001/\*  
002 \* Licensed to the Apache Software Foundation (ASF) under one or more  
003 \* contributor license agreements. See the NOTICE file distributed with  
004 \* this work for additional information regarding copyright ownership.  
005 \* The ASF licenses this file to You under the Apache License, Version 2.0  
006 \* (the "License"); you may not use this file except in compliance with  
007 \* the License. You may obtain a copy of the License at  
008 \*  
009 \* http://www.apache.org/licenses/LICENSE-2.0  
010 \*  
011 \* Unless required by applicable law or agreed to in writing, software  
012 \* distributed under the License is distributed on an "AS IS" BASIS,  
013 \* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
014 \* See the License for the specific language governing permissions and  
015 \* limitations under the License.  
016 \*/  
017package org.apache.commons.collections4.iterators;  
018  
019import java.util.ListIterator;  
020  
021import org.apache.commons.collections4.Unmodifiable;  
022  
023/\*\*  
024 \* Decorates a list iterator such that it cannot be modified.  
025 \* <p>  
026 \* Attempts to modify it will result in an UnsupportedOperationException.  
027 \* </p>  
028 \*  
029 \* @since 3.0  
030 \*/  
031public final class UnmodifiableListIterator<E> implements ListIterator<E>, Unmodifiable {  
032  
033 /\*\* The iterator being decorated \*/  
034 private final ListIterator<? extends E> iterator;  
035  
036 //-----------------------------------------------------------------------  
037 /\*\*  
038 \* Decorates the specified iterator such that it cannot be modified.  
039 \*  
040 \* @param <E> the element type  
041 \* @param iterator the iterator to decorate  
042 \* @return a new unmodifiable list iterator  
043 \* @throws NullPointerException if the iterator is null  
044 \*/  
045 public static <E> ListIterator<E> umodifiableListIterator(final ListIterator<? extends E> iterator) {  
046 if (iterator == null) {  
047 throw new NullPointerException("ListIterator must not be null");  
048 }  
049 if (iterator instanceof Unmodifiable) {  
050 @SuppressWarnings("unchecked") // safe to upcast  
051 final ListIterator<E> tmpIterator = (ListIterator<E>) iterator;  
052 return tmpIterator;  
053 }  
054 return new UnmodifiableListIterator<>(iterator);  
055 }  
056  
057 //-----------------------------------------------------------------------  
058 /\*\*  
059 \* Constructor.  
060 \*  
061 \* @param iterator the iterator to decorate  
062 \*/  
063 private UnmodifiableListIterator(final ListIterator<? extends E> iterator) {  
064 super();  
065 this.iterator = iterator;  
066 }  
067  
068 //-----------------------------------------------------------------------  
069 @Override  
070 public boolean hasNext() {  
071 return iterator.hasNext();  
072 }  
073  
074 @Override  
075 public E next() {  
076 return iterator.next();  
077 }  
078  
079 @Override  
080 public int nextIndex() {  
081 return iterator.nextIndex();  
082 }  
083  
084 @Override  
085 public boolean hasPrevious() {  
086 return iterator.hasPrevious();  
087 }  
088  
089 @Override  
090 public E previous() {  
091 return iterator.previous();  
092 }  
093  
094 @Override  
095 public int previousIndex() {  
096 return iterator.previousIndex();  
097 }  
098  
099 @Override  
100 public void remove() {  
101 throw new UnsupportedOperationException("remove() is not supported");  
102 }  
103  
104 @Override  
105 public void set(final E obj) {  
106 throw new UnsupportedOperationException("set() is not supported");  
107 }  
108  
109 @Override  
110 public void add(final E obj) {  
111 throw new UnsupportedOperationException("add() is not supported");  
112 }  
113  
114}