Java Collections Part 3(Map Interface)

Map Interface does not exactly come under collections.

Signature of Map Interface

**public** **interface** Map<K,V>

K – Keys

V – Values

Now, in collections we have <E> whereas map supports <K, V>. Map is not collections and collections are not Map. Yet it is part of Java Collections Framework.

Let us try to understand what makes Map interface different from other interfaces.

Map take two parameters as object namely Key and Value. The key is used to hold a single Object. Single object can be integer object, String Object or ArrayList<E> object.

Map cannot contain duplicate keys. If same key comes in map then old values gets replaced by new values.

Map can contain duplicate values for different keys.

HashMap<K, V> is one of the most used class that implements Map<K, V> interface. The data structure implemented beneath HashMap<K, V> is hash table.

Hash table hashes the key using hashCode() method. If two different keys has same hashCode() then there is collision at place where data is to be inserted. This leads to concept called as collision resolution strategy. Widely used collision resolution strategy is Separate chaining.

Separate chaining means that if two different keys has same hashCode() then the index in table will point to another data structure commonly used is LinkedList. This LinkedList will hold several Keys and Value as objects into it.

Java 7 or prior to it Separate chaining was used as collision resolution strategy. If the hashCode is poor then performance can degrade and hence lead to time complexity of O(n) where n is number of elements in Map.

But as of Java 8 the collision resolution strategy is changed to balanced trees. So even if the hashCode is pop the performance is guaranteed to be O(log n) where n is number of elements in Map.

Later on we will discuss Hash table data structure in detail as there is much to learn in Hash table data structure.