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.set;  
018  
019import java.util.Comparator;  
020import java.util.Set;  
021import java.util.SortedSet;  
022  
023/\*\*  
024 \* Decorates another <code>SortedSet</code> to provide additional behaviour.  
025 \* <p>  
026 \* Methods are forwarded directly to the decorated set.  
027 \* </p>  
028 \*  
029 \* @param <E> the type of the elements in the sorted set  
030 \* @since 3.0  
031 \*/  
032public abstract class AbstractSortedSetDecorator<E>  
033 extends AbstractSetDecorator<E>  
034 implements SortedSet<E> {  
035  
036 /\*\* Serialization version \*/  
037 private static final long serialVersionUID = -3462240946294214398L;  
038  
039 /\*\*  
040 \* Constructor only used in deserialization, do not use otherwise.  
041 \* @since 3.1  
042 \*/  
043 protected AbstractSortedSetDecorator() {  
044 super();  
045 }  
046  
047 /\*\*  
048 \* Constructor that wraps (not copies).  
049 \*  
050 \* @param set the set to decorate, must not be null  
051 \* @throws NullPointerException if set is null  
052 \*/  
053 protected AbstractSortedSetDecorator(final Set<E> set) {  
054 super(set);  
055 }  
056  
057 /\*\*  
058 \* Gets the set being decorated.  
059 \*  
060 \* @return the decorated set  
061 \*/  
062 @Override  
063 protected SortedSet<E> decorated() {  
064 return (SortedSet<E>) super.decorated();  
065 }  
066  
067 //-----------------------------------------------------------------------  
068 @Override  
069 public SortedSet<E> subSet(final E fromElement, final E toElement) {  
070 return decorated().subSet(fromElement, toElement);  
071 }  
072  
073 @Override  
074 public SortedSet<E> headSet(final E toElement) {  
075 return decorated().headSet(toElement);  
076 }  
077  
078 @Override  
079 public SortedSet<E> tailSet(final E fromElement) {  
080 return decorated().tailSet(fromElement);  
081 }  
082  
083 @Override  
084 public E first() {  
085 return decorated().first();  
086 }  
087  
088 @Override  
089 public E last() {  
090 return decorated().last();  
091 }  
092  
093 @Override  
094 public Comparator<? super E> comparator() {  
095 return decorated().comparator();  
096 }  
097  
098}