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;  
021import java.util.SortedMap;  
022  
023import org.apache.commons.collections4.OrderedMapIterator;  
024import org.apache.commons.collections4.SortedBidiMap;  
025import org.apache.commons.collections4.Unmodifiable;  
026import org.apache.commons.collections4.iterators.UnmodifiableOrderedMapIterator;  
027import org.apache.commons.collections4.map.UnmodifiableEntrySet;  
028import org.apache.commons.collections4.map.UnmodifiableSortedMap;  
029import org.apache.commons.collections4.set.UnmodifiableSet;  
030  
031/\*\*  
032 \* Decorates another {@link SortedBidiMap} to ensure it can't be altered.  
033 \* <p>  
034 \* Attempts to modify it will result in an {@link UnsupportedOperationException}.  
035 \* </p>  
036 \*  
037 \* @param <K> the type of the keys in this map  
038 \* @param <V> the type of the values in this map  
039 \* @since 3.0  
040 \*/  
041public final class UnmodifiableSortedBidiMap<K, V>  
042 extends AbstractSortedBidiMapDecorator<K, V> implements Unmodifiable {  
043  
044 /\*\* The inverse unmodifiable map \*/  
045 private UnmodifiableSortedBidiMap<V, K> inverse;  
046  
047 /\*\*  
048 \* Factory method to create an unmodifiable map.  
049 \* <p>  
050 \* If the map passed in is already unmodifiable, it is returned.  
051 \*  
052 \* @param <K> the key type  
053 \* @param <V> the value type  
054 \* @param map the map to decorate, must not be null  
055 \* @return an unmodifiable SortedBidiMap  
056 \* @throws NullPointerException if map is null  
057 \* @since 4.0  
058 \*/  
059 public static <K, V> SortedBidiMap<K, V> unmodifiableSortedBidiMap(final SortedBidiMap<K, ? extends V> map) {  
060 if (map instanceof Unmodifiable) {  
061 @SuppressWarnings("unchecked") // safe to upcast  
062 final SortedBidiMap<K, V> tmpMap = (SortedBidiMap<K, V>) map;  
063 return tmpMap;  
064 }  
065 return new UnmodifiableSortedBidiMap<>(map);  
066 }  
067  
068 //-----------------------------------------------------------------------  
069 /\*\*  
070 \* Constructor that wraps (not copies).  
071 \*  
072 \* @param map the map to decorate, must not be null  
073 \* @throws NullPointerException if map is null  
074 \*/  
075 @SuppressWarnings("unchecked") // safe to upcast  
076 private UnmodifiableSortedBidiMap(final SortedBidiMap<K, ? extends V> map) {  
077 super((SortedBidiMap<K, V>) map);  
078 }  
079  
080 //-----------------------------------------------------------------------  
081 @Override  
082 public void clear() {  
083 throw new UnsupportedOperationException();  
084 }  
085  
086 @Override  
087 public V put(final K key, final V value) {  
088 throw new UnsupportedOperationException();  
089 }  
090  
091 @Override  
092 public void putAll(final Map<? extends K, ? extends V> mapToCopy) {  
093 throw new UnsupportedOperationException();  
094 }  
095  
096 @Override  
097 public V remove(final Object key) {  
098 throw new UnsupportedOperationException();  
099 }  
100  
101 @Override  
102 public Set<Map.Entry<K, V>> entrySet() {  
103 final Set<Map.Entry<K, V>> set = super.entrySet();  
104 return UnmodifiableEntrySet.unmodifiableEntrySet(set);  
105 }  
106  
107 @Override  
108 public Set<K> keySet() {  
109 final Set<K> set = super.keySet();  
110 return UnmodifiableSet.unmodifiableSet(set);  
111 }  
112  
113 @Override  
114 public Set<V> values() {  
115 final Set<V> set = super.values();  
116 return UnmodifiableSet.unmodifiableSet(set);  
117 }  
118  
119 //-----------------------------------------------------------------------  
120 @Override  
121 public K removeValue(final Object value) {  
122 throw new UnsupportedOperationException();  
123 }  
124  
125 //-----------------------------------------------------------------------  
126 @Override  
127 public OrderedMapIterator<K, V> mapIterator() {  
128 final OrderedMapIterator<K, V> it = decorated().mapIterator();  
129 return UnmodifiableOrderedMapIterator.unmodifiableOrderedMapIterator(it);  
130 }  
131  
132 //-----------------------------------------------------------------------  
133 @Override  
134 public SortedBidiMap<V, K> inverseBidiMap() {  
135 if (inverse == null) {  
136 inverse = new UnmodifiableSortedBidiMap<>(decorated().inverseBidiMap());  
137 inverse.inverse = this;  
138 }  
139 return inverse;  
140 }  
141  
142 @Override  
143 public SortedMap<K, V> subMap(final K fromKey, final K toKey) {  
144 final SortedMap<K, V> sm = decorated().subMap(fromKey, toKey);  
145 return UnmodifiableSortedMap.unmodifiableSortedMap(sm);  
146 }  
147  
148 @Override  
149 public SortedMap<K, V> headMap(final K toKey) {  
150 final SortedMap<K, V> sm = decorated().headMap(toKey);  
151 return UnmodifiableSortedMap.unmodifiableSortedMap(sm);  
152 }  
153  
154 @Override  
155 public SortedMap<K, V> tailMap(final K fromKey) {  
156 final SortedMap<K, V> sm = decorated().tailMap(fromKey);  
157 return UnmodifiableSortedMap.unmodifiableSortedMap(sm);  
158 }  
159  
160}