**Trabalhando com Collections Java**

**O que é Collections?**

Collection é um objeto que agrupa múltiplos elementos (variáveis primitivas ou objetos) dentro de uma única unidade.

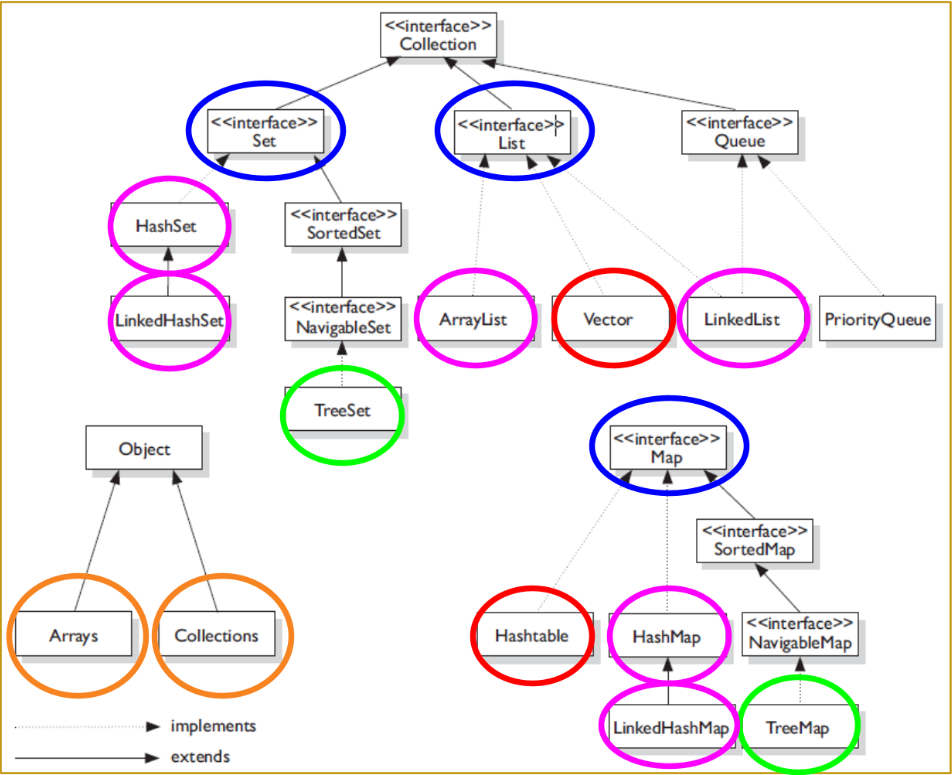
Serve para armazenar e processar conjuntos de dados de forma eficiente.

Antes do Java 2 (JDK 1.2), a implementação de coleções na linguagem Java incluía poucas classes e não tinha a organização de um framework.

**Composição Collections**

* **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.

**Hierarquia Collections**

****

**Coleções com iterações ordenadas: Listas**

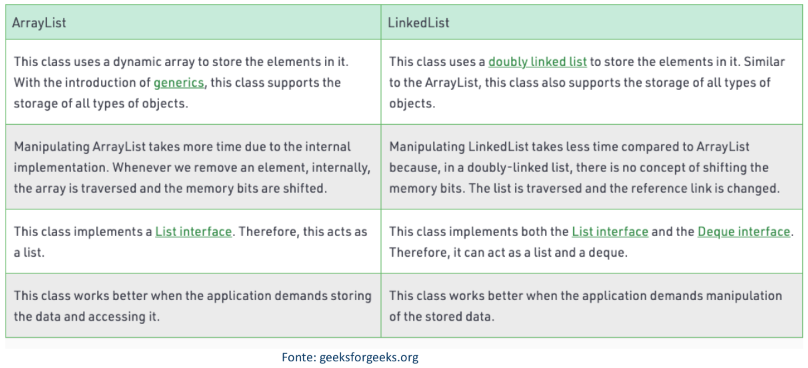
**1. Características da interface List**

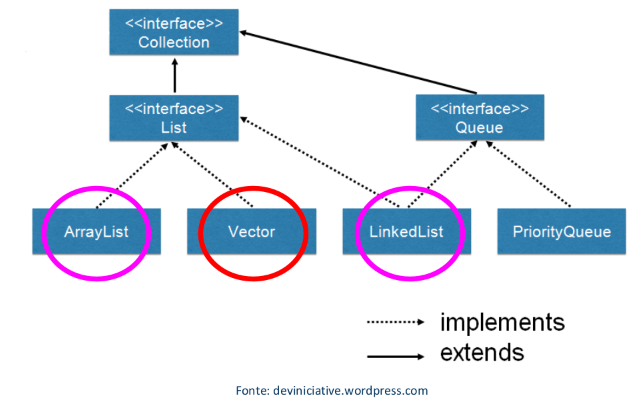
**2. Como e quando utilizar ArrayList e LinkedList**

* Arraylist usa uma array para poder guardar os elementos inseridos e deve ser utilizado onde mais operações de pesquisa são necessárias
* Linkedlist usa uma estrutura lista duplamente ligada (um pouco mais rápido) para guardar os elementos e deve ser usado onde mais operações de inserção e exclusão são necessárias.

