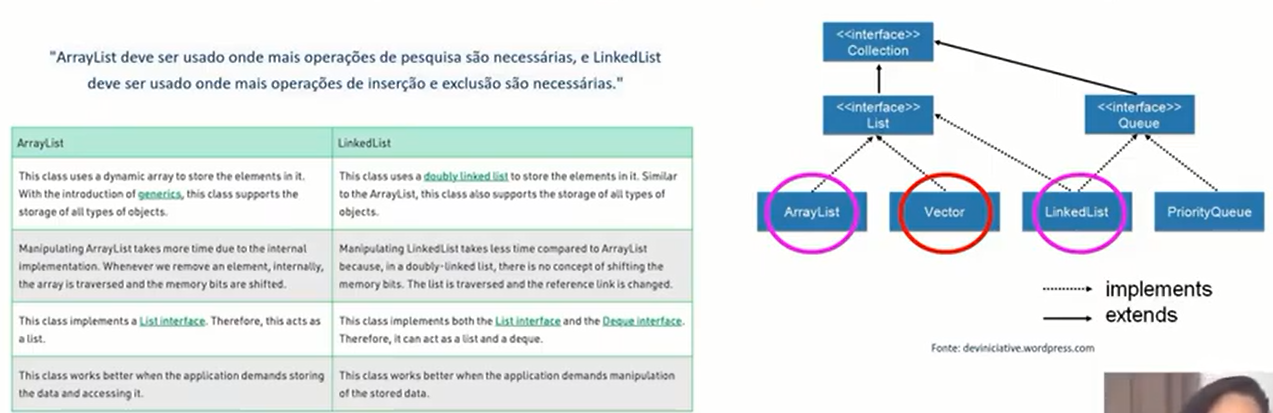
Aqui aprenderemos mais sobre o Collections framework e suas principais interfaces List, Set e Map.

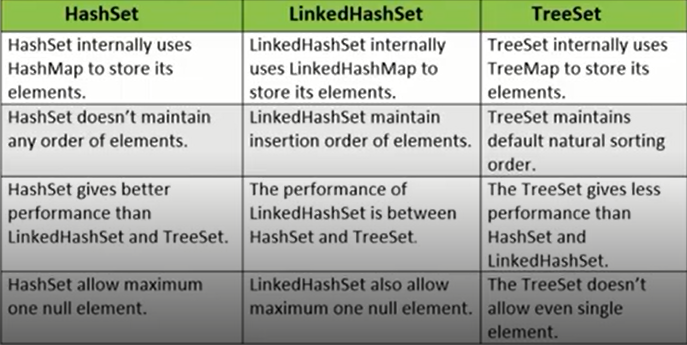
A Collection é um **objeto que agrupa multiplos elementos em uma única unidade**, uma variável com diversos elementos, sejam dados primitivos ou objetos, se tornando um conjunto. A collections assim permite armazenar e processar diversos dados de forma prática e eficiente com a organização de um framework antigamente isso era feito com arrays, o que era complexo e ineficiente. As interfaces que extendem diretamente a collections é a **List**, **Set** e **Queue**.

I**nterface** é um **contrato que quando assumido por uma classe deve ser implementado**.

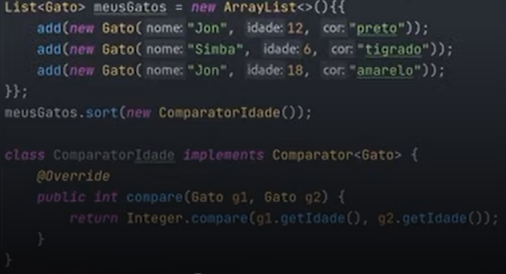
**Implementações e classes** são as maneiras de materialização das codificações de interface.

**Algorítmos** é uma sequência lógica, finita e clara de instruções para se resolver um problema.

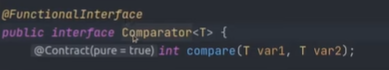
A **java.util.list** permite elementos duplicados e garante ordem de inserção.

O Java.util.**set** não permiti elementos duplicados e não possui indice, ou seja não possui métodos de get, index of, set, etc..

O java.util.**map** não extende da interface collection, fazendo parte do framework e permite colocar tanto outros objetos quanto variaveis primitivas dentro do objeto map, cada elemento do map possui uma **chave** e um **valor**, sendo a chave possuindo valores únicos ao map, podendo repetir o valor, **desde que não seja dentro do mesmo map**. Para passar um valor de chave utilizamos o comando *Map.put{key,value};*, sendo possível ordenar **os valores do mapa através dos valores da chave**.



**Classe anônima** é uma classe que não recebe nome sendo declarada e instanciada em uma única instrução, consideramos o uso de uma classe anônima quando criaremos uma classe que será utilizada apenas uma única vez.

**Functional interface** é uma interface com o SAM (**single abstract method**) é uma interface funcional e sua implementação pode ser tratada como expressões **lambda**. Aqui falaremos sobre as interfaces **comparator**, **consumer**, **function** e **predicate**.

Expressões **Lambda** são expressões sem declarações, sem precisar colocar nome, tipo de retorno e modificador de acesso. É um método que será utilizado no mesmo lugar que é mencionado (**como as classes anonimas**), as funções lambda em Java tem a sintaxe definida como *(argumento) -> (corpo)*.

**Method reference** é um recurso do Java 8 que permite referenciar um método ou construtor de uma classe e indicar que ele deve ser utilizado em um ponto específico do código, deixando- mais simples e legível. Para utiliza-lo coloca-se o simbolo “::” e o nome do método sem os parenteses no final.

**Streams API** traz uma nova opção para manipulação de coleções em Java seguindo os principios da programação funcional. Combinado com as expressões lambda, proporciona uma forma prática de lidar com conjuntos de elementos.

*Source 🡺 pipeline 🡺 terminal*