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.multimap;  
018  
019import java.io.IOException;  
020import java.io.ObjectInputStream;  
021import java.io.ObjectOutputStream;  
022import java.io.Serializable;  
023import java.util.HashMap;  
024import java.util.HashSet;  
025import java.util.Map;  
026  
027import org.apache.commons.collections4.MultiValuedMap;  
028  
029/\*\*  
030 \* Implements a {@code SetValuedMap}, using a {@link HashMap} to provide data  
031 \* storage and {@link HashSet}s as value collections. This is the standard  
032 \* implementation of a SetValuedMap.  
033 \* <p>  
034 \* <strong>Note that HashSetValuedHashMap is not synchronized and is not  
035 \* thread-safe.</strong> If you wish to use this map from multiple threads  
036 \* concurrently, you must use appropriate synchronization. This class may throw  
037 \* exceptions when accessed by concurrent threads without synchronization.  
038 \* </p>  
039 \*  
040 \* @param <K> the type of the keys in this map  
041 \* @param <V> the type of the values in this map  
042 \* @since 4.1  
043 \*/  
044public class HashSetValuedHashMap<K, V> extends AbstractSetValuedMap<K, V>  
045 implements Serializable {  
046  
047 /\*\* Serialization Version \*/  
048 private static final long serialVersionUID = 20151118L;  
049  
050 /\*\*  
051 \* The initial map capacity used when none specified in constructor.  
052 \*/  
053 private static final int DEFAULT\_INITIAL\_MAP\_CAPACITY = 16;  
054  
055 /\*\*  
056 \* The initial set capacity when using none specified in constructor.  
057 \*/  
058 private static final int DEFAULT\_INITIAL\_SET\_CAPACITY = 3;  
059  
060 /\*\*  
061 \* The initial list capacity when creating a new value collection.  
062 \*/  
063 private final int initialSetCapacity;  
064  
065 /\*\*  
066 \* Creates an empty HashSetValuedHashMap with the default initial  
067 \* map capacity (16) and the default initial set capacity (3).  
068 \*/  
069 public HashSetValuedHashMap() {  
070 this(DEFAULT\_INITIAL\_MAP\_CAPACITY, DEFAULT\_INITIAL\_SET\_CAPACITY);  
071 }  
072  
073 /\*\*  
074 \* Creates an empty HashSetValuedHashMap with the default initial  
075 \* map capacity (16) and the specified initial set capacity.  
076 \*  
077 \* @param initialSetCapacity the initial capacity used for value collections  
078 \*/  
079 public HashSetValuedHashMap(final int initialSetCapacity) {  
080 this(DEFAULT\_INITIAL\_MAP\_CAPACITY, initialSetCapacity);  
081 }  
082  
083 /\*\*  
084 \* Creates an empty HashSetValuedHashMap with the specified initial  
085 \* map and list capacities.  
086 \*  
087 \* @param initialMapCapacity the initial hashmap capacity  
088 \* @param initialSetCapacity the initial capacity used for value collections  
089 \*/  
090 public HashSetValuedHashMap(final int initialMapCapacity, final int initialSetCapacity) {  
091 super(new HashMap<K, HashSet<V>>(initialMapCapacity));  
092 this.initialSetCapacity = initialSetCapacity;  
093 }  
094  
095 /\*\*  
096 \* Creates an HashSetValuedHashMap copying all the mappings of the given map.  
097 \*  
098 \* @param map a <code>MultiValuedMap</code> to copy into this map  
099 \*/  
100 public HashSetValuedHashMap(final MultiValuedMap<? extends K, ? extends V> map) {  
101 this(map.size(), DEFAULT\_INITIAL\_SET\_CAPACITY);  
102 super.putAll(map);  
103 }  
104  
105 /\*\*  
106 \* Creates an HashSetValuedHashMap copying all the mappings of the given map.  
107 \*  
108 \* @param map a <code>Map</code> to copy into this map  
109 \*/  
110 public HashSetValuedHashMap(final Map<? extends K, ? extends V> map) {  
111 this(map.size(), DEFAULT\_INITIAL\_SET\_CAPACITY);  
112 super.putAll(map);  
113 }  
114  
115 // -----------------------------------------------------------------------  
116 @Override  
117 protected HashSet<V> createCollection() {  
118 return new HashSet<>(initialSetCapacity);  
119 }  
120  
121 // -----------------------------------------------------------------------  
122 private void writeObject(final ObjectOutputStream oos) throws IOException {  
123 oos.defaultWriteObject();  
124 doWriteObject(oos);  
125 }  
126  
127 private void readObject(final ObjectInputStream ois) throws IOException, ClassNotFoundException {  
128 ois.defaultReadObject();  
129 setMap(new HashMap<K, HashSet<V>>());  
130 doReadObject(ois);  
131 }  
132  
133}