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;  
022  
023/\*\*  
024 \* Factory 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 ConstantFactory<T> implements Factory<T>, Serializable {  
034  
035 /\*\* Serial version UID \*/  
036 private static final long serialVersionUID = -3520677225766901240L;  
037  
038 /\*\* Returns null each time \*/  
039 @SuppressWarnings("rawtypes") // The null factory works for all object types  
040 public static final Factory NULL\_INSTANCE = new ConstantFactory<>(null);  
041  
042 /\*\* The closures to call in turn \*/  
043 private final T iConstant;  
044  
045 /\*\*  
046 \* Factory method that performs validation.  
047 \*  
048 \* @param <T> the type of the constant  
049 \* @param constantToReturn the constant object to return each time in the factory  
050 \* @return the <code>constant</code> factory.  
051 \*/  
052 public static <T> Factory<T> constantFactory(final T constantToReturn) {  
053 if (constantToReturn == null) {  
054 return NULL\_INSTANCE;  
055 }  
056 return new ConstantFactory<>(constantToReturn);  
057 }  
058  
059 /\*\*  
060 \* Constructor that performs no validation.  
061 \* Use <code>constantFactory</code> if you want that.  
062 \*  
063 \* @param constantToReturn the constant to return each time  
064 \*/  
065 public ConstantFactory(final T constantToReturn) {  
066 super();  
067 iConstant = constantToReturn;  
068 }  
069  
070 /\*\*  
071 \* Always return constant.  
072 \*  
073 \* @return the stored constant value  
074 \*/  
075 @Override  
076 public T create() {  
077 return iConstant;  
078 }  
079  
080 /\*\*  
081 \* Gets the constant.  
082 \*  
083 \* @return the constant  
084 \* @since 3.1  
085 \*/  
086 public T getConstant() {  
087 return iConstant;  
088 }  
089  
090}