Enumeration

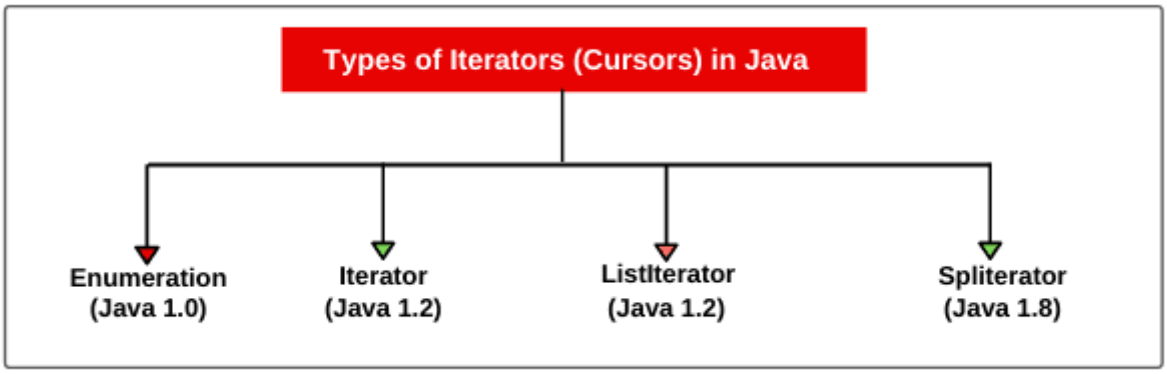
Iterators in Java

Iterators in Java are used to retrieve the elements one by one from a collection object. They
are also called cursors in Java. They allow us to iterate over a collection of objects and perform various operations on each object.

Types of Iterators in Java

There are four types of iterators or cursors available in Java. They are as follows:

