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.set;  
018  
019import java.util.Set;  
020  
021import org.apache.commons.collections4.Transformer;  
022import org.apache.commons.collections4.collection.TransformedCollection;  
023  
024/\*\*  
025 \* Decorates another <code>Set</code> to transform objects that are added.  
026 \* <p>  
027 \* The add methods are affected by this class.  
028 \* Thus objects must be removed or searched for using their transformed form.  
029 \* For example, if the transformation converts Strings to Integers, you must  
030 \* use the Integer form to remove objects.  
031 \* </p>  
032 \* <p>  
033 \* This class is Serializable from Commons Collections 3.1.  
034 \* </p>  
035 \*  
036 \* @param <E> the type of the elements in this set  
037 \* @since 3.0  
038 \*/  
039public class TransformedSet<E> extends TransformedCollection<E> implements Set<E> {  
040  
041 /\*\* Serialization version \*/  
042 private static final long serialVersionUID = 306127383500410386L;  
043  
044 /\*\*  
045 \* Factory method to create a transforming set.  
046 \* <p>  
047 \* If there are any elements already in the set being decorated, they  
048 \* are NOT transformed.  
049 \* Contrast this with {@link #transformedSet(Set, Transformer)}.  
050 \*  
051 \* @param <E> the element type  
052 \* @param set the set to decorate, must not be null  
053 \* @param transformer the transformer to use for conversion, must not be null  
054 \* @return a new transformed set  
055 \* @throws NullPointerException if set or transformer is null  
056 \* @since 4.0  
057 \*/  
058 public static <E> TransformedSet<E> transformingSet(final Set<E> set,  
059 final Transformer<? super E, ? extends E> transformer) {  
060 return new TransformedSet<>(set, transformer);  
061 }  
062  
063 /\*\*  
064 \* Factory method to create a transforming set that will transform  
065 \* existing contents of the specified set.  
066 \* <p>  
067 \* If there are any elements already in the set being decorated, they  
068 \* will be transformed by this method.  
069 \* Contrast this with {@link #transformingSet(Set, Transformer)}.  
070 \*  
071 \* @param <E> the element type  
072 \* @param set the set to decorate, must not be null  
073 \* @param transformer the transformer to use for conversion, must not be null  
074 \* @return a new transformed set  
075 \* @throws NullPointerException if set or transformer is null  
076 \* @since 4.0  
077 \*/  
078 public static <E> Set<E> transformedSet(final Set<E> set, final Transformer<? super E, ? extends E> transformer) {  
079 final TransformedSet<E> decorated = new TransformedSet<>(set, transformer);  
080 if (set.size() > 0) {  
081 @SuppressWarnings("unchecked") // set is type E  
082 final E[] values = (E[]) set.toArray(); // NOPMD - false positive for generics  
083 set.clear();  
084 for (final E value : values) {  
085 decorated.decorated().add(transformer.transform(value));  
086 }  
087 }  
088 return decorated;  
089 }  
090  
091 //-----------------------------------------------------------------------  
092 /\*\*  
093 \* Constructor that wraps (not copies).  
094 \* <p>  
095 \* If there are any elements already in the set being decorated, they  
096 \* are NOT transformed.  
097 \*  
098 \* @param set the set to decorate, must not be null  
099 \* @param transformer the transformer to use for conversion, must not be null  
100 \* @throws NullPointerException if set or transformer is null  
101 \*/  
102 protected TransformedSet(final Set<E> set, final Transformer<? super E, ? extends E> transformer) {  
103 super(set, transformer);  
104 }  
105  
106 @Override  
107 public boolean equals(final Object object) {  
108 return object == this || decorated().equals(object);  
109 }  
110  
111 @Override  
112 public int hashCode() {  
113 return decorated().hashCode();  
114 }  
115  
116}