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.map;  
018  
019import java.util.Comparator;  
020import java.util.SortedMap;  
021  
022import org.apache.commons.collections4.Factory;  
023import org.apache.commons.collections4.Transformer;  
024  
025/\*\*  
026 \* Decorates another <code>SortedMap</code> to create objects in the map on demand.  
027 \* <p>  
028 \* When the {@link #get(Object)} method is called with a key that does not  
029 \* exist in the map, the factory is used to create the object. The created  
030 \* object will be added to the map using the requested key.  
031 \* </p>  
032 \* <p>  
033 \* For instance:  
034 \* </p>  
035 \* <pre>  
036 \* Factory<Date> factory = new Factory<Date>() {  
037 \* public Date create() {  
038 \* return new Date();  
039 \* }  
040 \* }  
041 \* SortedMap<String, Date> lazy =  
042 \* LazySortedMap.lazySortedMap(new HashMap<String, Date>(), factory);  
043 \* Date date = lazy.get("NOW");  
044 \* </pre>  
045 \*  
046 \* <p>  
047 \* After the above code is executed, <code>date</code> will refer to  
048 \* a new <code>Date</code> instance. Furthermore, that <code>Date</code>  
049 \* instance is mapped to the "NOW" key in the map.  
050 \* </p>  
051 \* <p>  
052 \* <strong>Note that LazySortedMap is not synchronized and is not thread-safe.</strong>  
053 \* If you wish to use this map from multiple threads concurrently, you must use  
054 \* appropriate synchronization. The simplest approach is to wrap this map  
055 \* using {@link java.util.Collections#synchronizedSortedMap}. This class may throw  
056 \* exceptions when accessed by concurrent threads without synchronization.  
057 \* </p>  
058 \* <p>  
059 \* This class is Serializable from Commons Collections 3.1.  
060 \* </p>  
061 \*  
062 \* @param <K> the type of the keys in this map  
063 \* @param <V> the type of the values in this map  
064 \* @since 3.0  
065 \*/  
066public class LazySortedMap<K,V> extends LazyMap<K,V> implements SortedMap<K,V> {  
067  
068 /\*\* Serialization version \*/  
069 private static final long serialVersionUID = 2715322183617658933L;  
070  
071 /\*\*  
072 \* Factory method to create a lazily instantiated sorted map.  
073 \*  
074 \* @param <K> the key type  
075 \* @param <V> the value type  
076 \* @param map the map to decorate, must not be null  
077 \* @param factory the factory to use, must not be null  
078 \* @return a new lazy sorted map  
079 \* @throws NullPointerException if map or factory is null  
080 \* @since 4.0  
081 \*/  
082 public static <K, V> LazySortedMap<K, V> lazySortedMap(final SortedMap<K, V> map,  
083 final Factory<? extends V> factory) {  
084 return new LazySortedMap<>(map, factory);  
085 }  
086  
087 /\*\*  
088 \* Factory method to create a lazily instantiated sorted map.  
089 \*  
090 \* @param <K> the key type  
091 \* @param <V> the value type  
092 \* @param map the map to decorate, must not be null  
093 \* @param factory the factory to use, must not be null  
094 \* @return a new lazy sorted map  
095 \* @throws NullPointerException if map or factory is null  
096 \* @since 4.0  
097 \*/  
098 public static <K, V> LazySortedMap<K, V> lazySortedMap(final SortedMap<K, V> map,  
099 final Transformer<? super K, ? extends V> factory) {  
100 return new LazySortedMap<>(map, factory);  
101 }  
102  
103 //-----------------------------------------------------------------------  
104 /\*\*  
105 \* Constructor that wraps (not copies).  
106 \*  
107 \* @param map the map to decorate, must not be null  
108 \* @param factory the factory to use, must not be null  
109 \* @throws NullPointerException if map or factory is null  
110 \*/  
111 protected LazySortedMap(final SortedMap<K,V> map, final Factory<? extends V> factory) {  
112 super(map, factory);  
113 }  
114  
115 /\*\*  
116 \* Constructor that wraps (not copies).  
117 \*  
118 \* @param map the map to decorate, must not be null  
119 \* @param factory the factory to use, must not be null  
120 \* @throws NullPointerException if map or factory is null  
121 \*/  
122 protected LazySortedMap(final SortedMap<K,V> map, final Transformer<? super K, ? extends V> factory) {  
123 super(map, factory);  
124 }  
125  
126 //-----------------------------------------------------------------------  
127 /\*\*  
128 \* Gets the map being decorated.  
129 \*  
130 \* @return the decorated map  
131 \*/  
132 protected SortedMap<K,V> getSortedMap() {  
133 return (SortedMap<K,V>) map;  
134 }  
135  
136 //-----------------------------------------------------------------------  
137 @Override  
138 public K firstKey() {  
139 return getSortedMap().firstKey();  
140 }  
141  
142 @Override  
143 public K lastKey() {  
144 return getSortedMap().lastKey();  
145 }  
146  
147 @Override  
148 public Comparator<? super K> comparator() {  
149 return getSortedMap().comparator();  
150 }  
151  
152 @Override  
153 public SortedMap<K,V> subMap(final K fromKey, final K toKey) {  
154 final SortedMap<K,V> map = getSortedMap().subMap(fromKey, toKey);  
155 return new LazySortedMap<>(map, factory);  
156 }  
157  
158 @Override  
159 public SortedMap<K,V> headMap(final K toKey) {  
160 final SortedMap<K,V> map = getSortedMap().headMap(toKey);  
161 return new LazySortedMap<>(map, factory);  
162 }  
163  
164 @Override  
165 public SortedMap<K,V> tailMap(final K fromKey) {  
166 final SortedMap<K,V> map = getSortedMap().tailMap(fromKey);  
167 return new LazySortedMap<>(map, factory);  
168 }  
169  
170}