Collections

Uma Collection é uma Estrutura de Dados onde é realizado o processamento de um conjunto de dados com a ajuda de métodos predefinidos.

Este framework de coleções fornecem um conjunto de interfaces e classes para armazenar e manipular grupos de objetos de maneira eficiente e flexível. Na aula de hoje, veremos algumas maneiras de usar essa interface e classes deste framework Java.

A Collection contém os seguintes elementos:

Interfaces: são tipos abstratos que representam as coleções e permitem que a collection seja manipulada e orientada para a interface e não para a implementação.

Implementações: são as implementações concretas das interfaces.

Algoritmo: são os métodos que realizam as operações sobre os objetos das coleções. Esses métodos podem fazer serviços de busca e organização.

HIERARQUIA

A Java Collection Framework (JCF) é uma arquitetura unificada para representar e manipular a coleção de dados, e fornecem um conjunto de interfaces e classes para armazenar e manipular grupos de objetos de maneira eficiente e flexível. Veremos agora algumas dessas interfaces e classes desse framework Java.

Collection:

Essa é a interface raiz e está no topo da hierarquia. Algumas de suas subinterfaces incluem “List”, “Set” e “Queue”.

Os seus métodos principais são: “add()”, “remove()”, “size()”, “contains()”, “iterator()”, entre outros.

List:

Representa uma coleção ordenada de elementos que podem ou não ser duplicados. A coleção ordenada é representada por uma lista sequenciada.

As suas implementações são: “ArrayList”, “LinkedList”, “Vector” e “Stack”.

Os métodos principais são: “get(int index)”, “add(int index, E element)”, “remover(int index)”, “set(int index, E element)”.

Set:

Representa uma coleção que não perite elementos duplicados.

As suas implementações são: “HashSet”, “LinkedHashSet” e “TreeSet”.

Os métodos principais são: “add(E e)”, “remove(object o)”, “size()” e “contains(Object o)”.

Queue:

Representa uma coleção usada para armazenar elementos antes do processamento.

As suas implementações são: “LinkedList” e “PriorityQueue”.

Os métodos principais são: “offer(E e)”, “poll()”, “peek()” e “remove()”.

Map:

Representa uma coleção de pares chave-valor. Embora não faça parte da collection, é frequentemente usada junto com as coleções.

As suas implementações são: “HashMap”, “LinkedHashMap”, “TreeMap”, “Hashtable”.

Os seus métodos principais são: “put(K key, V value)”, “get(Object key)”, “remove(Object key)”, “constainsKey(Object key)”, “keyset()” e “values()”.

Abaixo observamos o resumo da hierarquia em um Diagrama:

Map

LinkedList

ArrayDeque

PriorityQueue

Queue

SortedSet

HashSet

LinkedHashSet

Set

TreeSet

Stack

Vector

LinkedList

ArrayList

List

Collection

Iterable

HashSet

LinkedHashMap

Hashtable

SortedMap

TreeMap

No Diagrama acima, conseguimos ver duas interfaces, sendo elas: Collections e a Map. Vale lembrar que interfaces é um recurso que “obriga” um determinado grupo de classes a ter métodos ou prioridades em comum, para existir em um determinado contexto.

Na Collections temos as interfaces List, Set e Queue:

Na List encontramos os métodos ArrayList, LinkedList, Vector, Stack.

Em Set encontramos os métodos HashSet, LinkedHashSet, SortedSet e TreeSet (encontrado dentro do SortedSet).

Na Queue encontramos os métodos LinkedList, ArrayDeque e PriorityQueue.

A outra interface que encontramos no diagrama é o Map, que há implementado dentro de si os seguintes métodos: HashMap, LinkedHashMap, Hashtable, sortedMap e TreeMap implementado ao SortedMap.

Conclusão

Referências

Java Collections: Como utilizar Collections

https://www.devmedia.com.br/java-collections-como-utilizar-collections/18450

Collections (Coleções)

https://github.com/Leon4rdoalves/CookBook-Java/blob/main/15.md

Entendendo interfaces em Java

https://www.devmedia.com.br/entendendo-interfaces-em-java/25502