How to use Iterator in Java?

‘Iterator’ is an interface which belongs to collection framework. It allows us to traverse the collection, access the data element and remove the data elements of the collection.  
**java.util** package has **public interface Iterator** and contains three methods:

1. **boolean hasNext()**: It returns true if Iterator has more element to iterate.
2. **Object next()**: It returns the next element in the collection until the hasNext()method return true. This method throws ‘NoSuchElementException’ if there is no next element.
3. **void remove()**: It removes the current element in the collection. This method throws ‘IllegalStateException’ if this function is called before next( ) is invoked.
4. // Java code to illustrate the use of iterator
5. import java.io.\*;
6. import java.util.\*;
8. class Test {
9. public static void main(String[] args)
10. {
11. ArrayList<String> list = new ArrayList<String>();
13. list.add("A");
14. list.add("B");
15. list.add("C");
16. list.add("D");
17. list.add("E");
19. // Iterator to traverse the list
20. Iterator iterator = list.iterator();
22. System.out.println("List elements : ");
24. while (iterator.hasNext())
25. System.out.print(iterator.next() + " ");
27. System.out.println();
28. }
29. }

**ListIterator**

‘ListIterator’ in Java is an Iterator which allows users to traverse Collection in both direction. It contains the following methods:

1. **void add(Object object)**: It inserts object immediately before the element that is returned by the next( ) function.
2. **boolean hasNext( )**: It returns true if the list has a next element.
3. **boolean hasPrevious( )**: It returns true if the list has a previous element.
4. **Object next( )**: It returns the next element of the list. It throws ‘NoSuchElementException’ if there is no next element in the list.
5. **Object previous( )**: It returns the previous element of the list. It throws ‘NoSuchElementException’ if there is no previous element.
6. **void remove( )**: It removes the current element from the list. It throws ‘IllegalStateException’ if this function is called before next( ) or previous( ) is invoked.

// Java code to illustrate the use of ListIterator

import java.io.\*;

import java.util.\*;

class Test {

    public static void main(String[] args)

    {

        ArrayList<String> list = new ArrayList<String>();

        list.add("A");

        list.add("B");

        list.add("C");

        list.add("D");

        list.add("E");

        // ListIterator to traverse the list

        ListIterator iterator = list.listIterator();

        // Traversing the list in forward direction

        System.out.println("Displaying list elements in forward direction : ");

        while (iterator.hasNext())

            System.out.print(iterator.next() + " ");

System.out.println();

        // Traversing the list in backward direction

        System.out.println("Displaying list elements in backward direction : ");

        while (iterator.hasPrevious())

            System.out.print(iterator.previous() + " ");

        System.out.println();

    }

}

output:--

Displaying list elements in forward direction :

A B C D E

Displaying list elements in backward direction :

E D C B A