**java.util.List**

* + Elementos duplicados e garante ordem de inserção





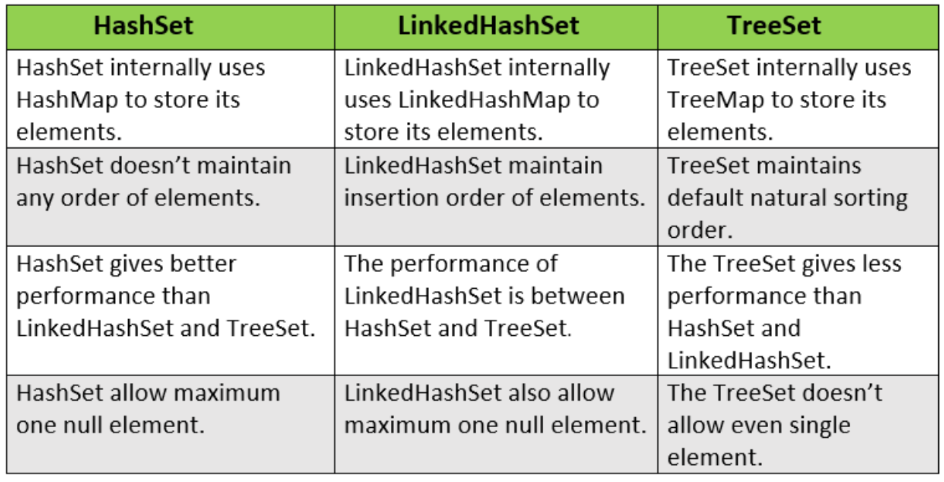
Método Comparable VS Comparable documentação:

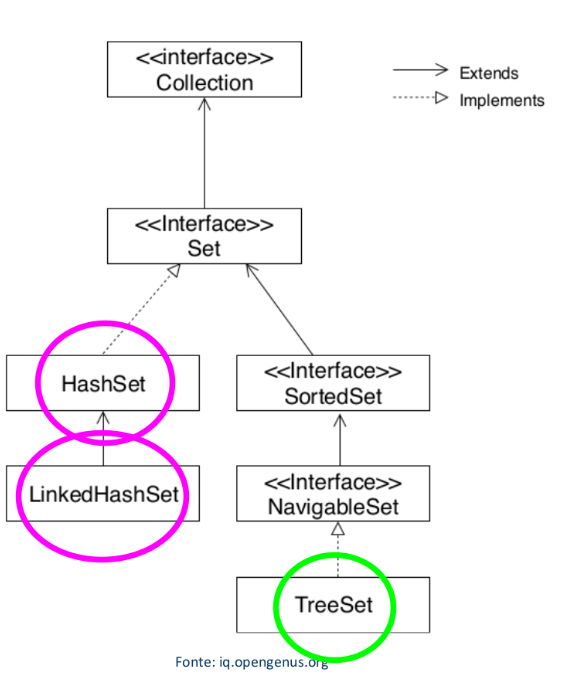
<https://www.geeksforgeeks.org/comparable-vs-comparator-in-java/>

