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.bidimap;  
018  
019import java.util.Map;  
020import java.util.Set;  
021  
022import org.apache.commons.collections4.BidiMap;  
023import org.apache.commons.collections4.MapIterator;  
024import org.apache.commons.collections4.Unmodifiable;  
025import org.apache.commons.collections4.iterators.UnmodifiableMapIterator;  
026import org.apache.commons.collections4.map.UnmodifiableEntrySet;  
027import org.apache.commons.collections4.set.UnmodifiableSet;  
028  
029/\*\*  
030 \* Decorates another {@link BidiMap} to ensure it can't be altered.  
031 \* <p>  
032 \* Attempts to modify it will result in an UnsupportedOperationException.  
033 \* </p>  
034 \*  
035 \* @param <K> the type of the keys in this map  
036 \* @param <V> the type of the values in this map  
037 \* @since 3.0  
038 \*/  
039public final class UnmodifiableBidiMap<K, V>  
040 extends AbstractBidiMapDecorator<K, V> implements Unmodifiable {  
041  
042 /\*\* The inverse unmodifiable map \*/  
043 private UnmodifiableBidiMap<V, K> inverse;  
044  
045 /\*\*  
046 \* Factory method to create an unmodifiable map.  
047 \* <p>  
048 \* If the map passed in is already unmodifiable, it is returned.  
049 \*  
050 \* @param <K> the key type  
051 \* @param <V> the value type  
052 \* @param map the map to decorate, must not be null  
053 \* @return an unmodifiable BidiMap  
054 \* @throws NullPointerException if map is null  
055 \* @since 4.0  
056 \*/  
057 public static <K, V> BidiMap<K, V> unmodifiableBidiMap(final BidiMap<? extends K, ? extends V> map) {  
058 if (map instanceof Unmodifiable) {  
059 @SuppressWarnings("unchecked") // safe to upcast  
060 final BidiMap<K, V> tmpMap = (BidiMap<K, V>) map;  
061 return tmpMap;  
062 }  
063 return new UnmodifiableBidiMap<>(map);  
064 }  
065  
066 //-----------------------------------------------------------------------  
067 /\*\*  
068 \* Constructor that wraps (not copies).  
069 \*  
070 \* @param map the map to decorate, must not be null  
071 \* @throws NullPointerException if map is null  
072 \*/  
073 @SuppressWarnings("unchecked") // safe to upcast  
074 private UnmodifiableBidiMap(final BidiMap<? extends K, ? extends V> map) {  
075 super((BidiMap<K, V>) map);  
076 }  
077  
078 //-----------------------------------------------------------------------  
079 @Override  
080 public void clear() {  
081 throw new UnsupportedOperationException();  
082 }  
083  
084 @Override  
085 public V put(final K key, final V value) {  
086 throw new UnsupportedOperationException();  
087 }  
088  
089 @Override  
090 public void putAll(final Map<? extends K, ? extends V> mapToCopy) {  
091 throw new UnsupportedOperationException();  
092 }  
093  
094 @Override  
095 public V remove(final Object key) {  
096 throw new UnsupportedOperationException();  
097 }  
098  
099 @Override  
100 public Set<Map.Entry<K, V>> entrySet() {  
101 final Set<Map.Entry<K, V>> set = super.entrySet();  
102 return UnmodifiableEntrySet.unmodifiableEntrySet(set);  
103 }  
104  
105 @Override  
106 public Set<K> keySet() {  
107 final Set<K> set = super.keySet();  
108 return UnmodifiableSet.unmodifiableSet(set);  
109 }  
110  
111 @Override  
112 public Set<V> values() {  
113 final Set<V> set = super.values();  
114 return UnmodifiableSet.unmodifiableSet(set);  
115 }  
116  
117 //-----------------------------------------------------------------------  
118 @Override  
119 public K removeValue(final Object value) {  
120 throw new UnsupportedOperationException();  
121 }  
122  
123 @Override  
124 public MapIterator<K, V> mapIterator() {  
125 final MapIterator<K, V> it = decorated().mapIterator();  
126 return UnmodifiableMapIterator.unmodifiableMapIterator(it);  
127 }  
128  
129 @Override  
130 public synchronized BidiMap<V, K> inverseBidiMap() {  
131 if (inverse == null) {  
132 inverse = new UnmodifiableBidiMap<>(decorated().inverseBidiMap());  
133 inverse.inverse = this;  
134 }  
135 return inverse;  
136 }  
137  
138}