Difference between HashSet and TreeSet

HashSet and TreeSet are concrete implementations of Set interface.

Previously, I had written posts on difference between [fail-safe vs fail-fast iterator](http://data-structure-learning.blogspot.com/2015/05/difference-between-fail-safe-iterator.html), [Collection vs Collections](http://data-structure-learning.blogspot.com/2015/05/difference-between-collection-and.html), [HashMap vs Hashtable](http://data-structure-learning.blogspot.com/2015/05/difference-between-hashtableand-hashmap.html), [ArrayList vs LinkedList](http://data-structure-learning.blogspot.com/2015/05/difference-between-arraylist-and_28.html), [ArrayList vs Vector](http://data-structure-learning.blogspot.com/2015/05/difference-between-arraylist-and-vector.html), [List vs Set](http://data-structure-learning.blogspot.com/2015/05/difference-between-list-and-set.html), [poll() vs remove() method in Queue](http://data-structure-learning.blogspot.com/2015/05/difference-between-poll-and-remove-in.html), [Iterator vs ListIterator](http://data-structure-learning.blogspot.com/2015/05/5-difference-between-iterator-and.html) and [Comparable vs Comparator](http://data-structure-learning.blogspot.com/2015/06/difference-between-comparable-and.html)

There are several parameters through which we will compare HashSet and TreeSet.

Ordering of Objects: HashSet is not ordered Collection. Let us see the below program and see the output for same.

**package** org.collections;

**import** java.util.HashSet;

**import** java.util.Set;

**public** **class** **HashSetDemo** {

**public** **static** **void** main(String[] args) {

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

set.add("Monday");

set.add("Tuesday");

set.add("Wednesday");

set.add("Thursday");

set.add("Friday");

set.add("Saturday");

set.add("Sunday");

System.***out***.println(set);

}

}

Output:

[Friday, Monday, Saturday, Sunday, Thursday, Tuesday, Wednesday]

TreeSet sorts the elements in natural order. All elements inserted into the Set must implement the Comparable interface. If it objects does not implement Comparable Interface then we can supply our own Comparable or Comparator object too.

**package** org.collections;

**import** java.util.Set;

**import** java.util.TreeSet;

**public** **class** TreeMapDemo {

**public** **static** **void** main(String[] args) {

Set<String> set = **new** TreeSet<String>();

set.add("Monday");

set.add("Tuesday");

set.add("Wednesday");

set.add("Thursday");

set.add("Friday");

set.add("Saturday");

set.add("Sunday");

System.***out***.println(set);

}

}

String class implements Comparable interface. So this program works properly.

Output

[Friday, Monday, Saturday, Sunday, Thursday, Tuesday, Wednesday]

Output is sorted in lexical order (A comes before B, B comes before C).

Null Element: HashSet can store one null element. Insertion of null element will replace old null element. TreeSet does not allow null elements. Inserting null into TreeMap will result in NullPointerException.

Performance: HashSet takes constant time on add, remove & contains. HashSet is backed by HashMap(Keys sections of Map is Set). TreeSet takes O(log n) times for add, remove & contains.

Speed: HashSet outperforms TreeSet because HashSet provides constant time access. Whereas TreeSet provides O(log n) time for operations. Hence HashSet is faster than TreeSet.

Methods: TreeSet definitely has rich set of methods than HashSet. As TreeSet implements NavigableSet<> and NavigableSet<> extends SortedSet<>, TreeSet is very rich in methods. HashSet has methods of Set<> interface.

Backing implementation: HashSet has backing implementation of HashMap and TreeSet has backing implementation of TreeMap.

equals and compareTo: HashSet uses equals() method for comparison and TreeSet uses compareTo() method for ordering of Objects.

In next post we will see the similarities of HashSet and TreeSet and when to use either of them.