Coleções singulares SET

Java.util.Set

* Não permite elementos duplicados, diferente da list.
* Não possui índice



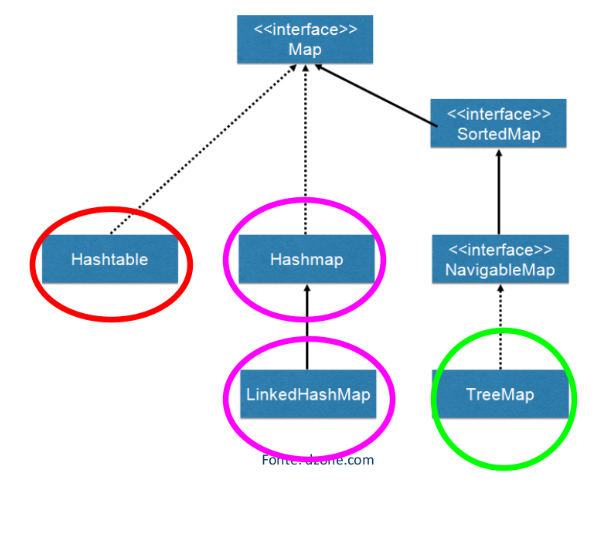


**Coleções de Pares: Map**

**Características da interface Map:**

**Java.util.Map**

* + - * Elementos únicos (key) para cada valor (value)
      * É um objeto que dentro podemos colocar múltiplos elementos, tanto outros objetos como variáveis primitivas. Cada elemento que tem dentro do map tem uma chave e um valor, as chaves desse elemento têm que ter valores únicos. Não podemos colocar chaves repetidas dentro do mesmo MAP.
      * HashMap e LinkedHashMap quando vamos comparar os métodos Get, Put, ContainsKey e Remove contém a mesma performance, já o TreeMap demora um pouco.
      * HashMap intera os elementos aleatoriamente, o LinkedHashMap é de acordo com a ordem de inserção, já o TreeMap é de acordo com a ordem natural das chaves.
      * Sobre chaves nulas, o HashMap e o LinkedHashMap permitem e o TreeMap não permite chaves nulas.
      * Sobres as interfaces que HashMap e LinkedHashMap implementam é o Map, já o TreeMap implementa map, SortedMap e NavigableMap.
      * Originalmente o HashMap, LinkedHashMap e o TreeMap não são sincronizáveis, mas podemos utilizar da Collections a SynchronizedMap() para poder sincronizar.
      * Na estrutura de dados o HashMap utiliza List of Buckets, o LinkedHashMap utiliza Doubly Linked List of Buckets e o TreeMap utiliza Red-Black Tree.



Java Streams

Classe Anônima

* + A classe anônima em Java é uma classe não recebeu um nome e é tanto declarado e instanciado em uma única instrução. Você deve considerar o uso de uma classe anônima sempre que você precisa para criar uma classe que será instanciado apenas uma vez.

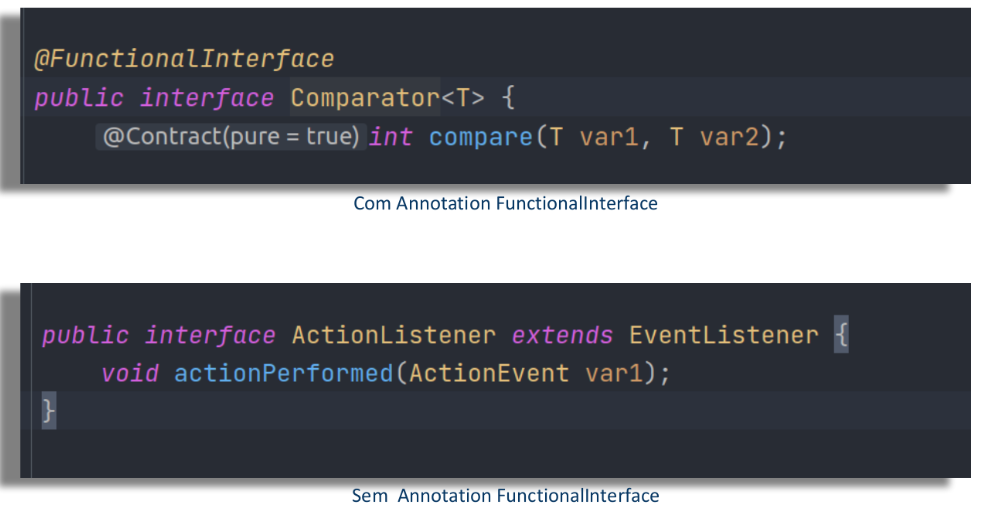




Functional Interface

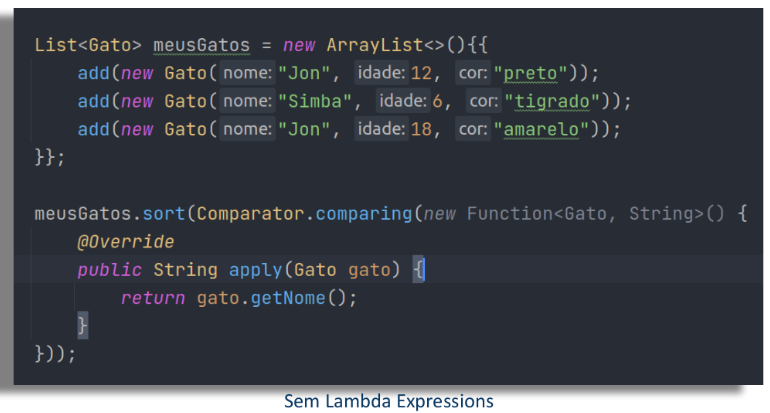
Qualquer interface com um SAM (Single Abstract Method) é uma interface funcional e sua implementação pode ser tratada como expressões lambda.

