CSCI 2120:
Software Design & Development II

UNIT 2: Collections Framework & Generics

Iterate LinkedList

Overview

Introduction

In the last lecture, we have discussed LinkedList and its various methods with various example programs. In this lecture, we will learn **how to iterate LinkedList in Java**.

Before going on this topic, I will recommend that you first clear all the basics of LinkedList in Java.

There are five ways in which LinkedList can be iterated in Java. They are as follows:

1. For loop
2. Enhanced For loop
3. While loop
4. Iterator
5. ListIterator

Iterate LinkedList using `for`, Enhanced `for`, & `while` Loops

Let’s create an example program where we iterate LinkedList using for-loop, enhanced for-loop, and while-loop.

Example 1: LinkedList - `for`, enhanced `for`, `while`

```java
import java.util.LinkedList;
public class IterateLinkedListTester1 {
   public static void main(String[] args) {
       // Create a generic LinkedList object of String type.
       LinkedList<String> list = new LinkedList<String>(); // An empty list.

       // Adding elements in the list.
       list.add("Red");
       list.add("Yellow");
       list.add("Green");
       list.add("White");

       // Iterating using for loop.
       System.out.println("**For loop**");
       for(int i = 0; i < list.size(); i++) {
           Object element = list.get(i); // Return type of get() method is an Object.
           System.out.println(element);
       }
       // Iterating using Advanced for loop.
       System.out.println("**Advanced For loop**");
       for(String str: list) {
           System.out.println(str);
       }
       // Iterating using while loop.
       System.out.println("**While Loop**");
       int num = 0;
       while (list.size() > num) {
           System.out.println(list.get(num));
           num++;
       }
   }
}
```

Example 1: LinkedList - `for`, enhanced `for`, `while`

Output:

```
**For loop**
Red
Yellow
Green
White
**Advanced For loop**
Red
Yellow
Green
White
**While Loop**
Red
Yellow
Green
White
```

Iterate LinkedList using Iterator

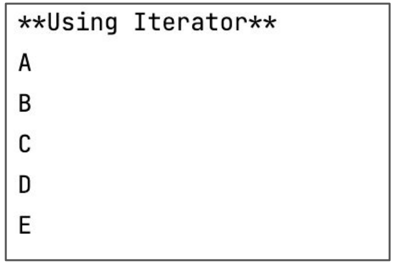
Let’s take an example program where we iterate elements of the LinkedList using the universal Iterator. Using Iterator, we can iterate the list in only the forward direction.

Example 2: LinkedList - Iterator

```java
import java.util.Iterator;
import java.util.LinkedList;
public class IterateLinkedListTester2 {
   public static void main(String[] args) {
       // Create a generic LinkedList object of Character type.
       LinkedList<Character> list = new LinkedList<Character>();

       // Adding elements in the list.
       list.add('A');
       list.add('B');
       list.add('C');
       list.add('D');
       list.add('E');

       // Iterating using Iterator.
       System.out.println("**Using Iterator**");
       Iterator<Character> itr = list.iterator();
       while(itr.hasNext()) {
           Object obj = itr.next();
           System.out.println(obj);
       }
   }
}
```

Example 2: LinkedList - Iterator

Output:



```
**Using Iterator**
A
B
C
D
E
```

Iterate LinkedList using ListIterator

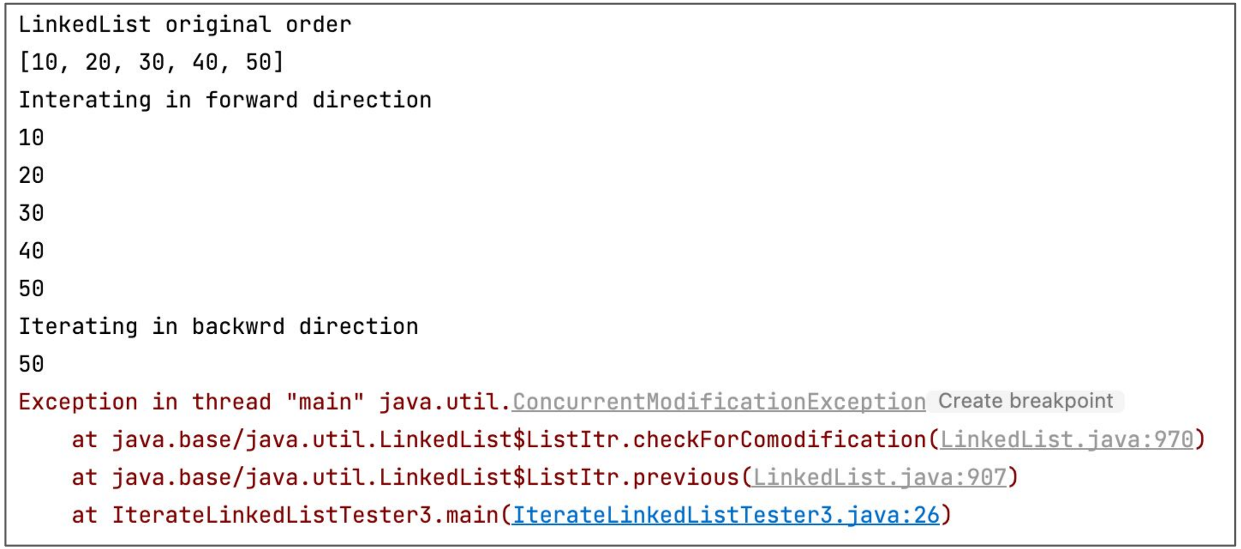
Let’s take an example program where we iterate elements of the LinkedList using ListIterator. We can iterate elements of the list in both forward and backward directions.

Example 3: LinkedList - ListIterator

```
import java.util.LinkedList;
import java.util.ListIterator;
public class IterateLinkedListTester3 {
   public static void main(String[] args) {
       // Create a generic LinkedList object of type Integer.
       LinkedList<Integer> list = new LinkedList<Integer>();

       // Adding elements in the list.
       list.add(10);
       list.add(20);
       list.add(30);
       list.add(40);
       list.add(50);

       System.out.println("LinkedList original order");
       System.out.println(list);

       ListIterator<Integer> litr = list.listIterator();
       System.out.println("Interating in forward direction");
       while(litr.hasNext()) {
           Object obj = litr.next();
           System.out.println(obj);
       }
       System.out.println("Iterating in backward direction");
       while(litr.hasPrevious()) {
           Object obj1 = litr.previous();
           System.out.println(obj1);
           list.add(60); // throws Concurrent Modification Exception because we cannot add or remove element in the LinkedList during iteration.
       }
       System.out.println(list);
   }
}
```

Example 3: LinkedList - ListIterator

Output:



```
LinkedList original order
[10, 20, 30, 40, 50]
Interating in forward direction
10
20
30
40
50
Iterating in backwrd direction
50
Exception in thread "main" java.util.ConcurrentModificationException
    at java.base/java.util.LinkedList$ListItr.checkForComodification(LinkedList.java:970)
    at java.base/java.util.LinkedList$ListItr.previous(LinkedList.java:907)
    at IterateLinkedListTester3.main(IterateLinkedListTester3.java:26)
```

END