Set [Interface]: HashSet

Definition

Java HashSet implements the Set interface, backed by a hash table, is used to store a collection of unique elements. This class permit the null element. This class offer constant time performance for the basic operations (add, remove, contains, and size), assuming the hash function disperse the elements properly among the buckets.

Implementation is not synchronized. If multiple threads access a hash set concurrently, and at least one of the threads modifies the set, it must be synchronized externally.

Features of HashSet

- Implements Set Interface
- The underlying data structure for HasSet is HashTable
- As it implements the Set Interface, duplicate values are not allowed.
- Objects that you insert in HashSet are no guaranteed to be inserted in the same order. Objects are inserted based on their hash code.
- NULL elements are allowed in HashSet
- HashSet also implements Serializable and Cloneable interfaces.

Iterating over HashSet requires time proportional to the sum of the HashSet instance size plus the capacity of the backing HashMap instance, number of buckets. Its import not to set the initial capacity too high (or the load factor too low) if iteration performance is important.

- **Initial capacity:** The initial capacity means the number of buckets when hashtable is created. The number of buckets will be automatically increased if the current size gets full.
- **Load Factor:** It's a measure of how full the HashSet is allowed to get before its capacity is automatically increased. When the number of entries in the hash table exceeds the product of the load factor and the current capacity, the hash Table is rehashed.

Example:

If the internal capacity is 100 and the load factor is 0.75 then the number of buckets will automatically get increased when the table has 75 elements in it.

Constructor summary

HashSet ()	Constructs an empty set ; the backing HashMap instance has default initial capacity (16) and load factor (0.75)
HashSet (Collection < ? extends E > c)	Constructs a new set containing the elements of the specified collection
HashSet (int initalCapacity)	Constructs an empty set with the specified initial capacity; the backing HashMap instance has specified initial capacity and load factor (0.75)

HashSet (int initalCapacity, float loadFactor)	Constructs a new, empty set, the backing HashMap instance has specified initial capacity and specified load factor.

Methods summary

Modifier and Type	**Method and Description**
boolean	**add**(E e) Appends 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.
boolean	**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?

- HashSet will be used to remove undesirable duplicate elements in an enumerable collection.
- HashSet will be used if the objective is to enumerate and check for existence.
- HashSet will be used if want optimization of the data structure in set operations (Union, Difference, intersect).
- HashSet is generally used for set operations where LINQ can't provide all the methods required.