- Enumeration
- Iterator
- ListIterator
- Spilterator



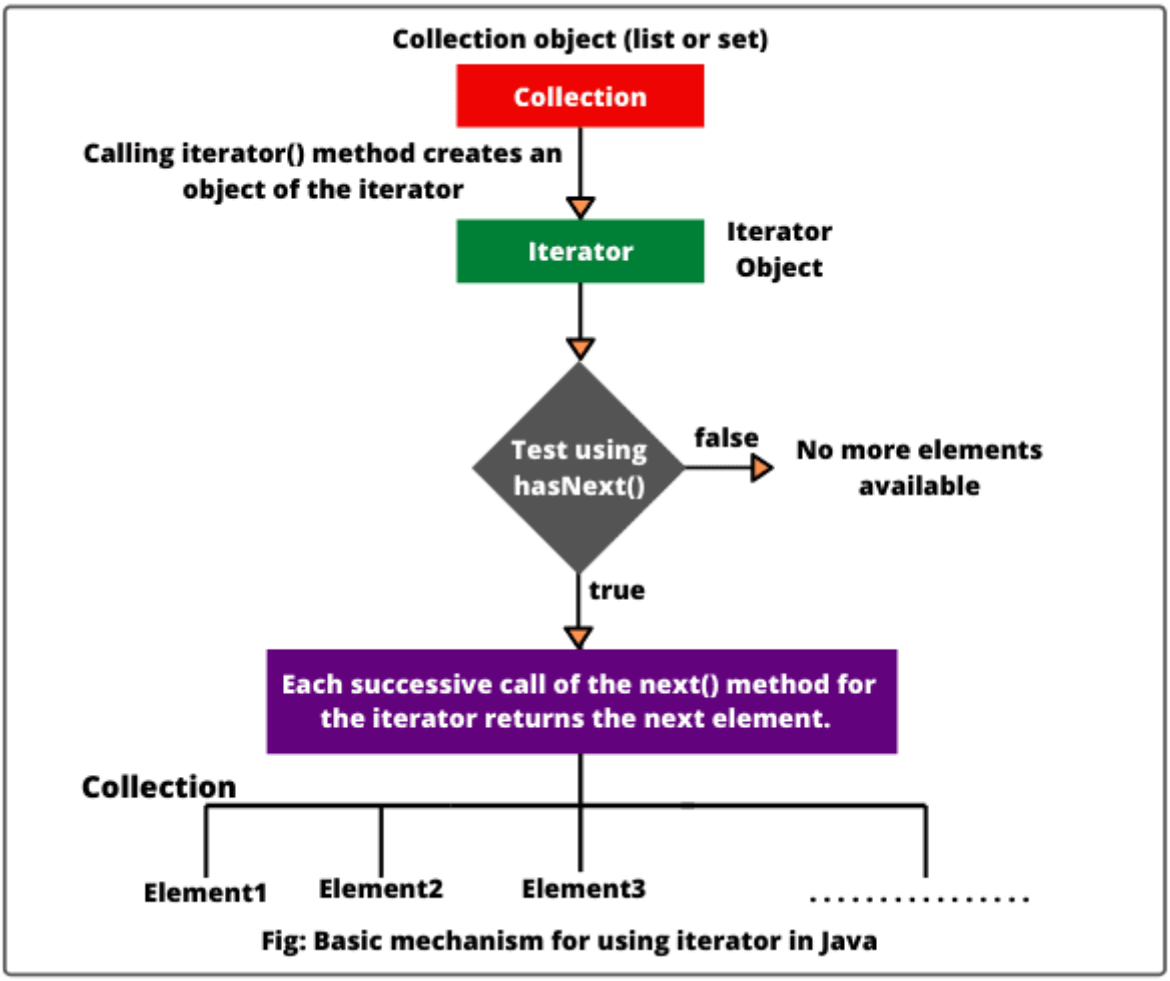


Fig: Basic mechanism for using iterator in Java

```
import java.util.Enumeration;
import java.util.Iterator;
import java.util.ListIterator;
import java.util.Vector;

public class InnerClassName {
public static void main(String[] args)
{
  Vector v = new Vector();
  Enumeration e = v.elements();
  Iterator itr = v.iterator();
  ListIterator litr = v.listIterator();

  System.out.println(e.getClass().getName());
  System.out.println(itr.getClass().getName());
  System.out.println(litr.getClass().getName());
 }
}
```

```
import java.util.Enumeration;
import java.util.Vector;
```

```
public class EnumerationTest
{
public static void main(String[] args)
{
// Create object of vector class without using generic.
   Vector v = new Vector();

// Add ten elements of integer type using addElement() method. For this we will use fo
r loop.
   for(int i = 0; i < = 10; i++)
   {
      v.addElement(i);
   }
   System.out.println(v);
// It will print all elements at a time like this [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

// Now we want to get elements one by one. So, we will require Enumeration concept.

// Create object of Enumeration by calling elements() method of vector class using obj
ect reference variable v.

// At the beginning, e (cursor) will point to index just before the first element in
 v.
   Enumeration e = v.elements();

// Checking the next element availability using reference variable e and while loop.
   while(e.hasMoreElements())
   {
 // Moving cursor to next element.
     Object o = e.nextElement();
  // Now type casting is required because the return type of nextElement() method is a
n object.
  // Therefore, it's compulsory to require type casting.
     Integer i = (Integer)o;

     System.out.println(i);
   }
  Enumeration en = v.elements();
  while(en.hasMoreElements())
  {
    Object o = en.nextElement();
    Integer it = (Integer)o;
 // Getting even elements one by one.
    if(it % 2 == 0)
    {
      System.out.println(it);
    }
  }
 }
}
```

