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.Closure;  
022import org.apache.commons.collections4.Transformer;  
023  
024/\*\*  
025 \* Closure implementation that calls a Transformer using the input object  
026 \* and ignore the result.  
027 \*  
028 \* @since 3.0  
029 \*/  
030public class TransformerClosure<E> implements Closure<E>, Serializable {  
031  
032 /\*\* Serial version UID \*/  
033 private static final long serialVersionUID = -5194992589193388969L;  
034  
035 /\*\* The transformer to wrap \*/  
036 private final Transformer<? super E, ?> iTransformer;  
037  
038 /\*\*  
039 \* Factory method that performs validation.  
040 \* <p>  
041 \* A null transformer will return the <code>NOPClosure</code>.  
042 \*  
043 \* @param <E> the type that the closure acts on  
044 \* @param transformer the transformer to call, null means nop  
045 \* @return the <code>transformer</code> closure  
046 \*/  
047 public static <E> Closure<E> transformerClosure(final Transformer<? super E, ?> transformer) {  
048 if (transformer == null) {  
049 return NOPClosure.<E>nopClosure();  
050 }  
051 return new TransformerClosure<>(transformer);  
052 }  
053  
054 /\*\*  
055 \* Constructor that performs no validation.  
056 \* Use <code>transformerClosure</code> if you want that.  
057 \*  
058 \* @param transformer the transformer to call, not null  
059 \*/  
060 public TransformerClosure(final Transformer<? super E, ?> transformer) {  
061 super();  
062 iTransformer = transformer;  
063 }  
064  
065 /\*\*  
066 \* Executes the closure by calling the decorated transformer.  
067 \*  
068 \* @param input the input object  
069 \*/  
070 @Override  
071 public void execute(final E input) {  
072 iTransformer.transform(input);  
073 }  
074  
075 /\*\*  
076 \* Gets the transformer.  
077 \*  
078 \* @return the transformer  
079 \* @since 3.1  
080 \*/  
081 public Transformer<? super E, ?> getTransformer() {  
082 return iTransformer;  
083 }  
084  
085}