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.util.Set;  
020  
021import org.apache.commons.collections4.Bag;  
022import org.apache.commons.collections4.collection.AbstractCollectionDecorator;  
023  
024/\*\*  
025 \* Decorates another <code>Bag</code> to provide additional behaviour.  
026 \* <p>  
027 \* Methods are forwarded directly to the decorated bag.  
028 \* </p>  
029 \*  
030 \* @param <E> the type of elements in this bag  
031 \* @since 3.0  
032 \*/  
033public abstract class AbstractBagDecorator<E>  
034 extends AbstractCollectionDecorator<E> implements Bag<E> {  
035  
036 /\*\* Serialization version \*/  
037 private static final long serialVersionUID = -3768146017343785417L;  
038  
039 /\*\*  
040 \* Constructor only used in deserialization, do not use otherwise.  
041 \* @since 3.1  
042 \*/  
043 protected AbstractBagDecorator() {  
044 super();  
045 }  
046  
047 /\*\*  
048 \* Constructor that wraps (not copies).  
049 \*  
050 \* @param bag the bag to decorate, must not be null  
051 \* @throws NullPointerException if bag is null  
052 \*/  
053 protected AbstractBagDecorator(final Bag<E> bag) {  
054 super(bag);  
055 }  
056  
057 /\*\*  
058 \* Gets the bag being decorated.  
059 \*  
060 \* @return the decorated bag  
061 \*/  
062 @Override  
063 protected Bag<E> decorated() {  
064 return (Bag<E>) super.decorated();  
065 }  
066  
067 @Override  
068 public boolean equals(final Object object) {  
069 return object == this || decorated().equals(object);  
070 }  
071  
072 @Override  
073 public int hashCode() {  
074 return decorated().hashCode();  
075 }  
076  
077 //-----------------------------------------------------------------------  
078  
079 @Override  
080 public int getCount(final Object object) {  
081 return decorated().getCount(object);  
082 }  
083  
084 @Override  
085 public boolean add(final E object, final int count) {  
086 return decorated().add(object, count);  
087 }  
088  
089 @Override  
090 public boolean remove(final Object object, final int count) {  
091 return decorated().remove(object, count);  
092 }  
093  
094 @Override  
095 public Set<E> uniqueSet() {  
096 return decorated().uniqueSet();  
097 }  
098  
099}