Property	Hash Map	Linked Hash Map	Tree Map
Time Complexity (Big O notation) Get, Put, Contains Key and Remove method	O(1)	O(1)	O(1)
Iteration Order	Random	Sorted according to either Insertion Order of Access Order (as specified during construction)	Sorted according to either natural Order of keys or comparator(as specified during construction)
Null Keys	allowed	allowed	Not allowed if keys uses Natural Ordering or Comparator does not support comparison on null Keys.
Interface	Map	Map	Map, Sorted Map and Navigable Map
Synchronization	None, use Collections. Synchronized Map()	None, use Collections. Synchronized Map()	None, use Collections. Synchronized Map()
Data Structure	List of buckets, if more than 8 entries in bucket then Java 8 will switch to balanced tree from linked list	Doubly Linked List of Buckets	Red-Black(a kind of self-balancing binary search tree) implementation of Binary Tree. This data structure offers O(log n) for insert, Delte and Search operations and O(n) space complexity.
Applications	General Purpose, fast retrieval, non-synchronized. Concurrent Hash Map can be used where concurrency is involved.	Can be used for LRU cache, other places where insertion or access order matters	Algorithms where Sorted or Navigable features are required. For example, find among the list of employees whose salary is next to given employee, Range Search, etc.
Requirements for Keys	Equals() and hash Code() needs to be overwritten.	Equals() and hash Code() needs to be overwritten.	Comparator needs to be supplied for key implementation, otherwise natural order will be used to sort the keys.