CSCI 2120:
Software Design & Development II

UNIT 2: Collections Framework & Generics

Iterate Set

Overview

1. Introduction
2. Iterate Set using Iterator
    a. Example 1
3. ConcurrentModificationException
    a. Example 2
4. Remove Set element with Iterator
    a. Example 3
5. Iterate Set using Enhanced for-loop
    a. Example 4
6. Iterate Set using forEach method
    a. Example 5

Introduction

In this lecturel, we will learn **how to iterate Set in Java**. Set interface does not provide any get() method like the *List interface* to retrieve elements.

Therefore, the only way to take out elements from the set can be by using Iterator() method but this method does not return elements from set in any particular order.

Let’s see how many ways to iterate set in Java.

There are mainly three ways to iterate a set in Java. We can iterate set by using any one of the following ways. They are as follows:

- Using *Iterator*
- Using Enhanced for loop
- Using forEach()

Iterate Set using Iterator

Using Iterator method, we can traverse only in the forward direction from the first to last element. We cannot traverse over elements in the backward direction using the iterator() method.

Let’s take an example program where we will iterate elements of set using the iterator() method. Follow all the steps in the coding.

Example 1: Iterate Set elements using iterator() method

```
import java.util.HashSet;
import java.util.Iterator;
import java.util.Set;
public class IterateSet1 {
   public static void main(String[] args) {
       // Create a generic set object of type String.
       Set<String> s = new HashSet<String>(); // s.size() is 0.
       int size = s.size();
       System.out.println("Size before adding elements: " +size);

       // Adding elements to set.
       s.add("Orange");                       // s.size() is 1.
       s.add("Red");                          // s.size() is 2.
       s.add("Blue");                         // s.size() is 3.
       s.add("Yellow");                       // s.size() is 4.
       s.add("Green");                        // s.size() is 5.
       System.out.println("Elements in set");
       System.out.println(s);

       Iterator<String> itr = s.iterator();
       System.out.println("Iteration using Iterator method");
       while(itr.hasNext()) {
           Object str = itr.next();
           System.out.println(str);
       }
   }
}
```

Example 1: Iterate Set elements using iterator() method

Output:

```
Size before adding elements: 0
Elements in set
[Red, Blue, Yellow, Orange, Green]
Iteration using Iterator method
Red
Blue
Yellow
Orange
Green
```

Concurrent Modification Exception

What is Concurrent Modification Exception?:

During iteration, we cannot add an element to a set at an iterator position. If we try to add an element during iteration, JVM will throw an exception named **ConcurrentModificationException**.

Key point:

Iteration means repeating the same operation multiple times. Universal Iterator doesn't support adding

Example 2:

Let’s take an example program where we will try to add an element during the iteration and see which exception is thrown by JVM?

Example2: ConcurrentModificationException

```
import java.util.HashSet;
import java.util.Iterator;
import java.util.Set;

public class IterateSet2 {
   public static void main(String[] args) {
       Set<String> set= new HashSet<>();
       set.add("Banana");
       set.add("Orange");
       set.add("Apple");
       set.add("Mango");

       Iterator<String> itr = set.iterator();
       while(itr.hasNext()) {
           Object str = itr.next();
           System.out.println(str);
           set.add("Grapes");           // Add element during iteration, throws Exception.
       }
   }
}
```

Example2: ConcurrentModificationException

Output:

```
Apple
Exception in thread "main" java.util.ConcurrentModificationException Create breakpoint
    at java.base/java.util.HashMap$HashIterator.nextNode(HashMap.java:1597)
    at java.base/java.util.HashMap$KeyIterator.next(HashMap.java:1620)
    at IterateSet2.main(IterateSet2.java:15)
```

Key Point:

List can be iterated by using the ListIterator methods but Set cannot be iterated by using ListIterator because it doesn't implement the List interface.

Remove Set element with Iterator

Remove Set elements:

We can remove a Set element at an iterator position, just as we do with the list iterator.

Example 3:

Let’s make a program where we will remove a set element at an iterator position during iteration. Look at the following code to understand better.

Example 3: Remove Set element with Iterator

```
import java.util.HashSet;
import java.util.Iterator;
import java.util.Set;

public class IterateSet3 {
   public static void main(String[] args) {
       Set<String> set= new HashSet<>();
       set.add("Banana");
       set.add("Orange");
       set.add("Apple");
       set.add("Mango");

       Iterator<String> itr = set.iterator();
       while(itr.hasNext()) {
           Object str = itr.next();
           System.out.println(str);

           // Removing Mango element.
           if(str.equals("Mango")) {
               itr.remove();
           }
       }
       System.out.println(set);
   }
}
```

Example 3: Remove Set element with Iterator

Output:

```
Apple
Mango
Orange
Banana
[Apple, Orange, Banana]
```

Iterate Set using Enhanced For loop

Key point:

In this section, we iterate elements of Set using enhanced for loop.

Example 4:

Let’s make a program where we will iterate set elements using enhanced for loop.

Example 4: Iterate Set with enhanced for-loop

```
import java.util.HashSet;
import java.util.Set;

public class IterateSet4 {
   public static void main(String[] args) {
       // Create Set object of type Integer.
       Set<Integer> s = new HashSet<Integer>();
       // Adding even numbers between 10 to 30 as elements.
       for(int i = 10; i <= 30; i++) {
           if(i % 2 == 0) {
               s.add(i);
           }
       }
       System.out.println("Even numbers between 10 to 30");
       System.out.println(s);
       System.out.println("Iteration Using Enhanced For Loop");
       for( Integer it: s ) {
           System.out.println(it);
       }
   }
}
```

Example 4: Iterate Set with enhanced for-loop

Output:

```
Even numbers between 10 to 30
[16, 18, 20, 22, 24, 10, 26, 12, 28, 14, 30]
Iteration Using Enchanced For Loop
16
18
20
22
24
10
26
12
28
14
30
```

Iterate Set using forEach method

Key point:

This is the third way to iterate Set elements using for forEach() method. It is a very simple way to iterate elements. It was introduced in Java 1.8. The forEach method takes as a parameter either a lambda expression or a method reference.

More info: https://www.baeldung.com/foreach-java

Example 5:

Let's take an example program based on this method.

Example 5: Iterate Set with forEach method

```
import java.util.HashSet;
import java.util.Set;

public class IterateSet5 {
   public static void main(String[] args) {
       Set<Character> s = new HashSet<Character>();
       s.add('A');
       s.add('B');
       s.add('C');
       s.add('D');
       s.add('E');
       System.out.println(s);

       System.out.println("Iterating using forEach loop with method reference");
       s.forEach(System.out::println);

       System.out.println("Iterating using forEach loop with lambda expression");
       s.forEach( i -> System.out.println(i) );
   }
}
```

Example 5: Iterate Set with forEach method

Output:

```
[A, B, C, D, E]
Iterating using forEach loop with method reference
A
B
C
D
E
Iterating using forEach loop with lambda expression
A
B
C
D
E
```

Key point:

The forEach method is defined in the Iterable interface which all Collection interface objects implement. Thus this approach would also work for all objects in the Collection Framework.

END