LinkedHashSet is a class in Java that is part of the Collection Framework.

It is a subclass of HashSet and implements the Set interface.

👉 It stores unique elements (no duplicates) like HashSet, but preserves insertion order using a doubly linked list that runs through its entries.

🔷 Key Features of LinkedHashSet

1.Uniqueness

2.Does not allow duplicate elements.If you try to add a duplicate, it is ignored.

3.Maintains Insertion Order

4.Unlike HashSet (which is unordered), LinkedHashSet maintains the order in which elements are inserted.

5.Null Value Allows at most one null element.

6.Performance:Slightly slower than HashSet (because it maintains order).

Underlying Data Structure

Combination of HashTable + Doubly Linked List.

*/

```
package collectionframework;

import java.util.LinkedHashSet;

import java.util.Iterator;

public class LinkedHashSetExample {

   public static void main(String[] args) {

      // ✅ Create LinkedHashSet

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

      // ✅ 1. add(E e) → Add elements

      lhs.add("India");

      lhs.add("America");

      lhs.add("London");

      lhs.add("Japan");

      lhs.add("India"); // Duplicate → ignored

      System.out.println("Initial LinkedHashSet: " + lhs);

      // ✅ 2. size() → Get number of elements

      System.out.println("Size: " + lhs.size());
```

```
        // ✅ 3. contains(Object o) → Check if element exists
        System.out.println("Contains 'London'? " + lhs.contains("London"));
        System.out.println("Contains 'China'? " + lhs.contains("China"));
        // ✅ 4. remove(Object o) → Remove specific element
        lhs.remove("Japan");
        System.out.println("After removing 'Japan': " + lhs);
        // ✅ 5. isEmpty() → Check if empty
        System.out.println("Is set empty? " + lhs.isEmpty());
        // ✅ 6. Iterator → Iterate elements in insertion order
        System.out.println("Iterating with Iterator:");
        Iterator<String> it = lhs.iterator();
        while (it.hasNext()) {
            System.out.println(it.next());
        }

        // ✅ 7. clear() → Remove all elements
        lhs.clear();
        System.out.println("After clear(): " + lhs);
    }
}
```