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.Factory;  
022import org.apache.commons.collections4.Transformer;  
023  
024/\*\*  
025 \* Transformer implementation that calls a Factory and returns the result.  
026 \*  
027 \* @since 3.0  
028 \*/  
029public class FactoryTransformer<I, O> implements Transformer<I, O>, Serializable {  
030  
031 /\*\* Serial version UID \*/  
032 private static final long serialVersionUID = -6817674502475353160L;  
033  
034 /\*\* The factory to wrap \*/  
035 private final Factory<? extends O> iFactory;  
036  
037 /\*\*  
038 \* Factory method that performs validation.  
039 \*  
040 \* @param <I> the input type  
041 \* @param <O> the output type  
042 \* @param factory the factory to call, not null  
043 \* @return the <code>factory</code> transformer  
044 \* @throws NullPointerException if the factory is null  
045 \*/  
046 public static <I, O> Transformer<I, O> factoryTransformer(final Factory<? extends O> factory) {  
047 if (factory == null) {  
048 throw new NullPointerException("Factory must not be null");  
049 }  
050 return new FactoryTransformer<>(factory);  
051 }  
052  
053 /\*\*  
054 \* Constructor that performs no validation.  
055 \* Use <code>factoryTransformer</code> if you want that.  
056 \*  
057 \* @param factory the factory to call, not null  
058 \*/  
059 public FactoryTransformer(final Factory<? extends O> factory) {  
060 super();  
061 iFactory = factory;  
062 }  
063  
064 /\*\*  
065 \* Transforms the input by ignoring the input and returning the result of  
066 \* calling the decorated factory.  
067 \*  
068 \* @param input the input object to transform  
069 \* @return the transformed result  
070 \*/  
071 @Override  
072 public O transform(final I input) {  
073 return iFactory.create();  
074 }  
075  
076 /\*\*  
077 \* Gets the factory.  
078 \*  
079 \* @return the factory  
080 \* @since 3.1  
081 \*/  
082 public Factory<? extends O> getFactory() {  
083 return iFactory;  
084 }  
085  
086}