Collections Java

- Collection é um objeto que agrupa múltiplos elementos (variáveis primitivas ou objetos) dentro de uma única unidade
- Serve para armazenar e processar conjuntos de dados de forma eficiente.

Composição

- Interfaces: É um contrato que quando assumido por uma classe deve ser implementado.
- Implementações ou Classes: são as materializações, a codificação das interfaces.
- Algoritmos: É uma sequência lógica, finita e definida de instruções que devem ser seguidas para resolver um problema.

Java.util.List

- Elementos duplicados e garante ordem de inserção
- ArrayList e Vector implementam a interface List
- LinkedList implementa tanto a interface List quando a interface Queue

Interface Comparable

Utilizado para comparar objetos na lista. Para ordenar de acordo com a ordem natural as classes de uma list, é preciso incrementar a interface Comparable, que obriga a sobrescrever o método compareTo.

Interface Comparator

Para comparar com outros tipos de dados das classes de uma lista, foi preciso criar uma classe, que implementa do Comparator. Dessa maneira é possível utilizar o sort() tanto do Collections quanto da própria lista.

Java.util.Set

- Não permite elementos duplicados
- Não possui índice

Com o HashSet os elementos não ficam organizados da mesma forma que foram criados, mas ficam organizados de maneira aleatória.

Com o LinkedHashSet obtemos os elementos organizados da forma que foram criados.

Com o TreeSet obtemos os elementos numéricos organizados em forma crescente. Por causa do Comparable envolvido no argumento, ele consegue organizar de forma crescente.

Java.util.Map

Não faz parte da hierarquia do Collections.

Utiliza-se put() ao invés de add().

Não tem como adicionar, mas tem como alterar o valor de uma chave já existente.

Como no Set, não há método que possa mostrar o índice.

Para exibir as chaves de um Map, são guardados em um Set do tipo String. Por outro lado, para exibir os valores das chaves de um Map, são guardados em uma Collection.

Stream API

Classe Anônima: É uma classe que não recebeu o nome e é declarado e instanceado em uma única instrução. Esta deve ser utilizado quando precisar criar uma classe que será instanceado apenas uma vez.

Functional Interface: Qualquer interface com um SAM (Single Abstract Method) é uma interface funcional e sua implementação pode ser tratada como expressões lambda.

Pode ter o FunctionalInterface ou não, ambos sendo funcionais.

Quais são as interfaces funcionais que vamos conhecer?

- Comparator

- Consumer

- Function

- Predicate

Lambda: É uma função sem declaração, sem nome, sem tipo de retorno e sem modificador de acesso. A ideia é que o método seja declarado no mesmo lugar em que será usado. As funções lambda em Java tem a sintaxe definida como (argumento) -> (corpo).

Reference Method: Permite fazer referência a um método ou construtor de uma classe (de forma funcional) e assim indicar que ele deve ser utilizado num ponto específico do código, deixando-o mais simples e legível. Para utilizá-lo, basta informar uma classe ou referência seguida do símbolo "::" e o nome do método sem os parênteses no final.
É uma forma de simplificar um lambda, e o lambda é uma forma de simplificar uma classe anônima.

Streams API: Traz uma nova opção para a manipulação de coleções em Java seguindo os princípios da programação funcional. Proporciona uma forma diferente de lidar com conjuntos

de elementos, oferecendo ao desenvolvedor uma maneira simples e concisa de escrever código que resulta em facilidade de manutenção e paralelização sem efeitos indesejados em tempo de execução.

CÓDIGOS, EXERCÍCIOS E DESAFIOS REALIZADOS! :D

Os códigos estão disponíveis no meu outro repositório, no link a seguir:
https://github.com/LanghiDev/Java-Collections-dio