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.FunctorException;  
022import org.apache.commons.collections4.Transformer;  
023  
024/\*\*  
025 \* Transformer implementation that always throws an exception.  
026 \*  
027 \* @since 3.0  
028 \*/  
029public final class ExceptionTransformer<I, O> implements Transformer<I, O>, Serializable {  
030  
031 /\*\* Serial version UID \*/  
032 private static final long serialVersionUID = 7179106032121985545L;  
033  
034 /\*\* Singleton predicate instance \*/  
035 @SuppressWarnings("rawtypes") // the static instance works for all types  
036 public static final Transformer INSTANCE = new ExceptionTransformer<>();  
037  
038 /\*\*  
039 \* Factory returning the singleton instance.  
040 \*  
041 \* @param <I> the input type  
042 \* @param <O> the output type  
043 \* @return the singleton instance  
044 \* @since 3.1  
045 \*/  
046 public static <I, O> Transformer<I, O> exceptionTransformer() {  
047 return INSTANCE;  
048 }  
049  
050 /\*\*  
051 \* Restricted constructor.  
052 \*/  
053 private ExceptionTransformer() {  
054 super();  
055 }  
056  
057 /\*\*  
058 \* Transforms the input to result by cloning it.  
059 \*  
060 \* @param input the input object to transform  
061 \* @return never  
062 \* @throws FunctorException always  
063 \*/  
064 @Override  
065 public O transform(final I input) {  
066 throw new FunctorException("ExceptionTransformer invoked");  
067 }  
068  
069 private Object readResolve() {  
070 return INSTANCE;  
071 }  
072  
073}