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

```
{
// Creating an object of ArrayList of String type.
   ArrayList<String> al = new ArrayList<String>();

// Adding elements in the array list.
   al.add("A");
   al.add("B");
   al.add("C");
   al.add("D");
   al.add("E");
   al.add("F");

// Creating an iterator object of String type.
   Iterator<String> itr = al.iterator();

// Checking the availability of next element in the collection using reference variabl
e itr.
   while (itr.hasNext())
   {
  // Moving cursor to next element using reference variable itr.
     String str = itr.next();
     System.out.print(str + " ");
   }
  }
}
```

```
package iteratorsTest;
import java.util.ArrayList;
import java.util.Iterator;
public class IteratorTest {
public static void main(String[] args)
{
// Create an object of ArrayList of type Integer.
  ArrayList<Integer> al = new ArrayList<Integer>();
  for(int i = 0; i < = 8; i++)
  {
     al.add(i);
  }
  System.out.println(al); // It will print all elements at a time.

// Create the object of Iterator by calling iterator() method using reference variable
al.
// At the beginning, itr (cursor) will point to index just before the first element in
al.
   Iterator<Integer> itr = al.iterator();

// Checking the next element availability using reference variable itr.
   while(itr.hasNext())
   {
 // Moving cursor to next element using reference variable itr.
    Integer i = itr.next(); // Here, Type casting does not require due to using of gen
eric with Iterator.
    System.out.println(i);

// Removing odd elements.
```

```
   if(i % 2! = 0)
     itr.remove();
   }
   System.out.println(al);
 }
}
```

```
import java.util.ArrayList;
import java.util.Collection;
import java.util.Iterator;
public class IteratorTest {
public static void main(String[] args)
{
   Collection<String> collection = new ArrayList<>();
// Adding elements in the array list.
   collection.add("Red");
   collection.add("Green");
   collection.add("Black");
   collection.add("White");
   collection.add("Pink");

   Iterator<String> iterator = collection.iterator();
   while (iterator.hasNext())
   {
      System.out.print(iterator.next().toUpperCase() + " ");
   }
   System.out.println();
 }
}
```

ListIterator

ListIterator in Java is the most powerful iterator or cursor that was introduced in Java 1.2 version. It is a bi-directional cursor.

