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.list;  
018  
019import java.util.Collection;  
020import java.util.Iterator;  
021import java.util.List;  
022import java.util.ListIterator;  
023import java.util.function.Predicate;  
024  
025import org.apache.commons.collections4.Unmodifiable;  
026import org.apache.commons.collections4.iterators.UnmodifiableIterator;  
027import org.apache.commons.collections4.iterators.UnmodifiableListIterator;  
028  
029/\*\*  
030 \* Decorates another <code>List</code> to ensure it can't be altered.  
031 \* <p>  
032 \* This class is Serializable from Commons Collections 3.1.  
033 \* </p>  
034 \* <p>  
035 \* Attempts to modify it will result in an UnsupportedOperationException.  
036 \* </p>  
037 \*  
038 \* @since 3.0  
039 \*/  
040public final class UnmodifiableList<E>  
041 extends AbstractSerializableListDecorator<E>  
042 implements Unmodifiable {  
043  
044 /\*\* Serialization version \*/  
045 private static final long serialVersionUID = 6595182819922443652L;  
046  
047 /\*\*  
048 \* Factory method to create an unmodifiable list.  
049 \*  
050 \* @param <E> the type of the elements in the list  
051 \* @param list the list to decorate, must not be null  
052 \* @return a new unmodifiable list  
053 \* @throws NullPointerException if list is null  
054 \* @since 4.0  
055 \*/  
056 public static <E> List<E> unmodifiableList(final List<? extends E> list) {  
057 if (list instanceof Unmodifiable) {  
058 @SuppressWarnings("unchecked") // safe to upcast  
059 final List<E> tmpList = (List<E>) list;  
060 return tmpList;  
061 }  
062 return new UnmodifiableList<>(list);  
063 }  
064  
065 //-----------------------------------------------------------------------  
066 /\*\*  
067 \* Constructor that wraps (not copies).  
068 \*  
069 \* @param list the list to decorate, must not be null  
070 \* @throws NullPointerException if list is null  
071 \*/  
072 @SuppressWarnings("unchecked") // safe to upcast  
073 public UnmodifiableList(final List<? extends E> list) {  
074 super((List<E>) list);  
075 }  
076  
077 //-----------------------------------------------------------------------  
078 @Override  
079 public Iterator<E> iterator() {  
080 return UnmodifiableIterator.unmodifiableIterator(decorated().iterator());  
081 }  
082  
083 @Override  
084 public boolean add(final Object object) {  
085 throw new UnsupportedOperationException();  
086 }  
087  
088 @Override  
089 public boolean addAll(final Collection<? extends E> coll) {  
090 throw new UnsupportedOperationException();  
091 }  
092  
093 @Override  
094 public void clear() {  
095 throw new UnsupportedOperationException();  
096 }  
097  
098 @Override  
099 public boolean remove(final Object object) {  
100 throw new UnsupportedOperationException();  
101 }  
102  
103 /\*\*  
104 \* @since 4.4  
105 \*/  
106 @Override  
107 public boolean removeIf(Predicate<? super E> filter) {  
108 throw new UnsupportedOperationException();  
109 }  
110  
111 @Override  
112 public boolean removeAll(final Collection<?> coll) {  
113 throw new UnsupportedOperationException();  
114 }  
115  
116 @Override  
117 public boolean retainAll(final Collection<?> coll) {  
118 throw new UnsupportedOperationException();  
119 }  
120  
121 //-----------------------------------------------------------------------  
122 @Override  
123 public ListIterator<E> listIterator() {  
124 return UnmodifiableListIterator.umodifiableListIterator(decorated().listIterator());  
125 }  
126  
127 @Override  
128 public ListIterator<E> listIterator(final int index) {  
129 return UnmodifiableListIterator.umodifiableListIterator(decorated().listIterator(index));  
130 }  
131  
132 @Override  
133 public void add(final int index, final E object) {  
134 throw new UnsupportedOperationException();  
135 }  
136  
137 @Override  
138 public boolean addAll(final int index, final Collection<? extends E> coll) {  
139 throw new UnsupportedOperationException();  
140 }  
141  
142 @Override  
143 public E remove(final int index) {  
144 throw new UnsupportedOperationException();  
145 }  
146  
147 @Override  
148 public E set(final int index, final E object) {  
149 throw new UnsupportedOperationException();  
150 }  
151  
152 @Override  
153 public List<E> subList(final int fromIndex, final int toIndex) {  
154 final List<E> sub = decorated().subList(fromIndex, toIndex);  
155 return new UnmodifiableList<>(sub);  
156 }  
157  
158}