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.lang.reflect.Array;  
020import java.util.Collection;  
021import java.util.Iterator;  
022import java.util.Map;  
023import java.util.Set;  
024import java.util.function.Predicate;  
025  
026import org.apache.commons.collections4.Unmodifiable;  
027import org.apache.commons.collections4.iterators.AbstractIteratorDecorator;  
028import org.apache.commons.collections4.keyvalue.AbstractMapEntryDecorator;  
029import org.apache.commons.collections4.set.AbstractSetDecorator;  
030  
031/\*\*  
032 \* Decorates a map entry <code>Set</code> to ensure it can't be altered.  
033 \* <p>  
034 \* Attempts to modify it will result in an UnsupportedOperationException.  
035 \* </p>  
036 \*  
037 \* @param <K> the type of the keys in the map  
038 \* @param <V> the type of the values in the map  
039 \*  
040 \* @since 3.0  
041 \*/  
042public final class UnmodifiableEntrySet<K, V>  
043 extends AbstractSetDecorator<Map.Entry<K, V>> implements Unmodifiable {  
044  
045 /\*\* Serialization version \*/  
046 private static final long serialVersionUID = 1678353579659253473L;  
047  
048 /\*\*  
049 \* Factory method to create an unmodifiable set of Map Entry objects.  
050 \*  
051 \* @param <K> the key type  
052 \* @param <V> the value type  
053 \* @param set the set to decorate, must not be null  
054 \* @return a new unmodifiable entry set  
055 \* @throws NullPointerException if set is null  
056 \* @since 4.0  
057 \*/  
058 public static <K, V> Set<Map.Entry<K, V>> unmodifiableEntrySet(final Set<Map.Entry<K, V>> set) {  
059 if (set instanceof Unmodifiable) {  
060 return set;  
061 }  
062 return new UnmodifiableEntrySet<>(set);  
063 }  
064  
065 //-----------------------------------------------------------------------  
066 /\*\*  
067 \* Constructor that wraps (not copies).  
068 \*  
069 \* @param set the set to decorate, must not be null  
070 \* @throws NullPointerException if set is null  
071 \*/  
072 private UnmodifiableEntrySet(final Set<Map.Entry<K, V>> set) {  
073 super(set);  
074 }  
075  
076 //-----------------------------------------------------------------------  
077 @Override  
078 public boolean add(final Map.Entry<K, V> object) {  
079 throw new UnsupportedOperationException();  
080 }  
081  
082 @Override  
083 public boolean addAll(final Collection<? extends Map.Entry<K, V>> coll) {  
084 throw new UnsupportedOperationException();  
085 }  
086  
087 @Override  
088 public void clear() {  
089 throw new UnsupportedOperationException();  
090 }  
091  
092 @Override  
093 public boolean remove(final Object object) {  
094 throw new UnsupportedOperationException();  
095 }  
096  
097 /\*\*  
098 \* @since 4.4  
099 \*/  
100 @Override  
101 public boolean removeIf(Predicate<? super Map.Entry<K, V>> filter) {  
102 throw new UnsupportedOperationException();  
103 }  
104  
105 @Override  
106 public boolean removeAll(final Collection<?> coll) {  
107 throw new UnsupportedOperationException();  
108 }  
109  
110 @Override  
111 public boolean retainAll(final Collection<?> coll) {  
112 throw new UnsupportedOperationException();  
113 }  
114  
115 //-----------------------------------------------------------------------  
116 @Override  
117 public Iterator<Map.Entry<K, V>> iterator() {  
118 return new UnmodifiableEntrySetIterator(decorated().iterator());  
119 }  
120  
121 @Override  
122 @SuppressWarnings("unchecked")  
123 public Object[] toArray() {  
124 final Object[] array = decorated().toArray();  
125 for (int i = 0; i < array.length; i++) {  
126 array[i] = new UnmodifiableEntry((Map.Entry<K, V>) array[i]);  
127 }  
128 return array;  
129 }  
130  
131 @Override  
132 @SuppressWarnings("unchecked")  
133 public <T> T[] toArray(final T[] array) {  
134 Object[] result = array;  
135 if (array.length > 0) {  
136 // we must create a new array to handle multi-threaded situations  
137 // where another thread could access data before we decorate it  
138 result = (Object[]) Array.newInstance(array.getClass().getComponentType(), 0);  
139 }  
140 result = decorated().toArray(result);  
141 for (int i = 0; i < result.length; i++) {  
142 result[i] = new UnmodifiableEntry((Map.Entry<K, V>) result[i]);  
143 }  
144  
145 // check to see if result should be returned straight  
146 if (result.length > array.length) {  
147 return (T[]) result;  
148 }  
149  
150 // copy back into input array to fulfill the method contract  
151 System.arraycopy(result, 0, array, 0, result.length);  
152 if (array.length > result.length) {  
153 array[result.length] = null;  
154 }  
155 return array;  
156 }  
157  
158 //-----------------------------------------------------------------------  
159 /\*\*  
160 \* Implementation of an entry set iterator.  
161 \*/  
162 private class UnmodifiableEntrySetIterator extends AbstractIteratorDecorator<Map.Entry<K, V>> {  
163  
164 protected UnmodifiableEntrySetIterator(final Iterator<Map.Entry<K, V>> iterator) {  
165 super(iterator);  
166 }  
167  
168 @Override  
169 public Map.Entry<K, V> next() {  
170 return new UnmodifiableEntry(getIterator().next());  
171 }  
172  
173 @Override  
174 public void remove() {  
175 throw new UnsupportedOperationException();  
176 }  
177 }  
178  
179 //-----------------------------------------------------------------------  
180 /\*\*  
181 \* Implementation of a map entry that is unmodifiable.  
182 \*/  
183 private class UnmodifiableEntry extends AbstractMapEntryDecorator<K, V> {  
184  
185 protected UnmodifiableEntry(final Map.Entry<K, V> entry) {  
186 super(entry);  
187 }  
188  
189 @Override  
190 public V setValue(final V obj) {  
191 throw new UnsupportedOperationException();  
192 }  
193 }  
194  
195}