1

Insertion order

-> HashMap does not maintains insertion order in java.

->Hashtable does not maintains insertion order in java.

->LinkedHashMap maintains insertion order in java.

->TreeMap is sorted by natural order of keys in java.

2

Performance

->HashMap is not synchronized, hence its operations are faster as compared to Hashtable.

->Hashtable is synchronized, hence its operations are slower as compared HashMap.

->If we are working not working in multithreading environment jdk recommends us to use HashMap.

->LinkedHashMap must be used only when we want to maintain insertion order. Time and space overhead is there because for maintaining order it internally uses Doubly Linked list.

->TreeMap must be used only when we want sorting based on natural order. Otherwise sorting operations cost performance. (Comparator is called for sorting purpose)

3

Null keys and values

->HashMap allows to store one null key and many null values i.e. many keys can have null value in java.

->Hashtable does not allow to store null key or null value.

Any attempt to store null key or value throws runtimeException (NullPointerException) in java.

->LinkedHashMap allows to store one null key and many null values i.e. any key can have null value in java.

->TreeMap does not allow to store null key but allow many null values.

Any attempt to store null key throws runtimeException (NullPointerException) in java.

4

Complexity of put, get and remove methods

->O(1)

->O(1)

->O(1)

overhead of updating Doubly Linked list for maintaining order it internally uses.

->O(log(n))