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.functors;  
018  
019import java.io.Serializable;  
020  
021import org.apache.commons.collections4.Transformer;  
022  
023/\*\*  
024 \* Transformer implementation that returns the same constant each time.  
025 \* <p>  
026 \* No check is made that the object is immutable. In general, only immutable  
027 \* objects should use the constant factory. Mutable objects should  
028 \* use the prototype factory.  
029 \* </p>  
030 \*  
031 \* @since 3.0  
032 \*/  
033public class ConstantTransformer<I, O> implements Transformer<I, O>, Serializable {  
034  
035 /\*\* Serial version UID \*/  
036 private static final long serialVersionUID = 6374440726369055124L;  
037  
038 /\*\* Returns null each time \*/  
039 @SuppressWarnings("rawtypes")  
040 public static final Transformer NULL\_INSTANCE = new ConstantTransformer<>(null);  
041  
042 /\*\* The closures to call in turn \*/  
043 private final O iConstant;  
044  
045 /\*\*  
046 \* Get a typed null instance.  
047 \*  
048 \* @param <I> the input type  
049 \* @param <O> the output type  
050 \* @return Transformer<I, O> that always returns null.  
051 \*/  
052 public static <I, O> Transformer<I, O> nullTransformer() {  
053 return NULL\_INSTANCE;  
054 }  
055  
056 /\*\*  
057 \* Transformer method that performs validation.  
058 \*  
059 \* @param <I> the input type  
060 \* @param <O> the output type  
061 \* @param constantToReturn the constant object to return each time in the factory  
062 \* @return the <code>constant</code> factory.  
063 \*/  
064 public static <I, O> Transformer<I, O> constantTransformer(final O constantToReturn) {  
065 if (constantToReturn == null) {  
066 return nullTransformer();  
067 }  
068 return new ConstantTransformer<>(constantToReturn);  
069 }  
070  
071 /\*\*  
072 \* Constructor that performs no validation.  
073 \* Use <code>constantTransformer</code> if you want that.  
074 \*  
075 \* @param constantToReturn the constant to return each time  
076 \*/  
077 public ConstantTransformer(final O constantToReturn) {  
078 super();  
079 iConstant = constantToReturn;  
080 }  
081  
082 /\*\*  
083 \* Transforms the input by ignoring it and returning the stored constant instead.  
084 \*  
085 \* @param input the input object which is ignored  
086 \* @return the stored constant  
087 \*/  
088 @Override  
089 public O transform(final I input) {  
090 return iConstant;  
091 }  
092  
093 /\*\*  
094 \* Gets the constant.  
095 \*  
096 \* @return the constant  
097 \* @since 3.1  
098 \*/  
099 public O getConstant() {  
100 return iConstant;  
101 }  
102  
103 /\*\*  
104 \* {@inheritDoc}  
105 \*/  
106 @Override  
107 public boolean equals(final Object obj) {  
108 if (obj == this) {  
109 return true;  
110 }  
111 if (obj instanceof ConstantTransformer == false) {  
112 return false;  
113 }  
114 final Object otherConstant = ((ConstantTransformer<?, ?>) obj).getConstant();  
115 return otherConstant == getConstant() || otherConstant != null && otherConstant.equals(getConstant());  
116 }  
117  
118 /\*\*  
119 \* {@inheritDoc}  
120 \*/  
121 @Override  
122 public int hashCode() {  
123 int result = "ConstantTransformer".hashCode() << 2;  
124 if (getConstant() != null) {  
125 result |= getConstant().hashCode();  
126 }  
127 return result;  
128 }  
129}