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.collection;  
018  
019import java.util.Collection;  
020import java.util.Iterator;  
021import java.util.function.Predicate;  
022  
023import org.apache.commons.collections4.Unmodifiable;  
024import org.apache.commons.collections4.iterators.UnmodifiableIterator;  
025  
026/\*\*  
027 \* Decorates another {@link Collection} to ensure it can't be altered.  
028 \* <p>  
029 \* This class is Serializable from Commons Collections 3.1.  
030 \* </p>  
031 \* <p>  
032 \* Attempts to modify it will result in an UnsupportedOperationException.  
033 \* </p>  
034 \*  
035 \* @param <E> the type of the elements in the collection  
036 \* @since 3.0  
037 \*/  
038public final class UnmodifiableCollection<E>  
039 extends AbstractCollectionDecorator<E>  
040 implements Unmodifiable {  
041  
042 /\*\* Serialization version \*/  
043 private static final long serialVersionUID = -239892006883819945L;  
044  
045 /\*\*  
046 \* Factory method to create an unmodifiable collection.  
047 \* <p>  
048 \* If the collection passed in is already unmodifiable, it is returned.  
049 \*  
050 \* @param <T> the type of the elements in the collection  
051 \* @param coll the collection to decorate, must not be null  
052 \* @return an unmodifiable collection  
053 \* @throws NullPointerException if collection is null  
054 \* @since 4.0  
055 \*/  
056 public static <T> Collection<T> unmodifiableCollection(final Collection<? extends T> coll) {  
057 if (coll instanceof Unmodifiable) {  
058 @SuppressWarnings("unchecked") // safe to upcast  
059 final Collection<T> tmpColl = (Collection<T>) coll;  
060 return tmpColl;  
061 }  
062 return new UnmodifiableCollection<>(coll);  
063 }  
064  
065 //-----------------------------------------------------------------------  
066 /\*\*  
067 \* Constructor that wraps (not copies).  
068 \*  
069 \* @param coll the collection to decorate, must not be null  
070 \* @throws NullPointerException if collection is null  
071 \*/  
072 @SuppressWarnings("unchecked") // safe to upcast  
073 private UnmodifiableCollection(final Collection<? extends E> coll) {  
074 super((Collection<E>) coll);  
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 E 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(final 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}