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  
021/\*\*  
022 \* Abstract Pair class to assist with creating correct  
023 \* {@link java.util.Map.Entry Map.Entry} implementations.  
024 \*  
025 \* @param <K> the type of keys  
026 \* @param <V> the type of mapped values  
027 \* @since 3.0  
028 \*/  
029public abstract class AbstractMapEntry<K, V> extends AbstractKeyValue<K, V> implements Map.Entry<K, V> {  
030  
031 /\*\*  
032 \* Constructs a new entry with the given key and given value.  
033 \*  
034 \* @param key the key for the entry, may be null  
035 \* @param value the value for the entry, may be null  
036 \*/  
037 protected AbstractMapEntry(final K key, final V value) {  
038 super(key, value);  
039 }  
040  
041 // Map.Entry interface  
042 //-------------------------------------------------------------------------  
043 /\*\*  
044 \* Sets the value stored in this <code>Map.Entry</code>.  
045 \* <p>  
046 \* This <code>Map.Entry</code> is not connected to a Map, so only the  
047 \* local data is changed.  
048 \*  
049 \* @param value the new value  
050 \* @return the previous value  
051 \*/  
052 @Override  
053 public V setValue(final V value) { // NOPMD  
054 return super.setValue(value);  
055 }  
056  
057 /\*\*  
058 \* Compares this <code>Map.Entry</code> with another <code>Map.Entry</code>.  
059 \* <p>  
060 \* Implemented per API documentation of {@link java.util.Map.Entry#equals(Object)}  
061 \*  
062 \* @param obj the object to compare to  
063 \* @return true if equal key and value  
064 \*/  
065 @Override  
066 public boolean equals(final Object obj) {  
067 if (obj == this) {  
068 return true;  
069 }  
070 if (obj instanceof Map.Entry == false) {  
071 return false;  
072 }  
073 final Map.Entry<?, ?> other = (Map.Entry<?, ?>) obj;  
074 return  
075 (getKey() == null ? other.getKey() == null : getKey().equals(other.getKey())) &&  
076 (getValue() == null ? other.getValue() == null : getValue().equals(other.getValue()));  
077 }  
078  
079 /\*\*  
080 \* Gets a hashCode compatible with the equals method.  
081 \* <p>  
082 \* Implemented per API documentation of {@link java.util.Map.Entry#hashCode()}  
083 \*  
084 \* @return a suitable hash code  
085 \*/  
086 @Override  
087 public int hashCode() {  
088 return (getKey() == null ? 0 : getKey().hashCode()) ^  
089 (getValue() == null ? 0 : getValue().hashCode());  
090 }  
091  
092}