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;  
018  
019import org.apache.commons.collections4.multiset.HashMultiSet;  
020import org.apache.commons.collections4.multiset.PredicatedMultiSet;  
021import org.apache.commons.collections4.multiset.SynchronizedMultiSet;  
022import org.apache.commons.collections4.multiset.UnmodifiableMultiSet;  
023  
024/\*\*  
025 \* Provides utility methods and decorators for {@link MultiSet} instances.  
026 \*  
027 \* @since 4.1  
028 \*/  
029public class MultiSetUtils {  
030  
031 /\*\*  
032 \* An empty unmodifiable multiset.  
033 \*/  
034 @SuppressWarnings("rawtypes") // OK, empty multiset is compatible with any type  
035 public static final MultiSet EMPTY\_MULTISET =  
036 UnmodifiableMultiSet.unmodifiableMultiSet(new HashMultiSet<>());  
037  
038 /\*\*  
039 \* Instantiation of MultiSetUtils is not intended or required.  
040 \*/  
041 private MultiSetUtils() {}  
042  
043 //-----------------------------------------------------------------------  
044 /\*\*  
045 \* Returns a synchronized (thread-safe) multiset backed by the given multiset.  
046 \* In order to guarantee serial access, it is critical that all access to the  
047 \* backing multiset is accomplished through the returned multiset.  
048 \* <p>  
049 \* It is imperative that the user manually synchronize on the returned multiset  
050 \* when iterating over it:  
051 \*  
052 \* <pre>  
053 \* MultiSet multiset = MultiSetUtils.synchronizedMultiSet(new HashMultiSet());  
054 \* ...  
055 \* synchronized(multiset) {  
056 \* Iterator i = multiset.iterator(); // Must be in synchronized block  
057 \* while (i.hasNext())  
058 \* foo(i.next());  
059 \* }  
060 \* }  
061 \* </pre>  
062 \*  
063 \* Failure to follow this advice may result in non-deterministic behavior.  
064 \*  
065 \* @param <E> the element type  
066 \* @param multiset the multiset to synchronize, must not be null  
067 \* @return a synchronized multiset backed by that multiset  
068 \* @throws NullPointerException if the MultiSet is null  
069 \*/  
070 public static <E> MultiSet<E> synchronizedMultiSet(final MultiSet<E> multiset) {  
071 return SynchronizedMultiSet.synchronizedMultiSet(multiset);  
072 }  
073  
074 /\*\*  
075 \* Returns an unmodifiable view of the given multiset. Any modification attempts  
076 \* to the returned multiset will raise an {@link UnsupportedOperationException}.  
077 \*  
078 \* @param <E> the element type  
079 \* @param multiset the multiset whose unmodifiable view is to be returned, must not be null  
080 \* @return an unmodifiable view of that multiset  
081 \* @throws NullPointerException if the MultiSet is null  
082 \*/  
083 public static <E> MultiSet<E> unmodifiableMultiSet(final MultiSet<? extends E> multiset) {  
084 return UnmodifiableMultiSet.unmodifiableMultiSet(multiset);  
085 }  
086  
087 /\*\*  
088 \* Returns a predicated (validating) multiset backed by the given multiset.  
089 \* <p>  
090 \* Only objects that pass the test in the given predicate can be added to  
091 \* the multiset. Trying to add an invalid object results in an  
092 \* IllegalArgumentException. It is important not to use the original multiset  
093 \* after invoking this method, as it is a backdoor for adding invalid  
094 \* objects.  
095 \*  
096 \* @param <E> the element type  
097 \* @param multiset the multiset to predicate, must not be null  
098 \* @param predicate the predicate for the multiset, must not be null  
099 \* @return a predicated multiset backed by the given multiset  
100 \* @throws NullPointerException if the MultiSet or Predicate is null  
101 \*/  
102 public static <E> MultiSet<E> predicatedMultiSet(final MultiSet<E> multiset,  
103 final Predicate<? super E> predicate) {  
104 return PredicatedMultiSet.predicatedMultiSet(multiset, predicate);  
105 }  
106  
107 /\*\*  
108 \* Get an empty <code>MultiSet</code>.  
109 \*  
110 \* @param <E> the element type  
111 \* @return an empty MultiSet  
112 \*/  
113 @SuppressWarnings("unchecked") // OK, empty multiset is compatible with any type  
114 public static <E> MultiSet<E> emptyMultiSet() {  
115 return EMPTY\_MULTISET;  
116 }  
117  
118}