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.util.Collection;  
020  
021import org.apache.commons.collections4.Predicate;  
022  
023/\*\*  
024 \* Predicate implementation that returns true if none of the  
025 \* predicates return true.  
026 \* If the array of predicates is empty, then this predicate returns true.  
027 \* <p>  
028 \* NOTE: In versions prior to 3.2 an array size of zero or one  
029 \* threw an exception.  
030 \* </p>  
031 \*  
032 \* @since 3.0  
033 \*/  
034public final class NonePredicate<T> extends AbstractQuantifierPredicate<T> {  
035  
036 /\*\* Serial version UID \*/  
037 private static final long serialVersionUID = 2007613066565892961L;  
038  
039 /\*\*  
040 \* Factory to create the predicate.  
041 \* <p>  
042 \* If the array is size zero, the predicate always returns true.  
043 \*  
044 \* @param <T> the type that the predicate queries  
045 \* @param predicates the predicates to check, cloned, not null  
046 \* @return the <code>any</code> predicate  
047 \* @throws NullPointerException if the predicates array is null  
048 \* @throws NullPointerException if any predicate in the array is null  
049 \*/  
050 public static <T> Predicate<T> nonePredicate(final Predicate<? super T>... predicates) {  
051 FunctorUtils.validate(predicates);  
052 if (predicates.length == 0) {  
053 return TruePredicate.<T>truePredicate();  
054 }  
055 return new NonePredicate<>(FunctorUtils.copy(predicates));  
056 }  
057  
058 /\*\*  
059 \* Factory to create the predicate.  
060 \* <p>  
061 \* If the collection is size zero, the predicate always returns true.  
062 \*  
063 \* @param <T> the type that the predicate queries  
064 \* @param predicates the predicates to check, cloned, not null  
065 \* @return the <code>one</code> predicate  
066 \* @throws NullPointerException if the predicates array is null  
067 \* @throws NullPointerException if any predicate in the array is null  
068 \*/  
069 public static <T> Predicate<T> nonePredicate(final Collection<? extends Predicate<? super T>> predicates) {  
070 final Predicate<? super T>[] preds = FunctorUtils.validate(predicates);  
071 if (preds.length == 0) {  
072 return TruePredicate.<T>truePredicate();  
073 }  
074 return new NonePredicate<>(preds);  
075 }  
076  
077 /\*\*  
078 \* Constructor that performs no validation.  
079 \* Use <code>nonePredicate</code> if you want that.  
080 \*  
081 \* @param predicates the predicates to check, not cloned, not null  
082 \*/  
083 public NonePredicate(final Predicate<? super T>... predicates) {  
084 super(predicates);  
085 }  
086  
087 /\*\*  
088 \* Evaluates the predicate returning false if any stored predicate returns false.  
089 \*  
090 \* @param object the input object  
091 \* @return true if none of decorated predicates return true  
092 \*/  
093 @Override  
094 public boolean evaluate(final T object) {  
095 for (final Predicate<? super T> iPredicate : iPredicates) {  
096 if (iPredicate.evaluate(object)) {  
097 return false;  
098 }  
099 }  
100 return true;  
101 }  
102  
103}