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.Predicate;  
023  
024/\*\*  
025 \* Predicate implementation that throws an exception if the input is null.  
026 \*  
027 \* @since 3.0  
028 \*/  
029public final class NullIsExceptionPredicate<T> implements PredicateDecorator<T>, Serializable {  
030  
031 /\*\* Serial version UID \*/  
032 private static final long serialVersionUID = 3243449850504576071L;  
033  
034 /\*\* The predicate to decorate \*/  
035 private final Predicate<? super T> iPredicate;  
036  
037 /\*\*  
038 \* Factory to create the null exception predicate.  
039 \*  
040 \* @param <T> the type that the predicate queries  
041 \* @param predicate the predicate to decorate, not null  
042 \* @return the predicate  
043 \* @throws NullPointerException if the predicate is null  
044 \*/  
045 public static <T> Predicate<T> nullIsExceptionPredicate(final Predicate<? super T> predicate) {  
046 if (predicate == null) {  
047 throw new NullPointerException("Predicate must not be null");  
048 }  
049 return new NullIsExceptionPredicate<>(predicate);  
050 }  
051  
052 /\*\*  
053 \* Constructor that performs no validation.  
054 \* Use <code>nullIsExceptionPredicate</code> if you want that.  
055 \*  
056 \* @param predicate the predicate to call after the null check  
057 \*/  
058 public NullIsExceptionPredicate(final Predicate<? super T> predicate) {  
059 super();  
060 iPredicate = predicate;  
061 }  
062  
063 /\*\*  
064 \* Evaluates the predicate returning the result of the decorated predicate  
065 \* once a null check is performed.  
066 \*  
067 \* @param object the input object  
068 \* @return true if decorated predicate returns true  
069 \* @throws FunctorException if input is null  
070 \*/  
071 @Override  
072 public boolean evaluate(final T object) {  
073 if (object == null) {  
074 throw new FunctorException("Input Object must not be null");  
075 }  
076 return iPredicate.evaluate(object);  
077 }  
078  
079 /\*\*  
080 \* Gets the predicate being decorated.  
081 \*  
082 \* @return the predicate as the only element in an array  
083 \* @since 3.1  
084 \*/  
085 @Override  
086 @SuppressWarnings("unchecked")  
087 public Predicate<? super T>[] getPredicates() {  
088 return new Predicate[] { iPredicate };  
089 }  
090  
091}