UPGRADE OU MORTE

INTRODUÇÃO

Uma Nova Esperança para os Desenvolvedores Java

Desde o seu lançamento em 1995, o Java tem sido uma força dominante no universo da programação. Com cada nova versão, a linguagem evolui, trazendo melhorias, otimizações e, por vezes, quebrando paradigmas. Entre as muitas versões, duas se destacam por sua popularidade e impacto: Java 8 e Java 11.

O Java 8, lançado em 2014, foi um divisor de águas, introduzindo recursos revolucionários como Lambdas e Streams API, que mudaram fundamentalmente a forma como escrevemos código em Java. Já o Java 11, uma versão de suporte de longo prazo (LTS) lançada em 2018, consolidou muitas das novidades que surgiram nas versões intermediárias (Java 9 e 10) e trouxe consigo otimizações e recursos focados na produtividade e na modularidade.

Neste ebook, embarcaremos em uma jornada para desvendar as principais diferenças entre essas duas gigantes do Java. Se você é um desenvolvedor que ainda está no Java 8 e pensa em migrar, ou simplesmente quer entender o que a versão 11 tem a oferecer, este guia é para você. Prepare-se para a batalha dos bytes!

O Legado do Java 8
O Despertar da Funcionalidade

O Java 8 marcou uma era dourada para a linguagem, trazendo consigo recursos que a tornaram mais concisa, expressiva e moderna.

Lambdas: A Força Simplificada

Antes do Java 8, a programação funcional em Java era um tanto quanto verbosa, exigindo classes anônimas para implementar interfaces funcionais. Com as expressões lambda, a sintaxe se tornou muito mais limpa e legível.

Java

```java
// Antes do Java 8
new Thread(new Runnable() {
    @Override
    public void run() {
        System.out.println("Olá do Java 8 (antigo)!");
    }
}).start();

// Com Java 8 Lambdas
new Thread(() -> System.out.println("Olá do Java 8 (com Lambdas)!"))
    .start();
```

Streams API: O Fluxo de Dados Otimizado

A Streams API revolucionou o processamento de coleções em Java. Ela permite operações declarativas e funcionais em sequências de elementos, facilitando a filtragem, mapeamento e redução de dados.

Java

```java
List<String> nomes = Arrays.asList("Luke", "Leia", "Han", "Chewie");

// Filtrando e imprimindo nomes com "e"
nomes.stream()
     .filter(nome -> nome.contains("e"))
     .forEach(System.out::println);
```

Outras Novidades Essenciais do Java 8

Default Methods em Interfaces: Permitiu adicionar novos métodos a interfaces sem quebrar implementações existentes, facilitando a evolução das APIs.

Optional: Uma classe para lidar com valores que podem ser nulos, reduzindo a incidência de NullPointerExceptions.

Date and Time API (java.time): Uma API moderna e imutável para trabalhar com datas e horas, corrigindo as deficiências da antiga java.util.Date e java.util.Calendar.

02

A Ascensão do Java 11
A Nova Ordem

O Java 11 é uma versão LTS (Long-Term Support), o que significa que receberá suporte e atualizações por um período prolongado, tornando-o uma escolha popular para ambientes de produção. Ele incorpora e aprimora muitos recursos introduzidos nas versões 9 e 10.

Módulos (Jigsaw): A Arquitetura Modular

Introduzido no Java 9 e consolidado no Java 11, o Sistema de Módulos Java (JPMS), também conhecido como Projeto Jigsaw, é uma das maiores mudanças arquitetônicas na plataforma. Ele permite criar aplicações mais robustas, seguras e com melhor desempenho, encapsulando componentes em módulos bem definidos.

Java

```java
// Exemplo básico de um arquivo module-info.java
module meu.modulo {
    requires java.base; // Módulo fundamental do Java
    exports com.minhaempresa.minhaapp; // Exporta pacotes para outros módulos
}
```

A modularização ajuda a reduzir o tamanho das aplicações (especialmente importante para microsserviços e contêineres) e a evitar o "classpath hell".

Var: Inferência de Tipo Local

Introduzido no Java 10 e disponível no Java 11, a palavra-chave var permite inferência de tipo para variáveis locais. Isso reduz a verbosidade do código, especialmente ao lidar com tipos complexos.

Java

```java
// Antes do Java 10/11
Map<String, List<String>> minhaVariavelComplexa = new HashMap<>();

// Com 'var'
var minhaVariavelComplexa = new HashMap<String, List<String>>();
```

É importante notar que var não transforma Java em uma linguagem dinamicamente tipada; o tipo ainda é inferido em tempo de compilação.

Novos Métodos em String
Poder para as Strings

O Java 11 adicionou vários métodos úteis à classe String, tornando a manipulação de strings mais eficiente:

isBlank(): Retorna true se a string estiver vazia ou contiver apenas espaços em branco.

