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 org.apache.commons.collections4.OrderedBidiMap;  
020import org.apache.commons.collections4.OrderedMapIterator;  
021  
022/\*\*  
023 \* Provides a base decorator that enables additional functionality to be added  
024 \* to an OrderedBidiMap via decoration.  
025 \* <p>  
026 \* Methods are forwarded directly to the decorated map.  
027 \* </p>  
028 \* <p>  
029 \* This implementation does not perform any special processing with the map views.  
030 \* Instead it simply returns the inverse from the wrapped map. This may be  
031 \* undesirable, for example if you are trying to write a validating implementation  
032 \* it would provide a loophole around the validation.  
033 \* But, you might want that loophole, so this class is kept simple.  
034 \* </p>  
035 \*  
036 \* @param <K> the type of the keys in this map  
037 \* @param <V> the type of the values in this map  
038 \* @since 3.0  
039 \*/  
040public abstract class AbstractOrderedBidiMapDecorator<K, V>  
041 extends AbstractBidiMapDecorator<K, V>  
042 implements OrderedBidiMap<K, V> {  
043  
044 /\*\*  
045 \* Constructor that wraps (not copies).  
046 \*  
047 \* @param map the map to decorate, must not be null  
048 \* @throws NullPointerException if the collection is null  
049 \*/  
050 protected AbstractOrderedBidiMapDecorator(final OrderedBidiMap<K, V> map) {  
051 super(map);  
052 }  
053  
054 /\*\*  
055 \* Gets the map being decorated.  
056 \*  
057 \* @return the decorated map  
058 \*/  
059 @Override  
060 protected OrderedBidiMap<K, V> decorated() {  
061 return (OrderedBidiMap<K, V>) super.decorated();  
062 }  
063  
064 //-----------------------------------------------------------------------  
065 @Override  
066 public OrderedMapIterator<K, V> mapIterator() {  
067 return decorated().mapIterator();  
068 }  
069  
070 @Override  
071 public K firstKey() {  
072 return decorated().firstKey();  
073 }  
074  
075 @Override  
076 public K lastKey() {  
077 return decorated().lastKey();  
078 }  
079  
080 @Override  
081 public K nextKey(final K key) {  
082 return decorated().nextKey(key);  
083 }  
084  
085 @Override  
086 public K previousKey(final K key) {  
087 return decorated().previousKey(key);  
088 }  
089  
090 @Override  
091 public OrderedBidiMap<V, K> inverseBidiMap() {  
092 return decorated().inverseBidiMap();  
093 }  
094  
095}