**Iterator Class for The General List Type**

**Iterator**

|  |
| --- |
| An **iterator** is an object that can be used to retrieve the elements from a list, one after the other. [used in a loop]  The iterator method in the Collection interface **returns** an Iterator object. |
| The **AbstractCollection class** provides default method definitions for all of the collection methods except **iterator** and **size**. These two methods must be implemented in classes that extend AbstractCollection. |

**An Iterator Example**

|  |
| --- |
| **ListIterator** is an interface in the JCF that extends the Iterator interface, made to work with a list. While an Iterator object can only move forward, a ListIterator object moves forward or backward.  *ListIterator<String> it = myList.ListIterator(); while(it.hasNext()){System.out.println((it.next()); }*  To move backward through a list: *while(it.hasPrevious()){System.out.println((it.previous); }* |

**Implementations**

|  |
| --- |
| **Iterable** is java.lang package. **interface Iterable<E>**: *Iterator<E> iterator()*  **Iterator** is java.util package **–** instantiates iterator. **interface Iterator<E>**: *boolean hasNext(), E next(), void remove()* |

**Enhanced for Loop**

|  |
| --- |
| The Java compiler converts the enhanced for loop to a traditional loop that uses iterator. |

**Class GeneralList**

|  |
| --- |
| **class** **GeneralList<E> implements Iterable<E>**  {  <<< ... >>>  **public** E **get**(**int** index)  {  **if**(index < 0 || index >= elements)  **throw** **new** **IndexOutOfBoundsException**();  **return** list[index];  }  **public** E **remove**(**int** index)  {  **if**(index < 0 || index >= elements)  **throw** **new** **IndexOutOfBoundsException**();  E temp = list[index];  list[index] = **null**;  **for**(**int** index2 = index + 1; index2 < elements; index2++)  list[index2 - 1] = list[index2];  elements--;  **return** temp;  }    **public** **int** **size**()  {  **return** elements;  }    **public** Iterator<E> getIterator()  {  **return** **new** Iterator<E>(**this**);  }  } |

**The Iterator Class**

|  |
| --- |
| **class** **Iterator<E> implements Iterator<E>**  {  **GeneralList<E>** list;  **private** **int** previous;  **private** **boolean** canRemove;    **public** **Iterator**(**GeneralList<E>** aList)  {  list = aList;  previous = -1;  canRemove = **false**;  }  >>>>> THE REST OF THE METHODS GO HERE <<<<<  } |

**The hasNext() Method**

|  |
| --- |
| **public** **boolean hasNext**()  {  **return** ((previous + 1) < list.**size**());  } |

**The next() Method**

|  |
| --- |
| **public** **E next**()  {  **if**(!**hasNext**())  **throw** **new** **NoSuchElementException()**; // import java.util.NoSuchElementException;  **else**  {  previous++;  canRemove = **true**;  **return** list.get(previous);  }  } |

**The remove() Method**

|  |
| --- |
| **public** **void remove**()  {  **if**(!canRemove)  **throw** **new** IllegalStateException();  **else**  {  list.remove(previous);  previous--;  canRemove = **false**;  }  } |

**The main() Method**

|  |
| --- |
| **public** **class** I**teratorDemo**  {  **public** **static** **void** main(String[] args)  {  GeneralList<String> myList = **new** GeneralList<String>();  myList.**add**("One"); myList.**add**("Two");    **Iterator<String>** iterator1 = **myList.getIterator();**  **Iterator<String>** iterator2 = **myList.getIterator();**    **while**(iterator1.hasNext())  {  **if**(iterator1.**next**().**equals**("One"))  iterator1.**remove**();  }  **while**(iterator2.**hasNext**())  System.*out*.print(iterator2.next() + " ");  }  } |