Set [Interface]: LinkedHashSet

Definition

The LinkedHashSet is an ordered version of the HashSets that maintains a doubly-linked list across all elements. When the iteration order is needed to be maintained this class is used. The elements of the LinkedHashSet iterates in the order in which they were inserted

Features of LinkedHashSet

- Contains unique elements only like HashSet. It extends the HashSet class and implements the Set Interface.
- Maintains insertion order.
- Provides all optional set operation and permits null elements
- Is not Synchronized

Hierarchy of LinkedHashSets class

The LinkedHashSet class extends HashSet class which implements Set interface. The Set interface inherits Collection and iterable interfaces in hierarchical order.

Constructor summary

LinkedHashSet ()	Constructs a default HashSet
LinkedHashSet (Collection C)	Constructs a new HashSet containing the elements of the specified collection
LinkedHashSet (int size)	Constructs an empty set with the specified initial size.
LinkedHashSet (int capacity, float fillRatio)	Use to initialize both the capacity and fill ratio, also called the load capacity of the LinkedHashSet with the arguments mentioned in the parameter. When the number of elements exceeds the capacity of the hash set is multiplied with the fill ratio thus expanding the capacity of the LinkedHashSet.

Methods summary

void	**add**() Adds the specified element to this set if it is not already present.
void	**Clear()** Removes all of the elements from this set.
Object	**clone**() Returns a shallow copy of this HashSet instance: the elements themselves are not cloned.
Object	**contains** (Object o)

	Returns true if this set contains the specified element.
boolean	**isEmpty**() Returns true if this set contains no elements
Iterator<E>	**iterator**() Returns the iterator over the elements in this set
boolean	**remove(Object** o**)** Removes the specified element from this set if it is present
int	**size()** Returns the number of elements in this set (its cardinally)

When is its use recommended?

- Desire when the insertion order is important. LinkedHashSet maintains insertion order of elements.
- When does not require high performance and order O(1) for insertion, removal and retrieval are desire.
- When we can accept one null element.
- When the optimizing memory occupation is not required or need.