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.bag;  
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 Bag}, using a {@link HashMap} to provide the  
028 \* data storage. This is the standard implementation of a bag.  
029 \* <p>  
030 \* A {@code Bag} 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. It is important to read the  
033 \* interface javadoc carefully as several methods violate the  
034 \* {@link Collection} interface specification.  
035 \* </p>  
036 \*  
037 \* @param <E> the type of elements in this bag  
038 \* @since 3.0 (previously in main package v2.0)  
039 \*/  
040public class HashBag<E> extends AbstractMapBag<E> implements Serializable {  
041  
042 /\*\* Serial version lock \*/  
043 private static final long serialVersionUID = -6561115435802554013L;  
044  
045 /\*\*  
046 \* Constructs an empty {@link HashBag}.  
047 \*/  
048 public HashBag() {  
049 super(new HashMap<E, MutableInteger>());  
050 }  
051  
052 /\*\*  
053 \* Constructs a bag containing all the members of the given collection.  
054 \*  
055 \* @param coll a collection to copy into this bag  
056 \*/  
057 public HashBag(final Collection<? extends E> coll) {  
058 this();  
059 addAll(coll);  
060 }  
061  
062 //-----------------------------------------------------------------------  
063 /\*\*  
064 \* Write the bag out using a custom routine.  
065 \*  
066 \* @param out the output stream  
067 \* @throws IOException if an error occurs while writing to the stream  
068 \*/  
069 private void writeObject(final ObjectOutputStream out) throws IOException {  
070 out.defaultWriteObject();  
071 super.doWriteObject(out);  
072 }  
073  
074 /\*\*  
075 \* Read the bag in using a custom routine.  
076 \*  
077 \* @param in the input stream  
078 \* @throws IOException if an error occurs while reading from the stream  
079 \* @throws ClassNotFoundException if an object read from the stream can not be loaded  
080 \*/  
081 private void readObject(final ObjectInputStream in) throws IOException, ClassNotFoundException {  
082 in.defaultReadObject();  
083 super.doReadObject(new HashMap<E, MutableInteger>(), in);  
084 }  
085  
086}