Java Collection Framework Linked HashSet

Java LinkedHashSet class is a linked list based implementation of Set interface. It is used to store unique elements. It uses hash table internally to store the elements. It implements Set interface and extends the HashSet class. Declaration of the class is given below.

Java LinkedHashSet class declaration

```
public class LinkedHashSet<E>extends HashSet<E>implements Set<E>,
Cloneable, Serializable
```

Copy

Important Points:

1. LinkedHashSet class extends HashSet class
2. It maintains a linked list of entries in the set.
3. It contains unique elements only
4. It allows to insert null value.
5. It stores elements in the order in which elements are inserted i.e it maintains the insertion order.
6. Java LinkedHashSet class is non synchronized.

LinkedHashSet Constructors

LinkedHashSet class has four constructors that can be used to create LinkedHashSet accordingly.

```
LinkedHashSet()  // This creates an empty LinkedHashSet

LinkedHashSet( Collection C )  // This creates a LinkedHashSet that is
initialized with the elements of the Collection C

LinkedHashSet( int capacity )  // This creates a LinkedHashSet that has
the specified initial capacity

LinkedHashSet( int initialCapacity, float loadFactor )
```

Copy

Example of LinkedHashSet class

In this example, we are creating a LinkedHashSet which is initially empty. Later on we will add items to this collection.

```
import java.util.*;

import java.util.*;

class Demo
```

```
{

  public static void main(String args[])

  {

    // Creating LinkedHashSet

    LinkedHashSet<String> hs = new LinkedHashSet<String>();

    // Displaying LinkedHashSet

    System.out.println(hs);

  }

}
```

Copy

```
[]
```

LinkedHashSet Method

It Inherits methods from HashSet class. So we can apply all the HashSet operations with LinkedHashSet. The given table contains the methods of HashSet.

Method	Description
add(E e)	It adds the specified element to this set if it is not already present.
clear()	It removes all of the elements from the set.
clone()	It returns a shallow copy of this LinkedHashSet instance: the elements themselves are not cloned.
contains(Object o)	It returns true if this set contains the specified element.
isEmpty()	It returns true if this set contains no elements.
iterator()	It returns an iterator over the elements in this set.
remove(Object o)	It removes the specified element from this set if it is present.

size()	It returns the number of elements in the set.
spliterator()	It creates a late-binding and fail-fast Spliterator over the elements in the set.

Add Elements to LinkedHashSet

In this example, we are creating a LinkedHashSet that store integer values. Since LinkedHashSet does not store duplicate elements, we tried to add a duplicate elements but the output contains only unique elements.

```
import java.util.*;

class Demo

{

  public static void main(String args[])

  {

    // Creating LinkedHashSet

    LinkedHashSet hs = new LinkedHashSet<>();

    // Adding elements

    hs.add(100);

    hs.add(200);

    hs.add(300);

    hs.add(100);

    // Displaying LinkedHashSet

    System.out.println(hs);

  }

}
```

Copy

```
[100, 200, 300]
```

Remove Elements from LinkedHashSet

To remove elements from the LinkedHashSet, we are using remove() method that remove the specified elements.

```
import java.util.*;

class Demo

{

  public static void main(String args[])

  {

    // Creating LinkedHashSet

    LinkedHashSet<Integer> hs = new LinkedHashSet<Integer>();

    // Adding elements

    hs.add(100);

    hs.add(200);

    hs.add(300);

    hs.add(100);

    // Displaying LinkedHashSet

    System.out.println(hs);

    // Removing elements

    hs.remove(300);

    System.out.println("After removing elements: \n"+hs);

  }

}
```

Copy
[100, 200, 300] After removing elements: [100, 200]

Traversing Elements of LinkedHashSet

Since LinkedHashSet is a collection then we can use loop to iterate its elements. In this example we are traversing elements using for loop. See the below example.

```
import java.util.*;

class Demo

{

  public static void main(String args[])
  {

    // Creating LinkedHashSet

    LinkedHashSet<Integer> hs = new LinkedHashSet<Integer>();

    // Adding elements

    hs.add(100);

    hs.add(200);

    hs.add(300);

    hs.add(100);

    // Traversing ArrayList

    for(Integer element : hs) {

      System.out.println(element);

    }

  }

}
```

Copy

```
100

200

300
```

Get size of LinkedHashSet

Sometimes we want to know number of elements an LinkedHashSet holds. In that case we use size() then returns size of LinkedHashSet which is equal to number of elements present in the list.

```
import java.util.*;

class Demo

{

  public static void main(String args[])

  {

    // Creating LinkedHashSet

    LinkedHashSet<Integer> hs = new LinkedHashSet<Integer>();

    // Adding elements
```

```java
    hs.add(100);

    hs.add(200);

    hs.add(300);

    hs.add(100);

    // Traversing ArrayList

    for(Integer element : hs) {

      System.out.println(element);

    }

    System.out.println("Total elements : "+hs.size());

  }

}
```

Copy

```
OUTPUT

100

200

300

Total elements : 3
```