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.SynchronizedCollection;  
023  
024/\*\*  
025 \* Decorates another {@link Bag} to synchronize its behaviour  
026 \* for a multi-threaded environment.  
027 \* <p>  
028 \* Methods are synchronized, then forwarded to the decorated bag.  
029 \* Iterators must be separately synchronized around the loop.  
030 \* </p>  
031 \* <p>  
032 \* This class is Serializable from Commons Collections 3.1.  
033 \* </p>  
034 \*  
035 \* @param <E> the type of elements in this bag  
036 \* @since 3.0  
037 \*/  
038public class SynchronizedBag<E> extends SynchronizedCollection<E> implements Bag<E> {  
039  
040 /\*\* Serialization version \*/  
041 private static final long serialVersionUID = 8084674570753837109L;  
042  
043 /\*\*  
044 \* Factory method to create a synchronized bag.  
045 \*  
046 \* @param <E> the type of the elements in the bag  
047 \* @param bag the bag to decorate, must not be null  
048 \* @return a new synchronized Bag  
049 \* @throws NullPointerException if bag is null  
050 \* @since 4.0  
051 \*/  
052 public static <E> SynchronizedBag<E> synchronizedBag(final Bag<E> bag) {  
053 return new SynchronizedBag<>(bag);  
054 }  
055  
056 //-----------------------------------------------------------------------  
057 /\*\*  
058 \* Constructor that wraps (not copies).  
059 \*  
060 \* @param bag the bag to decorate, must not be null  
061 \* @throws NullPointerException if bag is null  
062 \*/  
063 protected SynchronizedBag(final Bag<E> bag) {  
064 super(bag);  
065 }  
066  
067 /\*\*  
068 \* Constructor that wraps (not copies).  
069 \*  
070 \* @param bag the bag to decorate, must not be null  
071 \* @param lock the lock to use, must not be null  
072 \* @throws NullPointerException if bag or lock is null  
073 \*/  
074 protected SynchronizedBag(final Bag<E> bag, final Object lock) {  
075 super(bag, lock);  
076 }  
077  
078 /\*\*  
079 \* Gets the bag being decorated.  
080 \*  
081 \* @return the decorated bag  
082 \*/  
083 protected Bag<E> getBag() {  
084 return (Bag<E>) decorated();  
085 }  
086  
087 @Override  
088 public boolean equals(final Object object) {  
089 if (object == this) {  
090 return true;  
091 }  
092 synchronized (lock) {  
093 return getBag().equals(object);  
094 }  
095 }  
096  
097 @Override  
098 public int hashCode() {  
099 synchronized (lock) {  
100 return getBag().hashCode();  
101 }  
102 }  
103  
104 //-----------------------------------------------------------------------  
105  
106 @Override  
107 public boolean add(final E object, final int count) {  
108 synchronized (lock) {  
109 return getBag().add(object, count);  
110 }  
111 }  
112  
113 @Override  
114 public boolean remove(final Object object, final int count) {  
115 synchronized (lock) {  
116 return getBag().remove(object, count);  
117 }  
118 }  
119  
120 @Override  
121 public Set<E> uniqueSet() {  
122 synchronized (lock) {  
123 final Set<E> set = getBag().uniqueSet();  
124 return new SynchronizedBagSet(set, lock);  
125 }  
126 }  
127  
128 @Override  
129 public int getCount(final Object object) {  
130 synchronized (lock) {  
131 return getBag().getCount(object);  
132 }  
133 }  
134  
135 //-----------------------------------------------------------------------  
136 /\*\*  
137 \* Synchronized Set for the Bag class.  
138 \*/  
139 class SynchronizedBagSet extends SynchronizedCollection<E> implements Set<E> {  
140 /\*\* Serialization version \*/  
141 private static final long serialVersionUID = 2990565892366827855L;  
142  
143 /\*\*  
144 \* Constructor.  
145 \* @param set the set to decorate  
146 \* @param lock the lock to use, shared with the bag  
147 \*/  
148 SynchronizedBagSet(final Set<E> set, final Object lock) {  
149 super(set, lock);  
150 }  
151 }  
152  
153}