ListIterator is an interface (an extension of Iterator interface) in Java that is used to retrieve elements from a collection object in both forward and reverse directions.

```
import java.util.ArrayList;
import java.util.List;
import java.util.ListIterator;

public class ListIteratorTest {
public static void main(String[] args)
{
// Creating a list object.
  List<String> list = new ArrayList<>();
  list.add("A"); // Adding element A at index 0.
  list.add("B"); // Adding element B at index 1.
```

```
  list.add("C"); // Adding element C at index 2.

  System.out.println("List: " + list);

// Create the list iterator object by calling listIterator() method.
// | in the comments indicates the position of iterator.
   ListIterator<String> iterator = list.listIterator(); // |ABC
   System.out.println("List Iterator in Forward Direction:");

// Call hasNext() method to check elements are present in forward direction.
   boolean elementsPresent = iterator.hasNext(); // Return true.
   System.out.println(elementsPresent);

   int indexA = iterator.nextIndex();
   String elementA = iterator.next(); // A|BC
   System.out.println("IndexA = " +indexA +" "+ "Element: " +elementA);

   int indexB = iterator.nextIndex();
   String elementB = iterator.next(); // AB|C
   System.out.println("IndexB = " +indexB +" "+ "Element: " +elementB);

   int indexC = iterator.nextIndex();
   String elementC = iterator.next(); // ABC|
   System.out.println("IndexC = " +indexC +" "+ "Element: " +elementC);

   boolean elementsPresent2 = iterator.hasNext(); // Return false because the iterator
is at the end of the collection.
   System.out.println(elementsPresent2);
   String element = iterator.next(); // It will throw NoSuchElementException because t
here is not next element.
 }
}
```

```
Output:
       List: [A, B, C]
       List Iterator in Forward Direction:
       true
       IndexA = 0 Element: A
       IndexB = 1 Element: B
       IndexC = 2 Element: C
       false
       Exception in thread "main" java.util.NoSuchElementException
```

```
import java.util.LinkedList;
import java.util.List;
import java.util.ListIterator;

public class ListIteratorTest {
public static void main(String[] args)
{
// Creating a list object.
   List<String> list = new LinkedList<>();
```

```
// Adding elements in the list.
   list.add("A");
   list.add("B");
   list.add("C");

// Creating ListIterator object.
   ListIterator<String> listIterator = list.listIterator();

// Traversing elements in forwarding direction.
   System.out.println("Forward Direction Iteration:");

   while(listIterator.hasNext())
   {
      System.out.println(listIterator.next());
   }
// Traversing elements in the backward direction. The ListIterator cursor is at just a
fter the last element.
   System.out.println("Backward Direction Iteration:");

   while(listIterator.hasPrevious())
   {
     System.out.println(listIterator.previous());
   }
}}
```

```
Output:
       Forward Direction Iteration:
       A
       B
       C
       Backward Direction Iteration:
       C
       B
       A
```

```
import java.util.ArrayList;
import java.util.List;
import java.util.ListIterator;

public class ListIteratorTest {
public static void main(String[] args)
{
 List<String> list = new ArrayList<>();
  list.add("Red");
  list.add("Green");
  list.add("Yellow");
  list.add("Orange");
  list.add("Blue");
  list.add("White");
 System.out.println("List: " + list);

// Get the list iterator
   ListIterator<String> iterator = list.listIterator();
```

```
   System.out.println();

   System.out.println("List Iterator in Forward Direction:");
   while (iterator.hasNext())
   {
     int index = iterator.nextIndex();
     String element = iterator.next();
     System.out.println("Index = " + index + ", Element = " + element);
   }
   System.out.println();


   System.out.println("List Iterator in Backward Direction:");// Reuse the Java list i
terator to iterate from the end to the beginning.
