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;  
020import java.util.Map;  
021  
022import org.apache.commons.collections4.Transformer;  
023  
024/\*\*  
025 \* Transformer implementation that returns the value held in a specified map  
026 \* using the input parameter as a key.  
027 \*  
028 \* @since 3.0  
029 \*/  
030public final class MapTransformer<I, O> implements Transformer<I, O>, Serializable {  
031  
032 /\*\* Serial version UID \*/  
033 private static final long serialVersionUID = 862391807045468939L;  
034  
035 /\*\* The map of data to lookup in \*/  
036 private final Map<? super I, ? extends O> iMap;  
037  
038 /\*\*  
039 \* Factory to create the transformer.  
040 \* <p>  
041 \* If the map is null, a transformer that always returns null is returned.  
042 \*  
043 \* @param <I> the input type  
044 \* @param <O> the output type  
045 \* @param map the map, not cloned  
046 \* @return the transformer  
047 \*/  
048 public static <I, O> Transformer<I, O> mapTransformer(final Map<? super I, ? extends O> map) {  
049 if (map == null) {  
050 return ConstantTransformer.<I, O>nullTransformer();  
051 }  
052 return new MapTransformer<>(map);  
053 }  
054  
055 /\*\*  
056 \* Constructor that performs no validation.  
057 \* Use <code>mapTransformer</code> if you want that.  
058 \*  
059 \* @param map the map to use for lookup, not cloned  
060 \*/  
061 private MapTransformer(final Map<? super I, ? extends O> map) {  
062 super();  
063 iMap = map;  
064 }  
065  
066 /\*\*  
067 \* Transforms the input to result by looking it up in a <code>Map</code>.  
068 \*  
069 \* @param input the input object to transform  
070 \* @return the transformed result  
071 \*/  
072 @Override  
073 public O transform(final I input) {  
074 return iMap.get(input);  
075 }  
076  
077 /\*\*  
078 \* Gets the map to lookup in.  
079 \*  
080 \* @return the map  
081 \* @since 3.1  
082 \*/  
083 public Map<? super I, ? extends O> getMap() {  
084 return iMap;  
085 }  
086  
087}