Difference between Set, List and Map

Set, List and Map are very important [interfaces](http://data-structure-learning.blogspot.com/2015/05/java-collections-part-2-interfaces.html) in [Java Collections Framework](http://data-structure-learning.blogspot.com/2015/05/java-collections-part-1.html).

We have studied List, Set and Map interfaces as individual. We also saw [difference between List and Set](http://data-structure-learning.blogspot.com/2015/05/difference-between-list-and-set.html) and [when to you either of them](http://data-structure-learning.blogspot.com/2015/05/when-to-use-list-and-set.html).

In this post we will compare all 3 of them i.e. Set, List and Map.

Let us first see all of them individually

* Set – Provides a mathematical abstraction of Set. No duplicates allowed. Unordered Collection.
* [List](http://data-structure-learning.blogspot.com/2015/05/java-collections-part-5list-interface.html) – Index and Ordered Collection. Duplicated allowed.
* [Map](http://data-structure-learning.blogspot.com/2015/05/java-collections-part-3map-interface.html) – Provides one to one mapping or injective function. Unordered Collection.

There are several concrete implementations of this interfaces provided in Collections Framework.

Now let us see the difference between these 3 interfaces.

* Ordered or Unordered Collection

1. List – **List** is **ordered collection**. List defines a contract of maintaining insertion order and its concrete implementations follow that contract.
2. Set – **Set** is **unordered collection**. The contract does not specify any type of insertion order. But it specify contract of unique elements in Set. LinkedHashSet maintains the insertion order. TreeSet maintains the sorting order.
3. **Map** – **Map** is **unordered collection**. The contract specifies that keys much be unique and values can be repeated for different keys. Insertion order is not maintained. LinkedHashMap maintains the insertion order. TreeMap maintains the sorting order.

* Duplicate elements

1. List – **List** can **hold duplicate elements** or objects in to it.
2. Set – **Set** interface has a **contract for unique elements** and all the implementation of Set interface uphold this contract.
3. Map – **Map** interface has **contract of unique keys**. Map has an entry which has key and value. All keys in Map must be unique; values can be unique or repeated.

* Null elements

1. List – **List** can hold **multiple null elements** as it allows duplicates elements.
2. Set – **Set** allows **single null element** because it has contract of unique elements.
3. Map- **Map** allows **Single null key and multiple null values**. It actually depends on various implementations. For example, Hashtable does not allow null key or value neither does ConcurrentHashMap. [HashMap vs. Hashtable](http://data-structure-learning.blogspot.com/2015/05/difference-between-hashtableand-hashmap.html) is popular interview question too.

Read about Popular implementations of this interfaces and When to use them?