   while (iterator.hasPrevious())
   {
     int index = iterator.previousIndex();
     String element = iterator.previous();
     System.out.println("Index = " + index + ", Element = " + element);
   }
  }
}
```

```
Output:
      List: [Red, Green, Yellow, Orange, Blue, White]

      List Iterator in Forward Direction:
      Index = 0, Element = Red
      Index = 1, Element = Green
      Index = 2, Element = Yellow
      Index = 3, Element = Orange
      Index = 4, Element = Blue
      Index = 5, Element = White

      List Iterator in Backward Direction:
      Index = 5, Element = White
      Index = 4, Element = Blue
      Index = 3, Element = Orange
      Index = 2, Element = Yellow
      Index = 1, Element = Green
      Index = 0, Element = Red
```

```
import java.util.ArrayList;
import java.util.List;
import java.util.ListIterator;
public class AddDemo {
public static void main(String[] args)
{
// Create an object of ArrayList of String type.
   List<String> list = new ArrayList<>();

// Adding elements to array list.
   list.add("A");
```

```
   list.add("B");
   list.add("C");
   list.add("D");
   System.out.println("List: "+list);

   System.out.println();
   ListIterator<String> listIterator = list.listIterator();

   System.out.println("Forward Direction Iteration:");
   while(listIterator.hasNext())
   {
      System.out.println(listIterator.next());
   }
   listIterator.add("E"); // Adds an element before the iterator position.
   System.out.println();
   System.out.println(list);
   System.out.println();

   System.out.println("Backward Direction Iteration:");
   while(listIterator.hasPrevious()){
      System.out.println(listIterator.previous());
   }
   listIterator.set("J"); // It will update the last element returned by previous.
   System.out.println();
   System.out.println(list);
 }
}
```

```
Output:
      List: [A, B, C, D]

     Forward Direction Iteration:
     A
     B
     C
     D

     [A, B, C, D, E]

     Backward Direction Iteration:
     E
     D
     C
     B
     A

     [J, B, C, D, E]
```

```
import java.util.ArrayList;
import java.util.List;
import java.util.ListIterator;

public class RemoveDemo {
```

```
public static void main(String[] args)
{
 List<String> list = new ArrayList<>();
  list.add("A");
  list.add("B");
  list.add("C");
  list.add("D");
  System.out.println("List: "+list);

  ListIterator<String> listIterator = list.listIterator();
  System.out.println("Forward Direction Iteration:");

  while(listIterator.hasNext()) {
     System.out.println(listIterator.next());
  }
  listIterator.remove(); // Removes the last element returned by next method.
  System.out.println("New List: " +list);
 }
}
```

```
Output:
      List: [A, B, C, D]
      Forward Direction Iteration:
      A
      B
      C
      D
      New List: [A, B, C]
```

```
import java.util.ArrayList;
import java.util.ListIterator;
public class ListIteratorTest {
 public static void main(String[] args)
 {
   ArrayList al = new ArrayList();
   al.add("Apple");
   al.add("Orange");
   al.add("Banana");
   al.add("Guava");
   al.add("Pineapple");
  System.out.println(al);

// Create the object of ListIterator using listIterator() method.
   ListIterator litr = al.listIterator();
   while(litr.hasNext())
   {
     Object o = litr.next();
     String str = (String)o; // Type casting.
     if(str.equals("Orange"))
     {
       litr.remove(); // It will remove orange from the list.
       System.out.println(al);
     }
```

```
      else if(str.equals("Guava"))
      {
        litr.add("Grapes"); // Adding Grapes after guava.
        System.out.println(al);
      }
      else if(str.equals("Pineapple"))
      {
        litr.set("Pears"); // Replacing Pineapple element.
        System.out.println(al);
      }
    }
  }
}
```

```
Output:
      [Apple, Orange, Banana, Guava, Pineapple]
      [Apple, Banana, Guava, Pineapple]
      [Apple, Banana, Guava, Grapes, Pineapple]
      [Apple, Banana, Guava, Grapes, Pears]
```

