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.multiset;  
018  
019import java.io.IOException;  
020import java.io.ObjectInputStream;  
021import java.io.ObjectOutputStream;  
022import java.io.Serializable;  
023import java.util.Collection;  
024import java.util.HashMap;  
025  
026/\*\*  
027 \* Implements {@code MultiSet}, using a {@link HashMap} to provide the  
028 \* data storage. This is the standard implementation of a multiset.  
029 \* <p>  
030 \* A {@code MultiSet} stores each object in the collection together with a  
031 \* count of occurrences. Extra methods on the interface allow multiple copies  
032 \* of an object to be added or removed at once.  
033 \* </p>  
034 \*  
035 \* @param <E> the type held in the multiset  
036 \* @since 4.1  
037 \*/  
038public class HashMultiSet<E> extends AbstractMapMultiSet<E> implements Serializable {  
039  
040 /\*\* Serial version lock \*/  
041 private static final long serialVersionUID = 20150610L;  
042  
043 /\*\*  
044 \* Constructs an empty {@link HashMultiSet}.  
045 \*/  
046 public HashMultiSet() {  
047 super(new HashMap<E, MutableInteger>());  
048 }  
049  
050 /\*\*  
051 \* Constructs a multiset containing all the members of the given collection.  
052 \*  
053 \* @param coll a collection to copy into this multiset  
054 \*/  
055 public HashMultiSet(final Collection<? extends E> coll) {  
056 this();  
057 addAll(coll);  
058 }  
059  
060 //-----------------------------------------------------------------------  
061 /\*\*  
062 \* Write the multiset out using a custom routine.  
063 \*  
064 \* @param out the output stream  
065 \* @throws IOException if an error occurs while writing to the stream  
066 \*/  
067 private void writeObject(final ObjectOutputStream out) throws IOException {  
068 out.defaultWriteObject();  
069 super.doWriteObject(out);  
070 }  
071  
072 /\*\*  
073 \* Read the multiset in using a custom routine.  
074 \*  
075 \* @param in the input stream  
076 \* @throws IOException if an error occurs while reading from the stream  
077 \* @throws ClassNotFoundException if an object read from the stream can not be loaded  
078 \*/  
079 private void readObject(final ObjectInputStream in) throws IOException, ClassNotFoundException {  
080 in.defaultReadObject();  
081 setMap(new HashMap<E, MutableInteger>());  
082 super.doReadObject(in);  
083 }  
084  
085}