

↳ Foi a "demanda perfeita" para a época

→ Java foi para a programação Web e que o C foi para a programação de sistemas, uma revolução

### The C# connection:

→ Foi criado pela Microsoft (inspirado no Java) para suportar o framework .NET

→ Ambos tem uma sintaxe parecida, suportam programação distribuída e usam o mesmo modelo de objeto



(5)

## How Java impacted The Internet

→ Java simplificou a programação web no geral, também inovou com um novo tipo de programa em rede chamada Applet e em questões de portabilidade e segurança.

### Java Applets:

→ É um programa especial feito em Java que é transmitido pela internet e é automaticamente executado em um Web-browser compatível com Java.

↳ Se um usuário clica em um link que contém um applet, ele é automaticamente instalado e vai rodar no browser.

→ Em resumo, Applets permitem funcionalidades remotas movidas do servidor para o cliente, como exibir dados do servidor, receber input, prover funções simples etc.

→ A criação dos Applets foi importante para expandir a quantidade de objetos que podiam se mover praticamente de maneira livre no cyber-espaco.

↳ Existem duas categorias de objetos que circulam entre servidor e cliente: Informações passiva e dinâmica.

↳ Ex: Ler um e-mail: Visualizando informação passiva.

↳ Em contraste, os Applets são programas dinâmicos auto-executáveis.

→ Suporte para Applets foi removido totalmente no JDK 11.

⑥

Security

- Java é eficaz nesse quanto por qualquer download de arquivo, a aplicação é carregada no ambiente de execução do próprio Java, fazendo com que caso haja um virus no seu download, ele não tenha acesso a nenhuma informação fora do ambiente de execução.

Portabilidade

- Uma das grandes vantagens da Java, é a sua portabilidade para rodar seu programa em uma ampla gama de ambientes e CPU's

↳ A mesma aplicação deve funcionar em Toda os computadores

Java's magic: The bytecode

- A chave que ajudou a resolver os problemas de segurança e portabilidade: o output de um compilador Java não é código executável, mas sim, o bytecode!!!

↳ Bytecode é um conjunto de instruções que é executado pela Java Virtual Machine (JVM) que faz parte do Java Runtime Environment (JRE)

↳ Originalmente, em sua origem a JVM foi projetada como interpretador do Java

- A vantagem de traduzir Java para bytecode é no fato de que fica muito mais fácil rodar o programa em uma variedade de ambientes, uma vez que apenas a JVM precisa ser implementada para cada plataforma.

**spiral**

⑦

↳ Há um Dado um JRE que executa em tal sistema, qualquer programa em Java pode rodar nele

→ Todo JVM entende em o mesmo bytecode em Java bytecode.

↳ JVM é a forma mais fácil de criar programas verdadeiramente portáteis

→ Se um programa em Java for compilado para código nativo, então haveriam diferentes versões do mesmo programa para cada CPU conectada à Internet

→ JVM também ajuda na segurança, uma vez que ele está no controle, ele ~~monitors~~ gerencia a execução do programa

→ O código compilado para uma forma intermediária roda lentamente mas devagar do que um código diretamente compilado para código executável

→ Technologic HotSpot implementa um Just-In-Time (JIT) compilador para bytecode

\* Lembrando, Java foi feito para ser interpretado !!!

→ Um JIT compiler compila partes do bytecode em código executável em tempo real (não compila ele inteiro de uma vez!!!)

↳ Vai compilando quando necessário, o código restante é interpretado

↳ JIT garante performance melhorada

(8)

## Moving beyond Applets

~ Applets ainda existem, porém não considerados obsoletos

~ Outro alternativo para Applets, o Java Web Start

↳ Permite que uma aplicação seja dinamicamente instalada de uma página web

↳ Utilizado para aplicações de larga escala em Java

~ A diferença entre um Applet e o Java Web Start, é que o JWS roda nele mesmo e não dentro do browser, se parece com uma aplicação "normal"

~ JWS foi removido no JDK 7

~ Qual mecanismo é usado para fazer deploy de uma aplicação Java com os Applets e o JWS fora de java?

