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.OrderedBidiMap;  
023import org.apache.commons.collections4.OrderedMapIterator;  
024import org.apache.commons.collections4.Unmodifiable;  
025import org.apache.commons.collections4.iterators.UnmodifiableOrderedMapIterator;  
026import org.apache.commons.collections4.map.UnmodifiableEntrySet;  
027import org.apache.commons.collections4.set.UnmodifiableSet;  
028  
029/\*\*  
030 \* Decorates another {@link OrderedBidiMap} 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 UnmodifiableOrderedBidiMap<K, V>  
040 extends AbstractOrderedBidiMapDecorator<K, V> implements Unmodifiable {  
041  
042 /\*\* The inverse unmodifiable map \*/  
043 private UnmodifiableOrderedBidiMap<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 OrderedBidiMap  
054 \* @throws NullPointerException if map is null  
055 \* @since 4.0  
056 \*/  
057 public static <K, V> OrderedBidiMap<K, V> unmodifiableOrderedBidiMap(  
058 final OrderedBidiMap<? extends K, ? extends V> map) {  
059 if (map instanceof Unmodifiable) {  
060 @SuppressWarnings("unchecked") // safe to upcast  
061 final OrderedBidiMap<K, V> tmpMap = (OrderedBidiMap<K, V>) map;  
062 return tmpMap;  
063 }  
064 return new UnmodifiableOrderedBidiMap<>(map);  
065 }  
066  
067 //-----------------------------------------------------------------------  
068 /\*\*  
069 \* Constructor that wraps (not copies).  
070 \*  
071 \* @param map the map to decorate, must not be null  
072 \* @throws NullPointerException if map is null  
073 \*/  
074 @SuppressWarnings("unchecked") // safe to upcast  
075 private UnmodifiableOrderedBidiMap(final OrderedBidiMap<? extends K, ? extends V> map) {  
076 super((OrderedBidiMap<K, V>) map);  
077 }  
078  
079 //-----------------------------------------------------------------------  
080 @Override  
081 public void clear() {  
082 throw new UnsupportedOperationException();  
083 }  
084  
085 @Override  
086 public V put(final K key, final V value) {  
087 throw new UnsupportedOperationException();  
088 }  
089  
090 @Override  
091 public void putAll(final Map<? extends K, ? extends V> mapToCopy) {  
092 throw new UnsupportedOperationException();  
093 }  
094  
095 @Override  
096 public V remove(final Object key) {  
097 throw new UnsupportedOperationException();  
098 }  
099  
100 @Override  
101 public Set<Map.Entry<K, V>> entrySet() {  
102 final Set<Map.Entry<K, V>> set = super.entrySet();  
103 return UnmodifiableEntrySet.unmodifiableEntrySet(set);  
104 }  
105  
106 @Override  
107 public Set<K> keySet() {  
108 final Set<K> set = super.keySet();  
109 return UnmodifiableSet.unmodifiableSet(set);  
110 }  
111  
112 @Override  
113 public Set<V> values() {  
114 final Set<V> set = super.values();  
115 return UnmodifiableSet.unmodifiableSet(set);  
116 }  
117  
118 //-----------------------------------------------------------------------  
119 @Override  
120 public K removeValue(final Object value) {  
121 throw new UnsupportedOperationException();  
122 }  
123  
124 @Override  
125 public OrderedBidiMap<V, K> inverseBidiMap() {  
126 return inverseOrderedBidiMap();  
127 }  
128  
129 //-----------------------------------------------------------------------  
130 @Override  
131 public OrderedMapIterator<K, V> mapIterator() {  
132 final OrderedMapIterator<K, V> it = decorated().mapIterator();  
133 return UnmodifiableOrderedMapIterator.unmodifiableOrderedMapIterator(it);  
134 }  
135  
136 /\*\*  
137 \* Gets an unmodifiable view of this map where the keys and values are reversed.  
138 \*  
139 \* @return an inverted unmodifiable bidirectional map  
140 \*/  
141 public OrderedBidiMap<V, K> inverseOrderedBidiMap() {  
142 if (inverse == null) {  
143 inverse = new UnmodifiableOrderedBidiMap<>(decorated().inverseBidiMap());  
144 inverse.inverse = this;  
145 }  
146 return inverse;  
147 }  
148  
149}