ArrayList

ArrayList

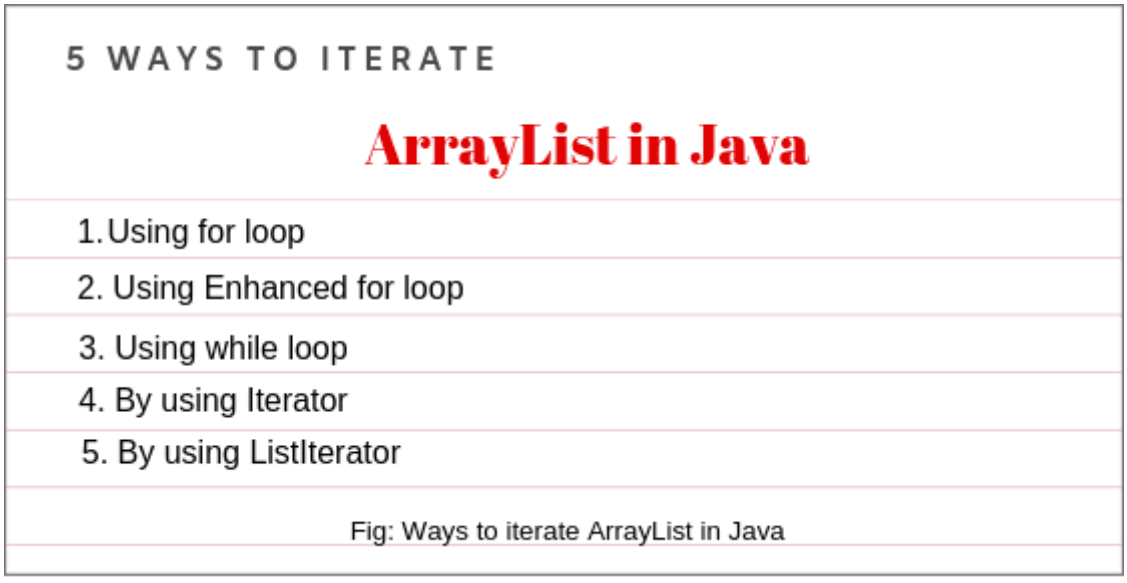


Fig: Ways to iterate ArrayList in Java

```
package iterateTest;
import java.util.ArrayList;
public class IterateArrayList
{
public static void main(String[] args)
{
// Create object of ArrayList of type String. In the list, we can add only String type
```

```
of elements.
   ArrayList<String> al = new ArrayList<String>();

// Call add() method to add the elements in the list using reference variable al.
   al.add("A"); // Adding element at index 0.
   al.add("B"); // Adding element at index 1.
   al.add("C"); // Adding element at index 2.
   al.add("D"); // Adding element at index 3.
   al.add("E"); // Adding element at index 4.

// Displaying original elements of the ArrayList.
   System.out.println(al); // It will display all elements of ArrayList at a time.

// Iterating ArrayList using for loop and call size() method to get the size of elemen
ts.
// Since the return type of size method is an integer.
// Therefore, we will store it using variable elementsize of type int.

   System.out.println("Using for loop");
   int elementsize = al.size();
   System.out.println("Size: " +elementsize);
   for(int i = 0; i < al.size(); i++)
   {
 // Call get() method to return elements on specified index after iterating.
    String getElement = al.get(i);
    System.out.println(getElement);
   }
   al.set(2, "G"); // It will replace current element at position 2 with element G.
   al.set(3, null); // adding null element at position 3.

// Iterating ArrayList using Enhance for loop.
   System.out.println("Using Enhance for loop");
   for(String element:al)
   {
      System.out.println(element);
    }
   }
 }
```

```
Output:
       [A, B, C, D, E]
       Using for loop
       Size: 5
       A B C D E
       Using Enhance for loop
       A B G null E
```

```
package iterateTest;
import java.util.ArrayList;
public class IterateUsingWhilelloop {
public static void main(String[] args)
{
// Create an object of ArrayList of type Integer.
```

```
   ArrayList<Integer> al = new ArrayList<Integer>();

// Adding elements in the array list.
   al.add(20);
   al.add(25);
   al.add(null);
   al.add(30);
   al.add(25);
  System.out.println(al);

// Iteration of ArrayList using while loop.
   System.out.println("Iteration using while loop");
   int i = 0;
   while(al.size() > i)
   {
     Integer itr = al.get(i);
     System.out.println(itr);
     i++;
   }
  }
 }
```

```
Output:
       [20, 25, null, 30, 25]
       Iteration using while loop
       20 25 null 30 25
```

```
package iterateTest;
import java.util.ArrayList;
import java.util.Iterator;
public class RemoveTest {
public static void main(String[] args)
{
// Creating an ArrayList object of type String.
   ArrayList<String> al = new ArrayList<String>();
   al.add("Apple");
   al.add("Mango");
   al.add("Banana");
   al.add("Guava");
   al.add("Pineapple");
   System.out.println(al); // It will print all elements at a time.

   System.out.println("Iteration using iterator concept.");
// Create an object of Iterator by calling iterator() method using reference variable
 al.
// At the beginning, itr(cursor) will point to index just before the first element in
 al.
   Iterator<String> itr = al.iterator();

// Checking the next element availability using reference variable itr.
   while(itr.hasNext())
   {
// Moving cursor to next element using reference variable itr.
```

```
     String str = itr.next();
     System.out.println(str);

// Removing the pineapple element.
   if(str.equals("Pineapple"))
   {
     itr.remove();
     System.out.println("After removing pineapple element");
     System.out.println(al);
    }
   }
 }
}
```

```
Output:
       [Apple, Mango, Banana, Guava, Pineapple]
       Iteration using iterator concept.
       Apple Mango Banana Guava Pineapple
       After removing pineapple element
       [Apple, Mango, Banana, Guava]
```

```
package iterateTest;
import java.util.ArrayList;
import java.util.Iterator;
