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.Predicate;  
022  
023/\*\*  
024 \* Predicate implementation that returns false if the input is null.  
025 \*  
026 \* @since 3.0  
027 \*/  
028public final class NullIsFalsePredicate<T> implements PredicateDecorator<T>, Serializable {  
029  
030 /\*\* Serial version UID \*/  
031 private static final long serialVersionUID = -2997501534564735525L;  
032  
033 /\*\* The predicate to decorate \*/  
034 private final Predicate<? super T> iPredicate;  
035  
036 /\*\*  
037 \* Factory to create the null false predicate.  
038 \*  
039 \* @param <T> the type that the predicate queries  
040 \* @param predicate the predicate to decorate, not null  
041 \* @return the predicate  
042 \* @throws NullPointerException if the predicate is null  
043 \*/  
044 public static <T> Predicate<T> nullIsFalsePredicate(final Predicate<? super T> predicate) {  
045 if (predicate == null) {  
046 throw new NullPointerException("Predicate must not be null");  
047 }  
048 return new NullIsFalsePredicate<>(predicate);  
049 }  
050  
051 /\*\*  
052 \* Constructor that performs no validation.  
053 \* Use <code>nullIsFalsePredicate</code> if you want that.  
054 \*  
055 \* @param predicate the predicate to call after the null check  
056 \*/  
057 public NullIsFalsePredicate(final Predicate<? super T> predicate) {  
058 super();  
059 iPredicate = predicate;  
060 }  
061  
062 /\*\*  
063 \* Evaluates the predicate returning the result of the decorated predicate  
064 \* once a null check is performed.  
065 \*  
066 \* @param object the input object  
067 \* @return true if decorated predicate returns true, false if input is null  
068 \*/  
069 @Override  
070 public boolean evaluate(final T object) {  
071 if (object == null) {  
072 return false;  
073 }  
074 return iPredicate.evaluate(object);  
075 }  
076  
077 /\*\*  
078 \* Gets the predicate being decorated.  
079 \*  
080 \* @return the predicate as the only element in an array  
081 \* @since 3.1  
082 \*/  
083 @Override  
084 @SuppressWarnings("unchecked")  
085 public Predicate<? super T>[] getPredicates() {  
086 return new Predicate[] { iPredicate };  
087 }  
088  
089}