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 org.apache.commons.collections4.KeyValue;  
020  
021/\*\*  
022 \* Abstract pair class to assist with creating <code>KeyValue</code>  
023 \* and {@link java.util.Map.Entry Map.Entry} implementations.  
024 \*  
025 \* @param <K> the type of keys  
026 \* @param <V> the type of values  
027 \* @since 3.0  
028 \*/  
029public abstract class AbstractKeyValue<K, V> implements KeyValue<K, V> {  
030  
031 /\*\* The key \*/  
032 private K key;  
033 /\*\* The value \*/  
034 private V value;  
035  
036 /\*\*  
037 \* Constructs a new pair with the specified key and given value.  
038 \*  
039 \* @param key the key for the entry, may be null  
040 \* @param value the value for the entry, may be null  
041 \*/  
042 protected AbstractKeyValue(final K key, final V value) {  
043 super();  
044 this.key = key;  
045 this.value = value;  
046 }  
047  
048 /\*\*  
049 \* Gets the key from the pair.  
050 \*  
051 \* @return the key  
052 \*/  
053 @Override  
054 public K getKey() {  
055 return key;  
056 }  
057  
058 protected K setKey(final K key) {  
059 final K old = this.key;  
060 this.key = key;  
061 return old;  
062 }  
063  
064 /\*\*  
065 \* Gets the value from the pair.  
066 \*  
067 \* @return the value  
068 \*/  
069 @Override  
070 public V getValue() {  
071 return value;  
072 }  
073  
074 protected V setValue(final V value) {  
075 final V old = this.value;  
076 this.value = value;  
077 return old;  
078 }  
079  
080 /\*\*  
081 \* Gets a debugging String view of the pair.  
082 \*  
083 \* @return a String view of the entry  
084 \*/  
085 @Override  
086 public String toString() {  
087 return new StringBuilder()  
088 .append(getKey())  
089 .append('=')  
090 .append(getValue())  
091 .toString();  
092 }  
093  
094}