**Comparação e Aplicações Práticas de Coleções Avançadas em Java: Set, LinkedHashSet, TreeSet e Map**

**1. Set (Interface)**

* **Características**:
  + Não permite elementos duplicados.
  + Ordem dos elementos não é garantida.
* **Uso comum**: Quando você precisa de uma coleção única de elementos, sem se preocupar com a ordem.
* **Exemplo**:

import java.util.HashSet;

public class SetExample {

public static void main(String[] args) {

Set<String> set = new HashSet<>();

set.add("A");

set.add("B");

set.add("A"); // Elemento duplicado será ignorado.

System.out.println(set); // Saída: [A, B]

}

}

**2. LinkedHashSet**

* **Características**:
  + Mantém a ordem de inserção dos elementos.
  + Não permite duplicados.
* **Uso comum**: Quando você precisa de uma coleção única de elementos e deseja manter a ordem de inserção.
* **Exemplo**:

import java.util.LinkedHashSet;

public class LinkedHashSetExample {

public static void main(String[] args) {

LinkedHashSet<Integer> linkedHashSet = new LinkedHashSet<>();

linkedHashSet.add(10);

linkedHashSet.add(20);

linkedHashSet.add(10); // Ignorado, pois é duplicado.

System.out.println(linkedHashSet); // Saída: [10, 20]

}

}

**3. TreeSet**

* **Características**:
  + Mantém os elementos em ordem natural (ou por um comparador personalizado).
  + Não permite duplicados.
* **Uso comum**: Quando você precisa de uma coleção única de elementos, mas em ordem crescente ou personalizada.
* **Exemplo**:

import java.util.TreeSet;

public class TreeSetExample {

public static void main(String[] args) {

TreeSet<String> treeSet = new TreeSet<>();

treeSet.add("Banana");

treeSet.add("Apple");

treeSet.add("Cherry");

System.out.println(treeSet); // Saída: [Apple, Banana, Cherry]

}

}

**4. Map (Interface)**

* **Características**:
  + Armazena pares de chave-valor.
  + Cada chave é única, mas os valores podem se repetir.
* **Implementações comuns**:
  + **HashMap**: Não garante ordem.
  + **LinkedHashMap**: Mantém a ordem de inserção.
  + **TreeMap**: Ordena as chaves em ordem natural ou personalizada.
* **Exemplo com HashMap**:

import java.util.HashMap;

public class MapExample {

public static void main(String[] args) {

HashMap<Integer, String> map = new HashMap<>();

map.put(1, "One");

map.put(2, "Two");

map.put(1, "Duplicate Key"); // Substitui o valor anterior.

System.out.println(map); // Saída: {1=Duplicate Key, 2=Two}

}

}

**Comparação entre as coleções**

|  |  |  |  |
| --- | --- | --- | --- |
| Coleção | Permite Duplicados? | Mantém Ordem? | Ordenação |
| Set | Não | Não | Não |
| LinkedHashSet | Não | Sim (inserção) | Não |
| TreeSet | Não | Sim | Ordem natural/customizada |
| Map | Sim (valores) | Depende da implementação | Depende da implementação |

Esses exemplos e características ajudam a compreender como e quando usar essas coleções em seus estudos e projetos futuros.