HashSet Class in Java with example

BY CHAITANYA SINGH | FILED UNDER: JAVA COLLECTIONS

This class implements the Set interface, backed by a hash table (actually a HashMap instance). It makes no guarantees as to the iteration order of the set; in particular, it does not guarantee that the order will remain constant over time. This class permits the null element. This class is not synchronized. However it can be synchronized explicitly like this: `Set s = Collections.synchronizedSet(new HashSet(...));`

Points to Note about HashSet:

1. HashSet doesn't maintain any order, the elements would be returned in any random order.
2. HashSet doesn't allow duplicates. If you try to add a duplicate element in HashSet, the old value would be overwritten.
3. HashSet allows null values however if you insert more than one nulls it would still return only one null value.
4. HashSet is non-synchronized.
5. The iterator returned by this class is fail-fast which means iterator would throw `ConcurrentModificationException` if HashSet has been modified after creation of iterator, by any means except iterator's own remove method.

HashSet Example

```
import java.util.HashSet;
public class HashSetExample {
   public static void main(String args[]) {
      // HashSet declaration
      HashSet<String> hset =
                new HashSet<String>();

      // Adding elements to the HashSet
      hset.add("Apple");
      hset.add("Mango");
      hset.add("Grapes");
      hset.add("Orange");
      hset.add("Fig");
      //Addition of duplicate elements
      hset.add("Apple");
      hset.add("Mango");
      //Addition of null values
      hset.add(null);
      hset.add(null);

      //Displaying HashSet elements
      System.out.println(hset);
    }
}
```

Output:

```
[null, Mango, Grapes, Apple, Orange, Fig]
```

As you can see there all the duplicate values are not present in the output including the duplicate null value.