Desenvolvimento Avançado em Java

Descrição	Aprenda em 8 horas de curso a executar de forma eficaz e ágil a linguagem Java e ainda conheça quais são as novidades nas versões da tecnologia: Java 10 e Java 11.
Produzido	AndersonFroes - https://andersonfroes.github.io/Portfolio/

`Formação com esse curso:`

1. Inter Java Developer

`Professor:`

João Paulo Oliveira Santos

`Aulas:`

▼ **Matéria**

▼ **Capitulo 1 - Paradigma Funcional no Java**

Objetivos da Aula

- Entender o Paradigma Funcional no Java
- Aprender como utilizar uma lambda e API Lambda do Java
- Entender paradigma da recursividade (Taill Call Optimization e Memorization)

Requisitos Básicos

- Conceitos básicos de Java
- Orientação Objeto
- Java Generics
- Collections: List e Set

Parte 1: Paradigma Funcional no Java

Programação funcional é o processo de construir software através de composição de funções puras, evitando compartilhamento de estados, dados mutáveis e efeitos colaterais. É declarativa ao invés de imperativa, essa é uma definição de Eric Elliott.

- Paradigma Imperativo: É aquele que expressa o código através de comandos ao computador, nele é possível ter controle de estado dos objetos.
- Paradigma Funcional: Damos uma regra, uma declaração de como queremos que o programa se comporte.

Conceitos fundamentais da programação funcional

- Composição de funções: É criar uma nova função através da composição de outras. Por exemplo, vamos criar uma função que vai filtrar um array, filtrando somente os números pares e multiplicando por dois.
- Funções Puras: É chamada de pura quando invocada mais de uma vez produz exatamente o mesmo resultado.
- Imutabilidade: Significa que uma vez uma variável que recebeu um valor, vai possuir esse valor para sempre, ou quando criamos um objeto ele não pode ser modificado.
- Imperativo x Declarativo:

  É muito comum aprender a programar de forma imperativa, onde mandamos alguém fazer algo, Busque o usuário 15 no banco de dados. Valide essas informações do usuário.

  Na programação funcional tentamos programar d forma declarativa, onde declaramos o que desejamos, sem explicitar como será feito. Qual o usuário 15? Quais os erros dessas informações?

Parte 2: Lambda no Java

Os lambdas obedecem o conceito do paradigma funcional, com eles podemos facilitar legibilidade do nosso código, além disso com a nova API Funcional do Java podemos ter uma alta produtividade para lidar com objetos.

Primeiramente, devemos entender o que são interfaces funcionais.

Interfaces funcionais: São interfaces que possuem apenas um método abstrato, Exemplo: *public interface Funcao { String gerar (String valor); }*

Geralmente as interfaces funcionais possuem uma anotação em nível de classe (@FunctionalInterface), para forçar o compilador a apontar um erro se a interface não estiver do acordo com as regras de uma interface funcional. Esta anotação não obrigatória, pois o compilador consegue reconhecer uma interface em tempo de compilação.

Antes do Java 8, se quiséssemos implementar um comportamento especifico para uma única classe deveríamos utilizar uma classe anônima para implementar este comportamento.

Agora que sabemos como se define uma interface funcional podemos, aprender como se define uma lambda.

Estrutura de uma lambda: InterfaceFuncional nomeVariavel = parametro → logica

Parte 3: Recursividade

Na recursividade, uma função chama a si mesma repetidamente, até atingir uma condição de parada. No caso de Java, um método chama a si mesmo, passando para si objetos primitivos. Cada chamada gera uma nova entrada na pilha de execução, e alguns dados podem ser disponibilizados em um escopo global ou local, através de parâmetros em um escopo global ou local.

Recursividade tem um papel importante em programação funcional, facilitando que evitemos estados mutáveis e matenhamos nosso programa mais declarativo, e menos imperativo.

Tail Call (Recursividade em cauda):

Recursão em cauda é uma recursão onde não há nenhuma linha de código após a chamada do próprio método e, sendo assim, não há nenhum tipo de processamento a ser feito após a chamada recursiva.

Observação: A JVM não suporta a recursão em cauda, ele lança um estouro de pilha (StackOverFlow).

Memorization

É uma técnica de otimização que consiste no cache do resultado de uma função, baseado nos parâmetros de entrada. Com isso, nas seguintes execuções conseguimos ter uma resposta mais rápida.

▼ Capitulo 2 - Interfaces Funcionais

Objetivos da Aula:

- Entender o Paradigma Funcional no Java.
- Aprender como utilizar uma lambda e API lambda do Java 8.

Requisitos Básicos:

- Conceitos básicos de Java.
- Orientação Objeto.

Parte 1: Funções de alta ordem

É uma função que retorna uma função ou que recebe uma função como paramêtro.