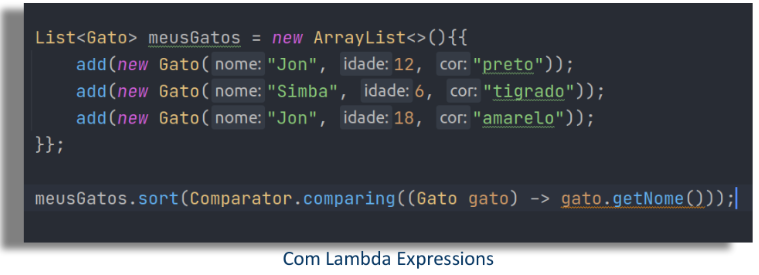
* + Comparador;
  + Consumer;
  + Function;
  + Predicate;



Lambda

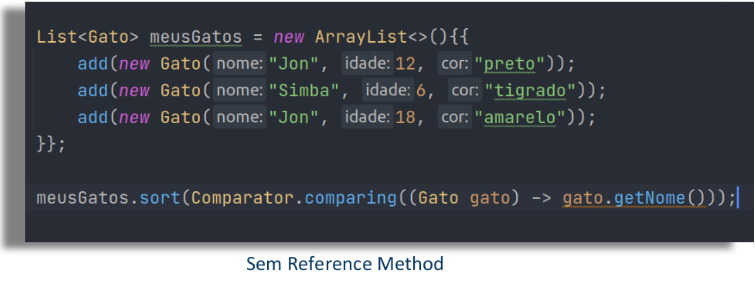
Uma função lambda é uma função sem declaração, isto é, não é necessário colocar um nome, um tipo de retorno e o modificador de acesso. A idéia é que o método seja declarado no mesmo lugar em que será usado. As funções lambda em Java tem a sintaxe definida como (argumento) -> (corpo). Simplificação da classe anônima.

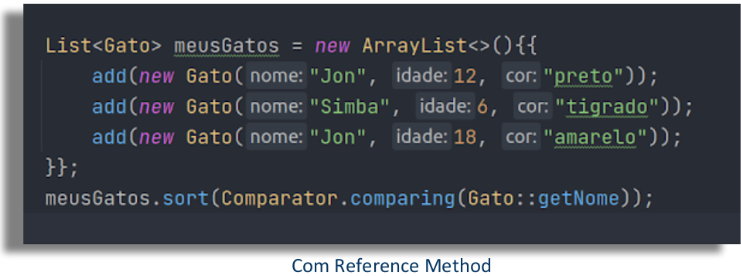




Reference Method

Method Reference é um novo recurso do Java 8 que 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 a forma de simplificar o Lambda.





Streams API

A Streams API traz uma nova opção para a manipulação de coleções em Java seguindo os princípios da programação funcional. Combinada com as expressões lambda, ela 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.



Fonte(Collection) Op. Intermediaria Op. Final

Arquivos para a leitura:

* + Qualquer interface com um SAM (Single Abstract Method) é uma interface funcional e sua implementação pode ser tratada como expressões lambda.
    - * <https://www.baeldung.com/java-8-functional-interfaces>
  + É uma função anônima com a qual podemos lidar como um cidadão de língua de primeira classe. Por exemplo, podemos passá-lo ou retorná-lo de um método.
    - * <https://www.baeldung.com/java-8-lambda-expressions-tips>
  + É um tipo especial de expressões lambda. Eles costumam ser usado para criar expressões lambda simples fazendo referência a métodos existentes.
    - * <https://www.baeldung.com/java-method-references>
  + Stream - Fluxo de dados que não altera a coleção
    - * <https://www.oracle.com/br/technical-resources/articles/java-stream-api.html>
      * <https://medium.com/@racc.costa/streams-no-java-8-e-no-java-9-36177c5c3313>
      * <https://winterbe.com/posts/2014/07/31/java8-stream-tutorial-examples/>
  + Operações finais - Fecha o fluxo
    - * <https://www.geeksforgeeks.org/stream-in-java/>
  + <https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/Optional.html>
  + <https://www.baeldung.com/java-optional>

Links para mais aulas:

Implementando Collections e Streams com Java

<https://web.dio.me/course/aprenda-collection-e-streams-na-linguagem-java/learning/c9f8940a-be04-4aa9-bdeb-ac3607ed8eec?back=/track/inter-java-developer&bootcamp_id=a531bc7a-f29e-4293-85eb-e4efd6072f2b>

Desenvolvimento Avançado em Java

<https://web.dio.me/course/desenvolvimento-avancado-em-java/learning/ac0c022e-a9e7-4898-abea-a9844d318925?back=/track/inter-java-developer&bootcamp_id=a531bc7a-f29e-4293-85eb-e4efd6072f2b>

Aprenda o que é estrutura de dados e algoritmos ✔

<https://web.dio.me/course/aprenda-o-que-sao-estrutura-de-dados-e-algoritmos/learning/a99f9576-69e9-4187-b3a7-e7ada5e5d6ad?back=/track/inter-java-developer&bootcamp_id=a531bc7a-f29e-4293-85eb-e4efd6072f2b>