Difference between HashSet and TreeSet in Java

Difference between HashSet and TreeSet in Java

There are several differences between a HashSet and a `TreeSet` are similar to what we discussed as a difference between `TreeMap` and `HashMap`. Anyway, Set and Map are two completely different interfaces so we will revisit those differences here. Probably the most important difference between HashSet and `TreeSet` is the performance. `HashSet` is faster than `TreeSet` which means if you need performance use `HashSet` but `HashSet` doesn't provide any kind of ordering so if you need ordering then you need to switch to `TreeSet` which provides sorting of keys.

Sorting can be natural order defined by a Comparable interface or any particular order defined by a Comparator interface in Java.

Apart from the differences between `HashSet` and `TreeSet`, there are some common things between them. let's see what is common between `HashSet` and `TreeSet` in Java.

By the way, this is one of the popular Java collection interview questions much like ArrayList vs Vector and Hashtable vs HashMap. If you are going for any Java programming interview, it's worth preparing.

What is Common in HashSet and TreeSet in Java

As I said there are a lot of things that are common between HashSet and TreeSet in Java, let's have a look:

1)Both `HashSet` and `TreeSet` implements `java.util.Set` interface which means they follow contract of `Set` interface and doesn't allow any duplicates.

2)Both `HashSet` and `TreeSet` are not thread-safe and not synchronized. Though you can make them synchronized by using the `Collections.synchronizedSet()` method.

3) The third similarity between `TreeSet` and `HashSet` is that Iterator of both classes is fail-fast in nature. They will throw `ConcurrentModificationException` if Iterator is modified once Iterator is created. this is not guaranteed and application code should not rely on this code but Java makes best effort to fail as soon as it detects a structural change in underlying `Set`.

HashSet vs TreeSet in Java

Now let's see a couple of differences between `HashSet` vs `TreeSet` in Java. This is enough to decide whether you should use `HashSet` or `TreeSet` in a given scenario.

1) The first major difference between `HashSet` and `TreeSet` is performance. `HashSet` is faster than `TreeSet` and should be the preferred choice if sorting of elements is not required. TreeSet is internally backed by a Red-Black tree. For a detailed description of the

Red-Black Tree, you should read a good book on data structure and algorithms like Introduction to Algorithms by Thomas Corman.

The performance difference comes from the underlying data structure used by TreeSet and HashSet i.e. a tree and a hash table. Adding an element of a tree is slower than adding it to a hash table but it is still much faster than adding it into the right place in the linked list or array.

If the tree contains n elements, then an average log2N comparisons are required to find the correct position for a new element. For example, if the tree contains 1000 elements then adding a new element requires about 10 comparisons.

2) Second difference between `HashSet` and `TreeSet` is that `HashSet` allows null object but `TreeSet` doesn't allow null Object and throw NullPointerException, Why, because `TreeSet` uses compareTo() method to compare keys and `compareTo()` will throw `java.lang.NullPointerException` as shown in below example :

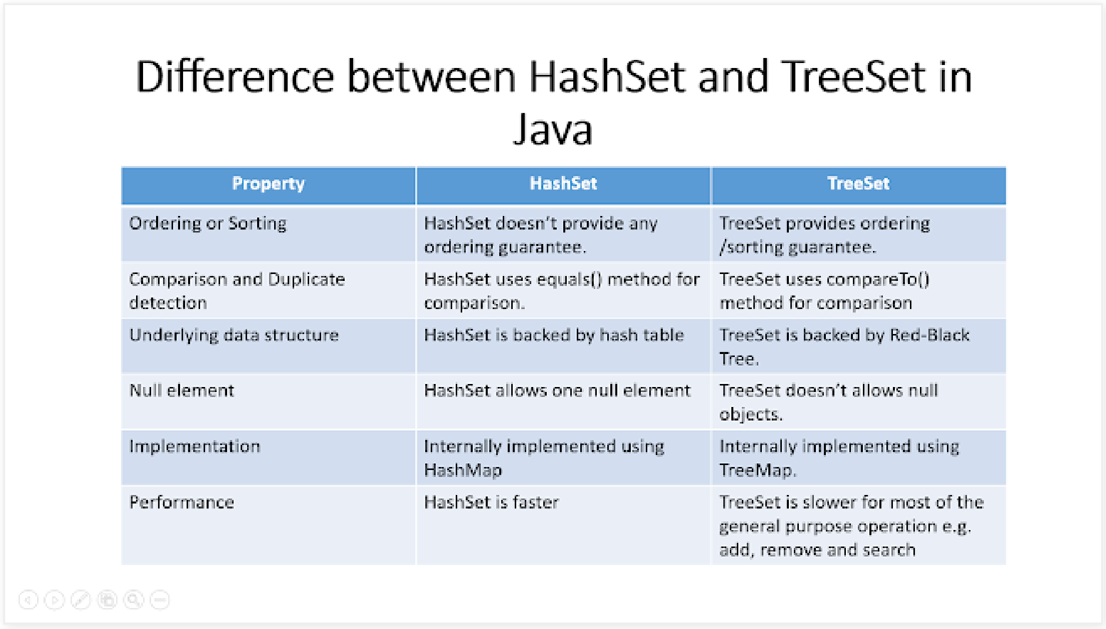
```
HashSet<String> hashSet = new HashSet<String>();
hashSet.add("Java");
hashSet.add(null);

TreeSet<String> treeSet = new TreeSet<String>();
treeSet.add("C++");
treeSet.add(null); //Java.lang.NullPointerException
Output:
Exception in thread "main" java.lang.NullPointerException
        at java.util.TreeMap.put(TreeMap.java:541)
        at java.util.TreeSet.add(TreeSet.java:238)
        at test.CollectionTest.main(CollectionTest.java:27)
Java Result: 1
```

3) Another significant difference between `HashSet` and `TreeSet` is that `HashSet` is backed by `HashMap` while `TreeSet` is backed by TreeMap in Java.

4) One more difference between `HashSet` and `TreeSet` which is worth remembering is that `HashSet` uses equals() method to compare two objects in Set and for detecting duplicates while `TreeSet` uses the `compareTo()` method for the same purpose. if equals() and compareTo() are not consistent, i.e. for two equal object equals should return true while `compareTo()` should return zero then it will break the contract of Set interface and will allow duplicates in Set implementations like `TreeSet`

5) Now the most important difference between `HashSet` and `TreeSet` is ordering. `HashSet` doesn't guarantee any order while `TreeSet` maintains objects in the Sorted order defined by either `Comparable` or `Comparator` method in Java.

Here is a nice summary slide of key differences between TreeSet and HashSet in Java, which compares both of these collections on ordering, sorting, performance, underlying data structure, the method used for duplicate detection, and how they are implemented in JDK.



Difference between HashSet and TreeSet in Java

Property	HashSet	TreeSet
Ordering or Sorting	HashSet doesn't provide any ordering guarantee.	TreeSet provides ordering /sorting guarantee.
Comparison and Duplicate detection	HashSet uses equals() method for comparison.	TreeSet uses compareTo() method for comparison
Underlying data structure	HashSet is backed by hash table	TreeSet is backed by Red-Black Tree.
Null element	HashSet allows one null element	TreeSet doesn't allows null objects.
Implementation	Internally implemented using HashMap	Internally implemented using TreeMap.
Performance	HashSet is faster	TreeSet is slower for most of the general purpose operation e.g. add, remove and search

That's all on the **difference between `HashSet` and `TreeSet` in Java**. Use `HashSet` if you don't need sorting and looking for better performance while `TreeSet` is the first choice if you need to maintain objects in sorted order in Java.