▼ Capitulo 3 - Processamento Assíncrono e Paralelo

Objetivos da Aula:

- Entender o conceito de síncrono e assíncrono.
- Entender o funcionamento da API de Threads do Java.
- Entender o funcionamento da função PararelStreams no Java.

Requisitos básicos:

- Assistir as aulas sobre Paradigma Funcional deste curso.

Parte 1: Threads

É um pequeno programa que trabalha como um subsistema, sendo uma forma de um processo se autodividir em duas ou mais tarefas. Essas tarefas múltiplas podem ser executadas simultaneamente para rodar mais rápido do que um programa em um único bloco ou praticamente juntas.

Processamento síncrono

- São todos os processamentos que ocorrem em sequência (sincronia). Os processos são executados em fila. É preciso que um processo termine para que outro processo seja executado.

  Exemplo: Imagine você lavando louça e de repente você se lembra que tem que fazer uma ligação. A ligação só poderá ser realizada quando o processo lavar louça for finalizado.

Processamento assíncrono

- Quando dois ou mais processos são realizados ao mesmo tempo, é dado o nome de processamento assíncrono. Os processos são realizados simultaneamente sem que um processo necessite que outro termine para ser executado.

  Exemplo: Lavar louça e falar ao telefone ao mesmo tempo. Se você não sabe como fazer isso, prenda o telefone entre a cabeça e o ombro e tenha as mãos livres para lavar louça.

▼ **Capitulo 4 - Por dentro da Modularização do Java**

Objetivos da Aula:

1 - Entender o que é o Jigsaw

Parte 1: Jigsaw

Há muito tempo se diz sobre modularizar a plataforma Java. É um plano que começou desde antes do Java 7, foi possibilidade no Java 8 e por fim, para permitir mais tempo de desenvolvimento, revisão e testes, foi movido para o Java 9.

O projeto Jigsaw, como foi chamado, é composto por uma série de JEPs. Algumas delas inclusive já disponíveis no Java 8, como os conhecidos Compact Profiles. A ideia por trás do projeto não é só criar um sistema de módulos, que poderemos usar em nossos projetos, mas também aplicá-lo em toda a plataforma e JDK em busca de melhor organização e desempenho.

Por padrão, todo sistema modular já vem com o módulo java.base , java.util e demais pacotes muitas vezes essenciais para a esmagadora maioria dos projetos.

▼ **Capitulo 5 - Novidades do Java 10**

Aplicando os novos releases da linguagem na prática I:

- Link do Docker = https://www.docker.com/get-started (escolha a versão equivalente ao seu sistema operacional).
- Link do video ensinando a fazer a instalação: https://www.youtube.com/watch?v=OweZAewo54A

Aplicando os novos releases da linguagem na prática II:

Algumas anotações sober o var:

Consegue

- variáveis local inicializadas
- variável suporte do enhaced for
- variável suporte do for iterativo
- variável try with resource

Não consegue

- var não pode ser utilizado em nível de classe
- var não pode ser utilizado como parâmetro
- var não pode ser utilizada em variáveis locais não inicializadas

Aplicando os novos releases da linguagem na prática III:

Comandos utilizados no docker/terminal

- docker = Verificar instalação
- docker container run -it -m512M --entrypoint bash openjdk:7-jdk (terminal da maquina JDK 7)
- java -XX:+PrintflagsFinal -version (Parâmetros de configuração do java)
- java -XX:+PrintflagsFinal -version | grep MaxHeapSize
- exit - sair da imagem
- docker run -it -m 512M --entrypoint bash openjdk:10-jdk (terminal da maquina JDK 10)
- java -XX:+PrintflagsFinal -version | grep MaxHeapSize
- docker container run -it --cpus 2 openjdk:10-jdk (entrar na pasta jshell)
- Runtime.getRuntime().availabeProcessors() - (Mostrar cpus disponiveis)

▼ Capitulo 6 - Novidades do Java 11

Parte 1: Http Client API

Um dos recursos que foram incluídos na próxima versão do JDK 11 é a API do cliente HTTP padronizada que visa substituir a classe HttpUrlConnection legada, que está presente no JDK desde os primeiros anos do Java. O problema com essa API antiga é descrito na proposta de aprimoramento, principalmente porque agora ela é considerada antiga e difícil de usar.

A nova API suporta HTTP / 1.1 e HTTP / 2. A versão mais recente do protocolo HTTP foi projetada para melhorar o desempenho geral do envio de solicitações por um cliente e do recebimento de respostas do servidor. Isso é conseguido através da introdução de várias alterações, como multiplexação de fluxo, compactação de cabeçalho e Push Promise. Além disso, o novo cliente HTTP também suporta nativamente WebSockets.