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.set;  
018  
019import java.util.Collection;  
020import java.util.Iterator;  
021import java.util.Set;  
022import java.util.function.Predicate;  
023  
024import org.apache.commons.collections4.Unmodifiable;  
025import org.apache.commons.collections4.iterators.UnmodifiableIterator;  
026  
027/\*\*  
028 \* Decorates another <code>Set</code> to ensure it can't be altered.  
029 \* <p>  
030 \* This class is Serializable from Commons Collections 3.1.  
031 \* </p>  
032 \* <p>  
033 \* Attempts to modify it will result in an UnsupportedOperationException.  
034 \* </p>  
035 \*  
036 \* @param <E> the type of the elements in this set  
037 \* @since 3.0  
038 \*/  
039public final class UnmodifiableSet<E>  
040 extends AbstractSerializableSetDecorator<E>  
041 implements Unmodifiable {  
042  
043 /\*\* Serialization version \*/  
044 private static final long serialVersionUID = 6499119872185240161L;  
045  
046 /\*\*  
047 \* Factory method to create an unmodifiable set.  
048 \*  
049 \* @param <E> the element type  
050 \* @param set the set to decorate, must not be null  
051 \* @return a new unmodifiable set  
052 \* @throws NullPointerException if set is null  
053 \* @since 4.0  
054 \*/  
055 public static <E> Set<E> unmodifiableSet(final Set<? extends E> set) {  
056 if (set instanceof Unmodifiable) {  
057 @SuppressWarnings("unchecked") // safe to upcast  
058 final Set<E> tmpSet = (Set<E>) set;  
059 return tmpSet;  
060 }  
061 return new UnmodifiableSet<>(set);  
062 }  
063  
064 //-----------------------------------------------------------------------  
065 /\*\*  
066 \* Constructor that wraps (not copies).  
067 \*  
068 \* @param set the set to decorate, must not be null  
069 \* @throws NullPointerException if set is null  
070 \*/  
071 @SuppressWarnings("unchecked") // safe to upcast  
072 private UnmodifiableSet(final Set<? extends E> set) {  
073 super((Set<E>) set);  
074 }  
075  
076 //-----------------------------------------------------------------------  
077 @Override  
078 public Iterator<E> iterator() {  
079 return UnmodifiableIterator.unmodifiableIterator(decorated().iterator());  
080 }  
081  
082 @Override  
083 public boolean add(final E object) {  
084 throw new UnsupportedOperationException();  
085 }  
086  
087 @Override  
088 public boolean addAll(final Collection<? extends E> coll) {  
089 throw new UnsupportedOperationException();  
090 }  
091  
092 @Override  
093 public void clear() {  
094 throw new UnsupportedOperationException();  
095 }  
096  
097 @Override  
098 public boolean remove(final Object object) {  
099 throw new UnsupportedOperationException();  
100 }  
101  
102 /\*\*  
103 \* @since 4.4  
104 \*/  
105 @Override  
106 public boolean removeIf(Predicate<? super E> filter) {  
107 throw new UnsupportedOperationException();  
108 }  
109  
110 @Override  
111 public boolean removeAll(final Collection<?> coll) {  
112 throw new UnsupportedOperationException();  
113 }  
114  
115 @Override  
116 public boolean retainAll(final Collection<?> coll) {  
117 throw new UnsupportedOperationException();  
118 }  
119  
120}