**Módulo 1: Aula de ArrayList**

ArrayList<Tipo> e é necessário instânciar uma nova classe, ex:

ArrayList<String> listaStrings = new ArrayList<>();

**Métodos:**

**Add:** Funciona como se fosse o push.

**Remove:** Existem 2 desses métodos, um deles recebe um índice e o outro recebe um obj.

**Get:** Recebe um índice e retorna o elemento que o possui, em uma determinada lista.

**Size:** Retorna o número de elementos em uma lista.

**Contains:** Retorna um boolean, caso o parâmetro recebido esteja dentro do conjunto. Esse método utiliza o **equals()** (método de objetos).

Existe a classe **Collections** com vários métodos estáticos, ela possui várias funcionalidades para lidar com coleções. Ex: Tem o método **sort** que ajuda a ordenar uma lista.

Collection**s** 🡪 Classe

Collection 🡪 Interface

**Módulo 2: Lista de objetos**

Diferente de uma String ou de tipos primitivos mais simples, o Collections.sort não sabe ordenar uma lista de Aula. De qual forma ele faria isso? Pelo nome da aula? Pela duração? Não daria para saber. Para que ele seja capaz de fazer isso, você precisa implementar a interface Comparable definindo um critério de comparação para os objetos desse tipo.

**Módulo 3: Relacionamentos com coleções**

A diferença de LinkedList e ArrayList é de performance, em alguns casos.

A ArrayList, como diz o nome, internamente usa uma array para guardar os elementos. Ela consegue fazer umas operações de maneira muito eficiente, como invocar o método get(indice). Se você precisa pegar o décimo quinto elemento, ele te devolve isso bem rápido. Onde uma ArrayList é lenta? Quando você for, por exemplo, inserir um novo elemento na primeira posição. Pois a implementação vai precisar mover todos os elementos que estão no começo da lista para a próxima posição. Se há muitos elementos, isso vai demorar... chamamos isso em computação de consumo de tempo linear.

Já a **LinkedList** possui uma grande vantagem aqui. Ela utiliza a estrutura de dados chamada lista ligada. Ela é muito rápida para adicionar e remover elementos na cabeça da lista, isso é, na primeira posição. Mas ela é lenta se você precisar acessar um determinado elemento, pois a implementação precisará percorrer todos os elementos até chegar ao décimo quinto, por exemplo.

**Módulo 5: O poder dos Sets**

Set é uma interface

HashSet é a classe

A diferença de um Set, comparado a uma List, é que o Set é um conjunto de dados únicos (não aceita elementos repetidos) e não há garantia de uma ordem padrão nesse conjunto.

Não possui o método sort para ordenação.

Possui vantagem de velocidade e performance.

Caso precise de uma ordenação, use o List.

O Set não possui os métodos que recebem o index no parâmetro, como o **.get()**. Porém os métodos em comum q ele possui com o ArrayList funcionam com maior velocidade.

**Módulo 7: Equals e Hashcode**

O método Equals (método de objetos) é diferente de igual (mesma ref na memória), precisamos dizer que um aluno é igual (equals) a outro caso tenham o mesmo nome e número de matrícula, e para isso teremos que reescrever o método Equals e o método HashCode() que é um identificador único de referência de determinado objeto na collection.

O próprio obj String possui o método hashCode().

Se 2 obj são iguais (equals), eles precisam ter o mesmo hashCode(), portanto se sobrescrever um método tbm terá que sobrescrever o outro.

**Módulo 8: Outros Sets e Iterators**

O TreeSet só funciona para classes Comparable, ela guarda os elementos na ordem natural dos obj (comparable).

