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.util.Map;  
020  
021import org.apache.commons.collections4.KeyValue;  
022  
023/\*\*  
024 \* Provides a base decorator that allows additional functionality to be  
025 \* added to a {@link java.util.Map.Entry Map.Entry}.  
026 \*  
027 \* @param <K> the type of keys  
028 \* @param <V> the type of mapped values  
029 \* @since 3.0  
030 \*/  
031public abstract class AbstractMapEntryDecorator<K, V> implements Map.Entry<K, V>, KeyValue<K, V> {  
032  
033 /\*\* The <code>Map.Entry</code> to decorate \*/  
034 private final Map.Entry<K, V> entry;  
035  
036 /\*\*  
037 \* Constructor that wraps (not copies).  
038 \*  
039 \* @param entry the <code>Map.Entry</code> to decorate, must not be null  
040 \* @throws NullPointerException if the collection is null  
041 \*/  
042 public AbstractMapEntryDecorator(final Map.Entry<K, V> entry) {  
043 if (entry == null) {  
044 throw new NullPointerException("Map Entry must not be null.");  
045 }  
046 this.entry = entry;  
047 }  
048  
049 /\*\*  
050 \* Gets the map being decorated.  
051 \*  
052 \* @return the decorated map  
053 \*/  
054 protected Map.Entry<K, V> getMapEntry() {  
055 return entry;  
056 }  
057  
058 //-----------------------------------------------------------------------  
059  
060 @Override  
061 public K getKey() {  
062 return entry.getKey();  
063 }  
064  
065 @Override  
066 public V getValue() {  
067 return entry.getValue();  
068 }  
069  
070 @Override  
071 public V setValue(final V object) {  
072 return entry.setValue(object);  
073 }  
074  
075 @Override  
076 public boolean equals(final Object object) {  
077 if (object == this) {  
078 return true;  
079 }  
080 return entry.equals(object);  
081 }  
082  
083 @Override  
084 public int hashCode() {  
085 return entry.hashCode();  
086 }  
087  
088 @Override  
089 public String toString() {  
090 return entry.toString();  
091 }  
092  
093}