Java 8 – forEach

BY CHAITANYA SINGH | FILED UNDER: JAVA 8 FEATURES

In Java 8, we have a newly introduced forEach method to iterate over collections and Streams in Java. In this guide, we will learn how to use forEach() and forEachOrdered() methods to loop a particular collection and stream.

Java 8 - forEach to iterate a Map

```
import java.util.Map;
import java.util.HashMap;
public class Example {
   public static void main(String[] args) {
      Map<Integer, String> hmap = new HashMap<Integer, String>();
      hmap.put(1, "Monkey");
      hmap.put(2, "Dog");
      hmap.put(3, "Cat");
      hmap.put(4, "Lion");
      hmap.put(5, "Tiger");
      hmap.put(6, "Bear");
      /* forEach to iterate and display each key and value pair
       * of HashMap.
       */
      hmap.forEach((key,value)->System.out.println(key+" - "+value));
      /* forEach to iterate a Map and display the value of a particular
       * key
       */
      hmap.forEach((key,value)->{
         if(key == 4){
            System.out.println("Value associated with key 4 is: "+value);
         }
      });
      /* forEach to iterate a Map and display the key associated with a
       * particular value
       */
      hmap.forEach((key,value)->{
         if("Cat".equals(value)){
            System.out.println("Key associated with Value Cat is: "+key);
         }
      });
   }
}
```

Java 8 - forEach to iterate a List

In this example, we are iterating an ArrayList using forEach() method. Inside forEach we are using a lambda expression to print each element of the list.

```
import java.util.List;
import java.util.ArrayList;
public class Example {
   public static void main(String[] args) {
```

```
      List<String> fruits = new ArrayList<String>();
      fruits.add("Apple");
      fruits.add("Orange");
      fruits.add("Banana");
      fruits.add("Pear");
      fruits.add("Mango");
      //lambda expression in forEach Method
      fruits.forEach(str->System.out.println(str));
   }
}
```

Output:

```
Apple
Orange
Banana
Pear
Mango
```

We can also use method reference in the forEach() method like this:

```
fruits.forEach(System.out::println);
```

Java 8 - forEach method to iterate a Stream

In this example we are iterating a Stream in Java using forEach() method.

```
import java.util.List;
import java.util.ArrayList;
public class Example {
   public static void main(String[] args) {
      List<String> names = new ArrayList<String>();
      names.add("Maggie");
      names.add("Michonne");
      names.add("Rick");
      names.add("Merle");
      names.add("Governor");
      names.stream() //creating stream
     .filter(f->f.startsWith("M")) //filtering names that starts with M
     .forEach(System.out::println); //displaying the stream using forEach
   }
}
```

Output:

```
Maggie
Michonne
Merle
```

Java - Stream forEachOrdered() Method Example

For sequential streams the order of elements is same as the order in the source, so the output would be same whether you use forEach or forEachOrdered. However when working with parallel streams, you would always want to use the forEachOrdered() method when the order matters to you, as this method guarantees that the order of elements would be same as

the source. Lets take an example to understand the difference between forEach() and forEachOrdered().

```
import java.util.List;
import java.util.ArrayList;
public class Example {
   public static void main(String[] args) {
      List<String> names = new ArrayList<String>();
      names.add("Maggie");
      names.add("Michonne");
      names.add("Rick");
      names.add("Merle");
      names.add("Governor");
      //forEach - the output would be in any order
      System.out.println("Print using forEach");
      names.stream()
     .filter(f->f.startsWith("M"))
     .parallel()
     .forEach(n->System.out.println(n));

     /* forEachOrdered - the output would always be in this order:
      * Maggie, Michonne, Merle
      */
     System.out.println("Print using forEachOrdered");
     names.stream()
     .filter(f->f.startsWith("M"))
     .parallel()
     .forEachOrdered(n->System.out.println(n));
   }
}
```

Output:

```
Print using forEach
Merle
Maggie
Michonne
Print using forEachOrdered
Maggie
Michonne
Merle
```