public class AddingElementUsingIteration {
public static void main(String[] args)
{
 ArrayList<String> al = new ArrayList<String>();
  al.add("Lion");
  al.add("Tiger");
  al.add("Elephant");
  al.add("Bear");

 Iterator<String> itr = al.iterator();
 while(itr.hasNext())
 {
   System.out.println(itr.next());
// Adding element during iteration.
// Since the return type of add() method is boolean, we will store it using variable b
with data type boolean.
   boolean b = al.add("Leopard"); // Compile time error. It will throw ConcurrentModif
icationException.
   System.out.println(b);
  }
 }
}
```

```
Output:
     Exception in thread "main" java.util.ConcurrentModificationException
```

```
package iterateTest;
import java.util.ArrayList;
import java.util.ListIterator;
public class ArrayListUsingListIterator {
public static void main(String[] args)
{
 ArrayList al = new ArrayList();
  al.add("First");
  al.add("Second");
  al.add("Third");
  al.add("Fourth");
  al.add("Fifth");
  System.out.println(al);

// Iterating using ListIterator.
// Call listIterator() method to create object of ListIterator using reference variabl
e al.
   ListIterator litr = al.listIterator(); // Here, we are not using generic. Therefor
e, typecasting is required.

// Checking the next element availability in the forward direction using reference var
iable litr.
   System.out.println("Iteration in the forward direction");
   while(litr.hasNext())
   {
  // Moving cursor to next element in the forward direction using reference variable l
itr.
     Object o = litr.next();
     String str = (String)o; // Typecasting is required because the return type of nex
t() method is an Object.
     System.out.println(str);
   }
 // Checking the previous element in the backward direction using reference variable l
itr1.
    System.out.println("Iteration in the backward direction.");
    while(litr.hasPrevious())
    {
  // Moving cursor to the previous element in the backward direction.
     Object o = litr.previous();
     String str1 = (String)o; // Typecasting.
     System.out.println(str1);
  }
 }
}
```

```
Output:
       [First, Second, Third, Fourth, Fifth]
       Iteration in the forward direction
       First Second Third Fourth Fifth
       Iteration in the backward direction.
       Fifth Fourth Third Second First
```

```
package iterateTest;
import java.util.ArrayList;
import java.util.ListIterator;
public class AddRemoveTest {
public static void main(String[] args)
{
 ArrayList<String> al = new ArrayList<String>();
  al.add("One");
  al.add("Two");
  al.add("Three");
  al.add("Nine");
  al.add("Five");
  al.add("Seven");
 System.out.println(al);

 ListIterator<String> litr = al.listIterator();
 while(litr.hasNext())
 {
   String str = litr.next();
   if(str.equals("Nine"))
   {
     litr.remove();
     litr.add("Four");
     System.out.println(al);
   }
   else if(str.equals("Seven"))
   {
     litr.set("Six");
     System.out.println(al);
   }
  }
 }
}
```

```
Output:
       [One, Two, Three, Nine, Five, Seven]
       [One, Two, Three, Four, Five, Seven]
       [One, Two, Three, Four, Five, Six]
```

```
package iterateTest;
import java.util.ArrayList;
import java.util.ListIterator;
public class SpecificElementTest
{
public static void main(String[] args)
{
  ArrayList<Integer> al = new ArrayList<Integer>();
  for(int i = 0 ; i < = 9 ; i++)
  {
     al.add(i);
  }
 System.out.println(al);
```

```
 ListIterator<Integer> litr = al.listIterator(4); // Iterating through a specific elem
ent '4'.
 while(litr.hasNext())
 {
   Integer it = litr.next();
   System.out.println(it);
  }
 while(litr.hasPrevious())
 {
// This statement will throw ConcurrentModificationException because we can not add an
element in the ArrayList during Iteration.
   al.add(20);
   Integer it1 = litr.next();
   System.out.println(it1);
  }
 }
}
```

```
Output:
       [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
       4 5 6 7 8 9
       Exception in thread "main" java.util.ConcurrentModificationException
```