**What is a Tree Map ?**   
  
Treemap class is like HashMap which stores key- value pairs . The major difference is that Treemap  sorts  
the key in ascending order.

Treemap is sorted according to the natural ordering of its keys, or by a Comparator provided at map creation time, depending on which constructor is used.  
This implementation provides guaranteed log(n) time cost for the containsKey, get, put and remove operations. Algorithms are adaptations of those in Cormen, Leiserson, and Rivest's Introduction to Algorithms.

**How TreeMap works in java** ?  
  
TreeMap is a Red-Black tree based NavigableMap implementation.In other words , it sorts the TreeMap object keys using Red-Black tree algorithm.

**So we learned that TreeMap uses Red Black tree algorithm internally to sort the elements.**   
  
  
Red Black algorithm is a complex algorithm . We should read the pseudo code of Red Black algorithm in order to understand the internal implementation .  
  
Red Black tree has the following properties :  
  
1. As the name of the algorithm suggests ,color of every node in the tree is either red or black.  
  
2. Root node must be Black in color.  
  
3. Red node can not have a red color neighbor node.  
  
4. All paths from root node to the null should consist the same number of black nodes .

**Why and when we use TreeMap ?**

We need TreeMap  to get the sorted list of keys in ascending order.

**What is the runtime performance of the get() method in TreeMap and HashMap  ,where n represents the number of elements ?**

TreeMap implementation provides guaranteed log(n) time cost for the containsKey,get,put and remove operations.

HashMap implementation provides constant-time performance for the basic operations (get and put), assuming the hash function disperses the elements properly among the buckets.  
  
One liner : TreeMap : log(n)   HashMap : Constant time performance assuming elements disperses properly

**What is "natural ordering" in TreeMap ?**  
  
"Natural" ordering is the ordering implied by the implementation of the Comparable interface by the objects used as keys in the TreeMap. Essentially, RBTree must be able to tell which key is smaller than the other key, and there are two ways to supply that logic to the RBTree implementation:  
  
1.Implement Comparable interface in the class(es) used as keys to TreeMap, or  
2.Supply an implementation of the Comparator that would do comparing outside the key class itself.

Natural ordering is the order provided by the Comparable interface .If somebody puts the key  that do not implement natural order then it will throw ClassCastException.

**Why do we need TreeMap when we have sortedMap ?**  
sortedMap is a interface and TreeMap is the class implementing it .As we know one can not create objects of the interface . Interface tells us which methods a sortedMap implementation should provide .TreeMap is such an implementation.

**Which data structure you will prefer in your code : HashMap or TreeMap ?**  
  
HashMap is faster while  TreeMap is sorted .Thus we choose them according to their advantage.  
  
If you do not want to sort the elements but just to insert and retrieve the elements then use HashMap .  
  
But if you want to maintain the  order of the elements then TreeMap should be preferred because the result is alphabetically sorted .While iterating HashMap there is no ordering of the elements ,on the other hand , TreeMap iterates in the natural key order.

**What happens if the TreeMap is concurrently modified while iterating the elements ?**  
  
The iterator *fails fast* and quickly if structurally modified at any time after the iterator is created (in any way except through the iterator's own remove method ).

**Which copy technique (deep or shallow ) is used by the TreeMap clone() method ?**

clone() method returns the shallow copy of the TreeMap instance . In shallow copy object B points to object A location in memory . In other words , both object A and B are sharing the same elements .The keys and values  themselves are not cloned .

**Why  java's  treemap does not allow an initial size ?**  
  
HashMap reallocates its internals as the new one gets inserted while TreeMap does not reallocate nodes on adding new ones. Thus , the size of the TreeMap  dynamically increases if needed , without shuffling the internals. So it is meaningless to set the initial size of the TreeMap .