**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.

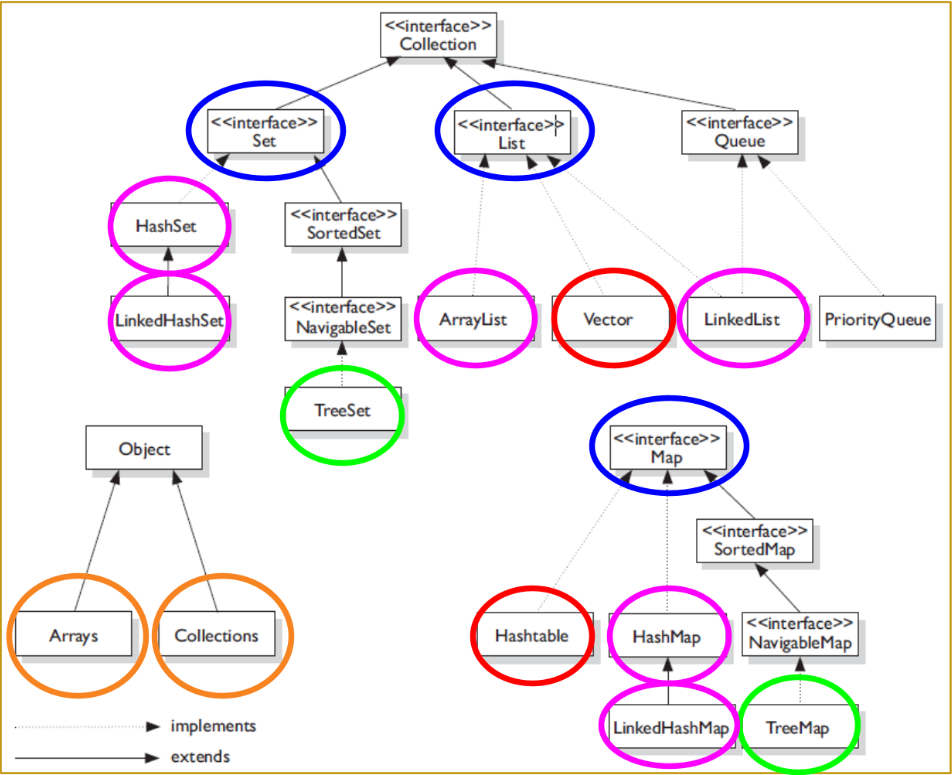
Serve para armazenar e processar conjuntos de dados de forma eficiente.

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**

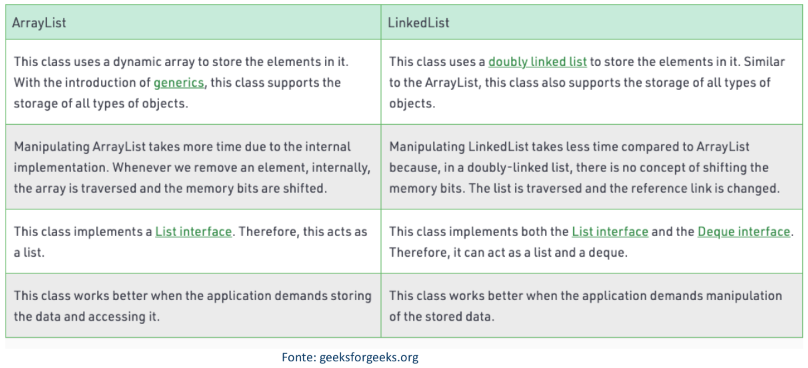
**1. Características da interface List**

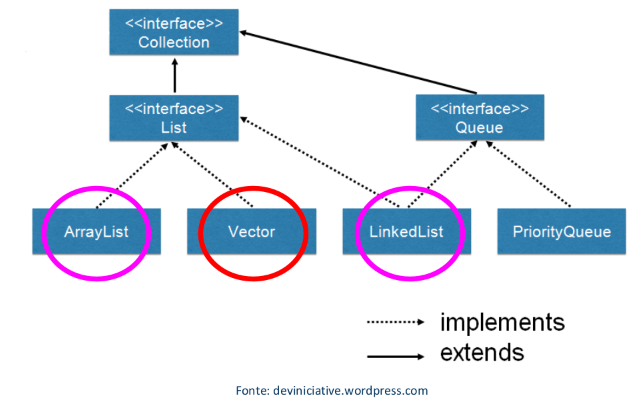
**2. Como e quando utilizar ArrayList e LinkedList**

* 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





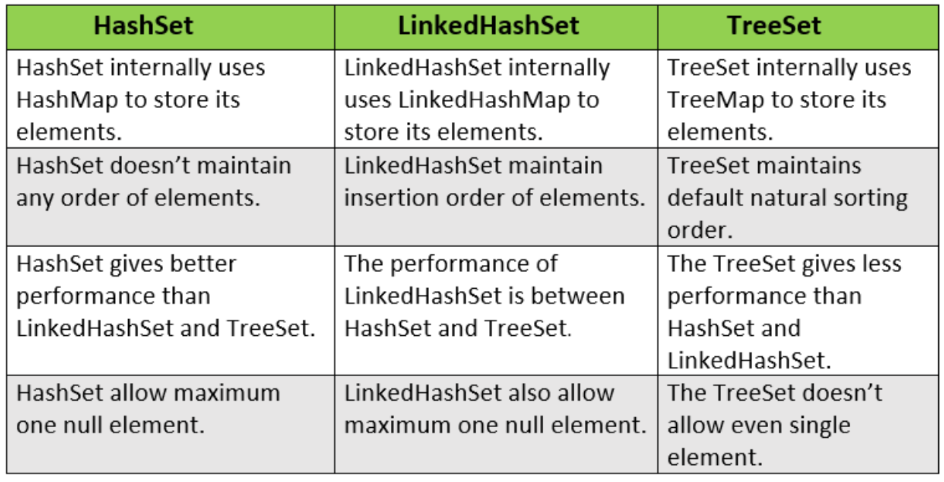
Método Comparable VS Comparable documentação:

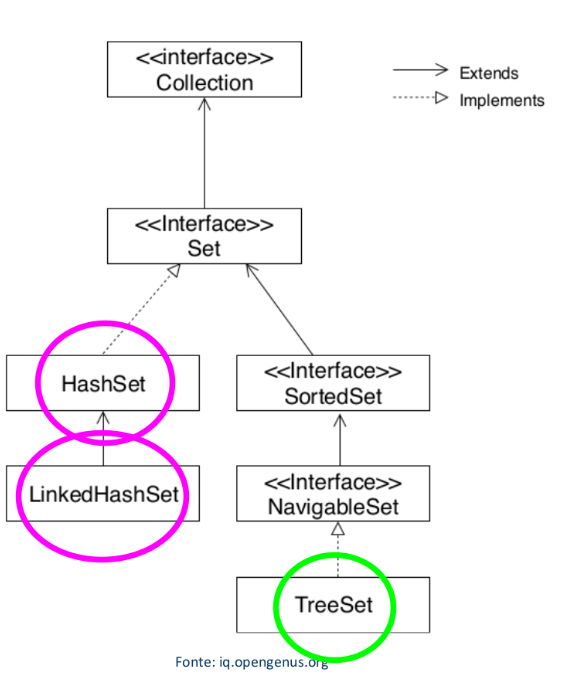
<https://www.geeksforgeeks.org/comparable-vs-comparator-in-java/>

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 vários 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.