↳ Jlink

↳ Jpackage

## A faster releases Schedule:

~ Antigamente, novas versões saíam de 2 a 3 anos, hoje em dia, não lançadas de 6 a 9 meses

~ Em um intervalo de 3 anos, não lançadas as versões Long Term Support (LTS)

↳ Enviar versões contínuas novas e recendo update por mais **spiral** que 6 meses

9

Servlets; Java on the server side

→ Java também pode ser usado no server side

↳ Para isso não criarem os Servlets, programas que executam no servidor?

↳ São usados para criar conteúdo gerado dinamicamente que é servido para o cliente

→ Assim como todo programa em Java, ele não compilador para bytecode e executado pela JVM e portanto, não extremamente performático

↳ O mesmo servlet pode ser usado em diversos ambientes

The Java Buzzwords

- Simple
- Secure
- Portable
- Object-oriented
- Robust
- Multithreaded (executa múltiplas tarefas de uma vez)
- Architecture-neutral
- Interpreted
- High performance
- Distributed
- Dynamic

Simple:

→ De fácil aprendizado (principalmente para quem já sabe C, C++)

10

Object-Oriented

- Considerado por ser fácil e pragmático de usar no contexto de orientação a objetos

Reliability

- Para ter ter um foco para Wile, onde há demanda ~~extraordinária~~, o programa deve ser robusto e confiável.
- Java te restringe um "error chain" para te forçar a olhar seus erros mais cedo no decorrer do código.
- Como Java é uma linguagem tipada, ela checa seu código na hora da compilação
  - ↳ Porém, ela checa ele na hora de rodar também !!!
  - ↳ Inve impossibilita a criação de vários bugs em Java
- Java se aloca e desaloca memória para você automaticamente

Architecture-neutral:

- Esse é o objetivo "Write once; Run anywhere, any Time, forever" foi alcançado

Interpreted and high performance:

- Java bytecode permite alta performance e criação de programas cross-platform

spiral

(1)

Distributed:

→ Java foi desenhado para o ambiente distribuído da Internet pois lido com os protocolos TCP/IP

↳ Acessar um recurso utilizando uma URL não difere muito de usar um arquivo para isso

→ Java também suporta o Remote Method Invocation (RMI)

↳ Permite que o programa invogue métodos a partir da rede

Dynamic:

→ Os programas Java carregam conosco quantidades substanciais de Run-Time type information que são usadas para verificar e rever acessos a objetos em tempo de execução

↳ Isso torna possível vincular dinamicamente código de maneira segura e conveniente

The evolution of Java:

→ Java 1.0 para 1.1 = muito diferente

→ Java 2 marcou o que ficou conhecido como "Era moderna do Java"

↳ Suporte para um grande número de novos recursos, também contém algumas depreciações / descontinuamentos

⑫

↳ Algunas depreciações mais famosas foram das métodos suspend(), resume() e stop()

\* J2SE (Java 2 platform standard edition)

↳ O lançamento do J2SE 5 foi revolucionário !!!

- Generics
- Annotations
- Auto boxing and auto-unboxing
- Enumeration
- Enhanced for-each style for loops
- Variable-length arguments (varargs)
- Static import
- Formatted I/O
- Concurrency utilities

↳ O bat de desenvolvimento se chamou JDK 5

→ No lançamento seguinte, o nome passou de J2SE 5 para Java SE 6 (Java platform standard edition)

↳ JDK 6

→ JDK 8 trouxe o recurso da Lambda expression

↳ Adicionaram recursos de programação funcional, ela pode simplificar e reduzir a quantidade de código fonte necessário para criar certas construções, como alguns tipos de classes anônimas.

(13)

→ JDK 9 trouxe o Jshell, ferramenta que suporta programação interativa

→ JDK 17, uma das versões mais recentes!

### A culture of innovation

→ Desde a sua criação, Java esteve no centro de uma cultura de inovação

↳ JVM, bytecode e JCP foram e continuam sendo revolucionários