Iterator<E> interface.

Iterator<E> is used to traverse over a collection. Iterators are usually created by corresponding container class by invoking iterator().

To see how to iterate ArrayList<E> in 5 different ways click [here](http://data-structure-learning.blogspot.com/2015/05/java-collections-part-6iterating-over.html).

Iterator<E> provides 3 methods.

|  |  |
| --- | --- |
| **Method** | **Description** |
| **boolean** hasNext(); | Returns true if there are more elements. OR, returns true if next() returns element rather than throwing Exception. |
| E next(); | Returns the next element in iteration. It will throw NoSuchElementException if if there are no more elements. |
| **void** remove(); | removes the element returned by next(). For one next() call there is only one remove(). |

As of Java 8 Iterator<E> interface has changed

**void** remove();

to

**default** **void** remove() {

**throw** **new** UnsupportedOperationException("remove");

}

And one more method is added forEachRemaining(Consumer<? **super** E> action) – it performs mentioned action on each remaining elements till all elements are processed or throws Exception.

**IMPORTANT:**

1. Iterator in ArrayList class throws ConcurrentModificationException. It means that the object was modified in way which is not permissible. For example, while iterating through the ArrayList<E> using Iterator<E> we cannot remove element by using remove(index) or remove(object) method of ArrayList<E> class. Doing so results in ConcurrentModificationException.

Below is the example for it.

Iterator<String> iter = list.iterator();

**while** (iter.hasNext()) {

System.***out***.print(iter.next());

list.remove(2);

**if** (iter.hasNext()) {

System.***out***.print(", ");

}

}

Output for it is: Java is first element in list.

Java, Exception in thread "main" java.util.ConcurrentModificationException

at java.util.ArrayList$Itr.checkForComodification(Unknown Source)

at java.util.ArrayList$Itr.next(Unknown Source)

at iterator.IteratorImpl.main(IteratorImpl.java:20)

So make sure to use remove() provided in iterator.

1. **ConcurrentModificationException can also be thrown when multiple threads try to modify list which is not permissible.**

Below is the code that uses iterator to traverse in ArrayList<E>

**package** iterator;

**import** java.util.ArrayList;

**import** java.util.Iterator;

**import** java.util.List;

**public** **class** IteratorImpl {

**public** **static** **void** main(String[] args) {

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

list.add("Java");

list.add("C#");

list.add("MySQL");

list.add("MongoDB");

list.add("Oracle");

*iterateByIterator*(list);

}

**public** **static** **void** iterateByIterator(List<String> list) {

/\*\*

\* Gets the iterator.

\* Itr is private class in ArrayList<E> which implements Iterator<E>.

\* \*/

System.***out***.print("Using iterator() -> ");

Iterator<String> iter = list.iterator();

**while** (iter.hasNext()) {

System.***out***.print(iter.next());

**if** (iter.hasNext()) {

System.***out***.print(", ");

}

}

System.***out***.println();

/\*\*

\* Use of remove method

\* \*/

Iterator<String> iterator = list.iterator();

System.***out***.print("Using iterator() and remove() -> ");

**while** (iterator.hasNext()) {

String str=iterator.next();

/\*\*

\* if returned element is Java remove it.

\* We have used equalsIgnoreCase. So case does not matter.

\* \*/

**if**(str.equalsIgnoreCase("java")){

iterator.remove();

}

**else**{

System.***out***.print(str);

**if**(iterator.hasNext()){

System.***out***.print(", ");

}

}

}

System.***out***.println();

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

}

}

Output:

Using iterator() -> Java, C#, MySQL, MongoDB, Oracle

Using iterator() and remove() -> C#, MySQL, MongoDB, Oracle

List [C#, MySQL, MongoDB, Oracle]