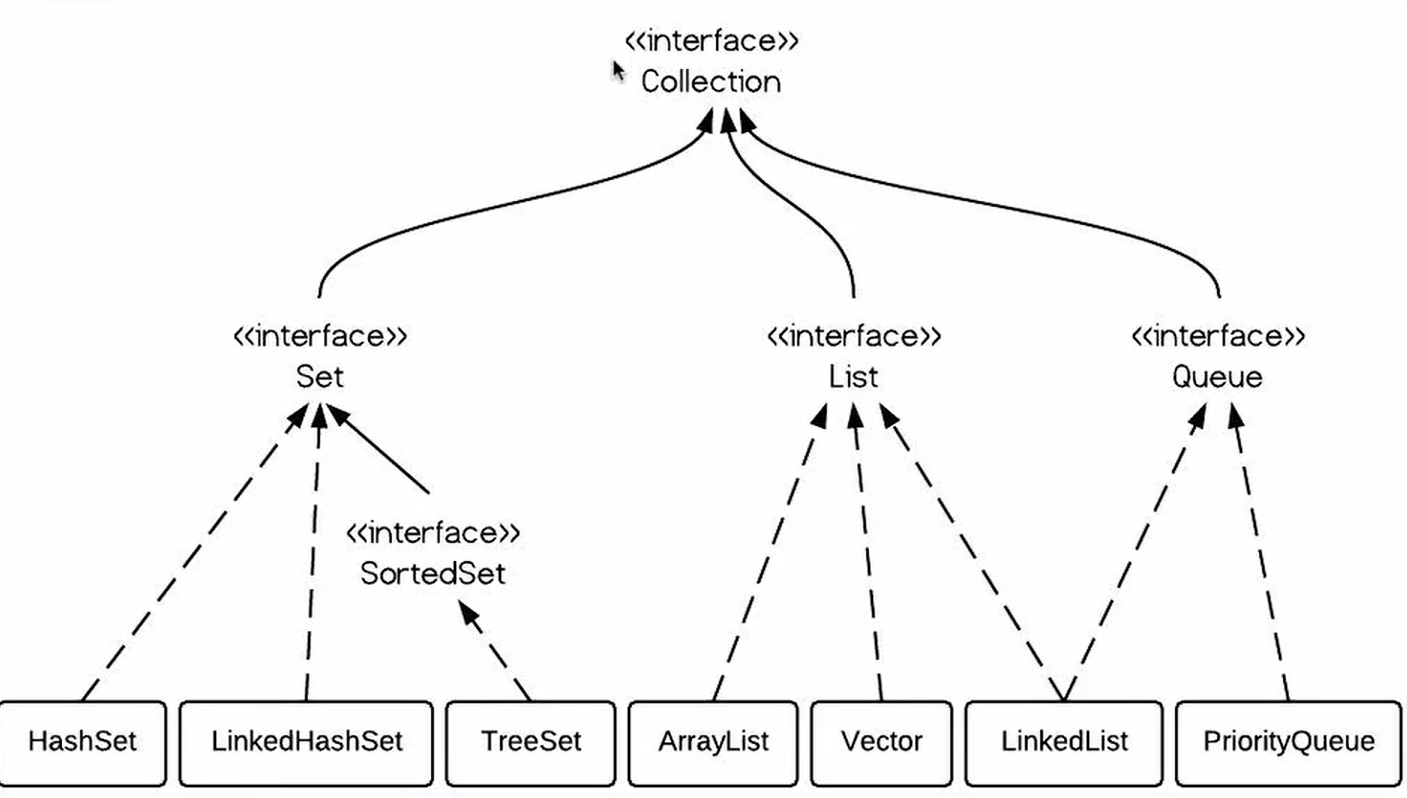
Todas as coleções possuem um Iterator (obj) que serve para iterarmos entre os objetos dentro dessa coleção.

**Módulo 9: Qual Collection Usar?**

Collection é uma interface.

**Interfaces** filhas da Collections: **Set** e **List**

**Classes** que implementam essas interfaces: HashSet, LinkedHashSet, TreeSet, ArrayList, LinkedList.



**Módulo 10: Mapas**

Map é uma interface que não é filha de Collection, ela mapeia chave e valor.

**Map<Chave, Valor>**

A implementação dessa interface é pela classe **HashMap**<>();

Na classe de curso, vamos criar uma propriedade matriculaParaAluno e modificar a nossa função de matricular um aluno.

Também faz buscas com alta velocidade, semelhante ao HashSet.

É importante que a chave do Map seja **única**.

Use o método .**put()** para adicionar um elemento no Map.