Collection	Implementation	Synchronization Information	Duplicate Value	Iterator	Difference
			SET INTERFACE		
HashSet	HashTable Set interface	Not Synchronized Can be synchronized by 'synchronizedSet'	No Duplicates Allowed	Iterators are used, 'fail-fast' Behaviour used only for Detecting bugs Directly propotional to capacity	Not ordered, allows null element,constant time performance for the Basic operations (as the hashfunction has datas as buckets), initial capacity is kept low for better Performance as iteration times depend upon the capacity
LinkedHashSet	HashTable LinkedList (doubly-linkedlist)	Not Synchronized Can be synchronized by 'synchronizedSet'	No Duplicates Allowed	Iterators are used, 'fail-fast' Behaviour used only for Detecting bugs capacity doesn't effect iteration	Insertion order,allows null element,Performance is little less than HashSet Due to additional maintanence of linkedlist , two capacity parameters Initial capacity and load factor doesn't affect the iteration like HashSet
TreeSet	NavigableSet of TreeMap	Not Synchronized Can be synchronized by 'synchronizedSet'	No Duplicates Allowed	Iterators are used, 'fail-fast' Behaviour used only for Detecting bugs	Natural Order(ascending order), log(n) time cost for the basic operations This order should be consistent with *equals()* which is used by set But the TreeSet method uses *compareTo()* for all its comaprison operation
			LIST INTERFACE		
ArrayList	Re-sizeable array	Not Synchronized Can be synchronized by 'synchronizedSet'	Duplicates are allowed	Iterators and listIterator Are used, 'fail-fast' Behaviour used only for Detecting bugs	Insertion order, allows the manipulation of array size, all the basic operations are performed in same time as this class implements RandomAccess interface, the constantfactor is low compared to LinkedList,the increase in Capacity will make the older list an Anonymous as it is garbage collected.Whenever the capacity is manipulated it Increases to it's half , ArrayList is index-based
LinkedList	Doubly-linked list Implementation of Deque and List	Not Synchronized Can be synchronized by 'synchronizedSet'	Duplicates are allowed	Iterators and listIterator Are used, 'fail-fast' Behaviour used only for Detecting bugs	Insertion order, implements the doubly linked list so it can traverse from beginning or from end, faster than ArrayList while inserting or deleting because ArrayList takes some time to move the elements to its adjacent Location, since it doesn't implement RandomAccess interface the time for manipulations won't be same.
Vector	Growable array of Objects.	Synchronized	Duplicates are allowed	Iterators and listIterator Are used, 'fail-fast' Behaviour used only for Detecting bugs	Depricated as of now, the vectors can shrink or grow as needed to accommodate items, vectors try to optimize Storage by maintaining capacity and capacityIncrement
			MAP INTERFACE		
HashMap	HashTable Map interface	Not Synchronized Can be synchronized by 'synchronizedSet'	Duplicate Value is permitted but not Key	Iterators are used, 'fail-fast' Behaviour used only for Detecting bugs Directly propotional to capacity	Unlike the List and Set interface Map interface is 'key-value pairs' and this doesn't come under "Collection Framework",Allows null values and null key, guarantees the order as hash table stores elements in buckets ,Constant-Time performance for the basic operations assuming the hash function disperses the elements properly Among the buckets, as the iteration depends upon capacity, It is adviced not to keep initial capacity Unlike the set and list interfaces map Initial capacity High or load factor low
HashTable	HashTable	Not Synchronized Can be synchronized by 'synchronizedSet'	Duplicate Value is permitted but not Key	Iterators are used, 'fail-fast' Behaviour used only for Detecting bugs Directly propotional to capacity	Depricated, null key or value is not allowed,initial capacity and load factor affects the performance,
LinkedHashMap	HashTable LinkedList	Not Synchronized Can be synchronized by 'synchronizedSet'	Duplicate Value is permitted but not Key	Iterators are used, 'fail-fast' Behaviour used only for Detecting bugs Directly propotional to capacity	Insertion order, which doesn't affect when the key is re-inserted into map Useful if a modul takes input as map and manipulates it and returns the resuls order determined by copy
TreeMap	NavigableMap	Not Synchronized Can be synchronized by 'synchronizedSet'	Duplicate Value is permitted but not Key	Iterators are used, 'fail-fast' Behaviour used only for Detecting bugs Directly propotional to capacity	Natural Order(ascending order), log(n) time cost for the basic operations This order should be consistent with *equals()* which is used by Map But the TreeMap method uses *compareTo()* for all its comaprison operation