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.keyvalue;  
018  
019import java.io.Serializable;  
020import java.util.Map;  
021  
022import org.apache.commons.collections4.KeyValue;  
023  
024/\*\*  
025 \* A {@link java.util.Map.Entry Map.Entry} tied to a map underneath.  
026 \* <p>  
027 \* This can be used to enable a map entry to make changes on the underlying  
028 \* map, however this will probably mess up any iterators.  
029 \* </p>  
030 \*  
031 \* @param <K> the type of keys  
032 \* @param <V> the type of mapped values  
033 \* @since 3.0  
034 \*/  
035public class TiedMapEntry<K, V> implements Map.Entry<K, V>, KeyValue<K, V>, Serializable {  
036  
037 /\*\* Serialization version \*/  
038 private static final long serialVersionUID = -8453869361373831205L;  
039  
040 /\*\* The map underlying the entry/iterator \*/  
041 private final Map<K, V> map;  
042  
043 /\*\* The key \*/  
044 private final K key;  
045  
046 /\*\*  
047 \* Constructs a new entry with the given Map and key.  
048 \*  
049 \* @param map the map  
050 \* @param key the key  
051 \*/  
052 public TiedMapEntry(final Map<K, V> map, final K key) {  
053 super();  
054 this.map = map;  
055 this.key = key;  
056 }  
057  
058 // Map.Entry interface  
059 //-------------------------------------------------------------------------  
060 /\*\*  
061 \* Gets the key of this entry  
062 \*  
063 \* @return the key  
064 \*/  
065 @Override  
066 public K getKey() {  
067 return key;  
068 }  
069  
070 /\*\*  
071 \* Gets the value of this entry direct from the map.  
072 \*  
073 \* @return the value  
074 \*/  
075 @Override  
076 public V getValue() {  
077 return map.get(key);  
078 }  
079  
080 /\*\*  
081 \* Sets the value associated with the key direct onto the map.  
082 \*  
083 \* @param value the new value  
084 \* @return the old value  
085 \* @throws IllegalArgumentException if the value is set to this map entry  
086 \*/  
087 @Override  
088 public V setValue(final V value) {  
089 if (value == this) {  
090 throw new IllegalArgumentException("Cannot set value to this map entry");  
091 }  
092 return map.put(key, value);  
093 }  
094  
095 /\*\*  
096 \* Compares this <code>Map.Entry</code> with another <code>Map.Entry</code>.  
097 \* <p>  
098 \* Implemented per API documentation of {@link java.util.Map.Entry#equals(Object)}  
099 \*  
100 \* @param obj the object to compare to  
101 \* @return true if equal key and value  
102 \*/  
103 @Override  
104 public boolean equals(final Object obj) {  
105 if (obj == this) {  
106 return true;  
107 }  
108 if (obj instanceof Map.Entry == false) {  
109 return false;  
110 }  
111 final Map.Entry<?,?> other = (Map.Entry<?,?>) obj;  
112 final Object value = getValue();  
113 return  
114 (key == null ? other.getKey() == null : key.equals(other.getKey())) &&  
115 (value == null ? other.getValue() == null : value.equals(other.getValue()));  
116 }  
117  
118 /\*\*  
119 \* Gets a hashCode compatible with the equals method.  
120 \* <p>  
121 \* Implemented per API documentation of {@link java.util.Map.Entry#hashCode()}  
122 \*  
123 \* @return a suitable hash code  
124 \*/  
125 @Override  
126 public int hashCode() {  
127 final Object value = getValue();  
128 return (getKey() == null ? 0 : getKey().hashCode()) ^  
129 (value == null ? 0 : value.hashCode());  
130 }  
131  
132 /\*\*  
133 \* Gets a string version of the entry.  
134 \*  
135 \* @return entry as a string  
136 \*/  
137 @Override  
138 public String toString() {  
139 return getKey() + "=" + getValue();  
140 }  
141  
142}