- lines(): Retorna um Stream de linhas da string.
- repeat(int count): Repete a string o número de vezes especificado.
- strip(), stripLeading(), stripTrailing(): Melhores do que trim()

Java

```
String texto = " Java ";
System.out.println(texto.repeat(3)); // Saída: Java  Java  Java
System.out.println("  \n  ".isBlank()); // Saída: true
```

HTTP Client API (Padrão)
Conexões Mais Fortes

O cliente HTTP assíncrono e síncrono, introduzido como incubador no Java 9 e padronizado no Java 11, oferece uma API moderna e flexível para fazer requisições HTTP.

Java

```java
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;

// ...

HttpClient client = HttpClient.newBuilder().build();
HttpRequest request = HttpRequest.newBuilder()
                                 .uri(URI.create("https://api.github.com/users/goo
                                 .build();

client.sendAsync(request, HttpResponse.BodyHandlers.ofString())
      .thenApply(HttpResponse::body)
      .thenAccept(System.out::println);
```

Single-File Source Code Programs
Execução Simplificada

Java 11 permite executar arquivos de código-fonte de um único arquivo diretamente, sem a necessidade de compilação explícita. Isso é excelente para scripts rápidos e aprendizado.

Bash

```
# Salve o código abaixo como HelloWorld.java
// HelloWorld.java
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Olá, Mundo de Java 11!");
    }
}

# Execute no terminal
java HelloWorld.java
```

03

A Colisão dos Mundos
Migrando do Java 8 para o Java 11

Migrar do Java 8 para o Java 11 pode ser um processo tranquilo, mas requer atenção a alguns pontos chave, especialmente por causa das mudanças no sistema de módulos e na remoção de algumas APIs.

Removendo Módulos Java EE e CORBA

Uma das maiores mudanças é a remoção de módulos Java EE (Java Enterprise Edition) e CORBA do JDK padrão. Se sua aplicação depende de APIs como JAXB, JAX-WS ou CORBA, você precisará adicionar essas dependências explicitamente ao seu projeto.

Solução: Adicione as dependências como JARs ou via gerenciadores de pacotes (Maven/Gradle) se você estiver usando um projeto modular.

Gerenciamento de Dependências

Certifique-se de que todas as bibliotecas e frameworks que seu projeto utiliza sejam compatíveis com Java 11. A maioria das bibliotecas populares já oferece suporte, mas é crucial verificar as versões.

Ferramentas de Build e IDEs

Atualize suas ferramentas de build (Maven, Gradle) e IDEs (IntelliJ IDEA, Eclipse, VS Code) para versões que ofereçam suporte total ao Java 11.

Jdeps
Analisando Dependências

Atualize suas ferramentas de build (Maven, Gradle) e IDEs (IntelliJ IDEA, Eclipse, VS Code) para versões que ofereçam suporte total ao Java 11.

Bash

```
jdeps -s seu-aplicativo.jar
```

Testes Rigorosos

Após a migração, realize um conjunto abrangente de testes para garantir que sua aplicação funcione corretamente na nova versão do Java.

CONCLUSÃO

O Futuro é Modular e Otimizado

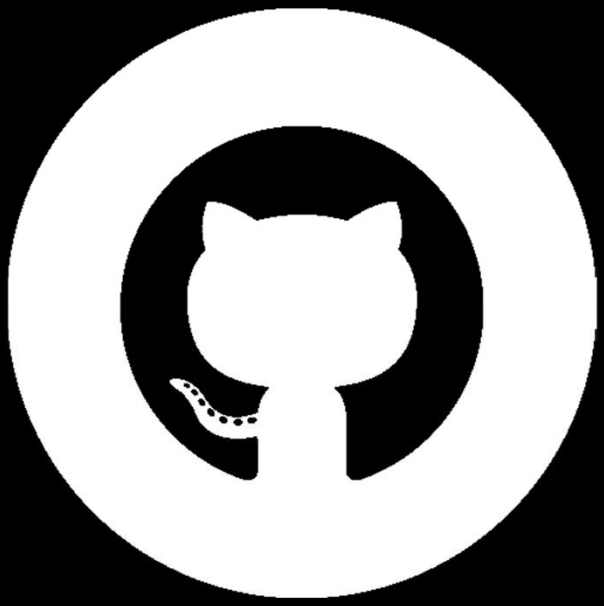
A transição do Java 8 para o Java 11 representa um salto significativo na evolução da plataforma. Enquanto o Java 8 nos presenteou com a concisão das Lambdas e o poder das Streams, o Java 11 consolidou a modularidade, otimizou a performance e trouxe melhorias na produtividade.

Migrar para o Java 11 não é apenas uma questão de acompanhar as novas funcionalidades, mas também de abraçar uma plataforma mais segura, eficiente e preparada para os desafios da computação moderna. Com um ecossistema mais modular e uma performance aprimorada, o Java continua a ser uma ferramenta poderosa nas mãos dos desenvolvedores que buscam construir sistemas robustos e escaláveis.

QUE A FORÇA ESTEJA COM VOCÊ EM SUA JORNADA COM JAVA!

OBRIGADO POR LER ATÉ AQUI

Este ebook foi criado através de um prompt no chatgpt e sua diagramação foi feita por um humano.



HTTPS://GITHUB.COM/MEDEIROS000