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.Comparator;  
020import java.util.SortedMap;  
021  
022import org.apache.commons.collections4.SortedBidiMap;  
023  
024/\*\*  
025 \* Provides a base decorator that enables additional functionality to be added  
026 \* to a SortedBidiMap via decoration.  
027 \* <p>  
028 \* Methods are forwarded directly to the decorated map.  
029 \* </p>  
030 \* <p>  
031 \* This implementation does not perform any special processing with the map views.  
032 \* Instead it simply returns the inverse from the wrapped map. This may be  
033 \* undesirable, for example if you are trying to write a validating implementation  
034 \* it would provide a loophole around the validation.  
035 \* But, you might want that loophole, so this class is kept simple.  
036 \* </p>  
037 \*  
038 \* @param <K> the type of the keys in this map  
039 \* @param <V> the type of the values in this map  
040 \* @since 3.0  
041 \*/  
042public abstract class AbstractSortedBidiMapDecorator<K, V>  
043 extends AbstractOrderedBidiMapDecorator<K, V> implements SortedBidiMap<K, V> {  
044  
045 /\*\*  
046 \* Constructor that wraps (not copies).  
047 \*  
048 \* @param map the map to decorate, must not be null  
049 \* @throws NullPointerException if the collection is null  
050 \*/  
051 public AbstractSortedBidiMapDecorator(final SortedBidiMap<K, V> map) {  
052 super(map);  
053 }  
054  
055 /\*\*  
056 \* Gets the map being decorated.  
057 \*  
058 \* @return the decorated map  
059 \*/  
060 @Override  
061 protected SortedBidiMap<K, V> decorated() {  
062 return (SortedBidiMap<K, V>) super.decorated();  
063 }  
064  
065 //-----------------------------------------------------------------------  
066 @Override  
067 public SortedBidiMap<V, K> inverseBidiMap() {  
068 return decorated().inverseBidiMap();  
069 }  
070  
071 @Override  
072 public Comparator<? super K> comparator() {  
073 return decorated().comparator();  
074 }  
075  
076 @Override  
077 public Comparator<? super V> valueComparator() {  
078 return decorated().valueComparator();  
079 }  
080  
081 @Override  
082 public SortedMap<K, V> subMap(final K fromKey, final K toKey) {  
083 return decorated().subMap(fromKey, toKey);  
084 }  
085  
086 @Override  
087 public SortedMap<K, V> headMap(final K toKey) {  
088 return decorated().headMap(toKey);  
089 }  
090  
091 @Override  
092 public SortedMap<K, V> tailMap(final K fromKey) {  
093 return decorated().tailMap(fromKey);  
094 }  
095  
096}