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.Closure;  
022import org.apache.commons.collections4.Predicate;  
023  
024/\*\*  
025 \* Closure implementation acts as an if statement calling one or other closure  
026 \* based on a predicate.  
027 \*  
028 \* @since 3.0  
029 \*/  
030public class IfClosure<E> implements Closure<E>, Serializable {  
031  
032 /\*\* Serial version UID \*/  
033 private static final long serialVersionUID = 3518477308466486130L;  
034  
035 /\*\* The test \*/  
036 private final Predicate<? super E> iPredicate;  
037 /\*\* The closure to use if true \*/  
038 private final Closure<? super E> iTrueClosure;  
039 /\*\* The closure to use if false \*/  
040 private final Closure<? super E> iFalseClosure;  
041  
042 /\*\*  
043 \* Factory method that performs validation.  
044 \* <p>  
045 \* This factory creates a closure that performs no action when  
046 \* the predicate is false.  
047 \*  
048 \* @param <E> the type that the closure acts on  
049 \* @param predicate predicate to switch on  
050 \* @param trueClosure closure used if true  
051 \* @return the <code>if</code> closure  
052 \* @throws NullPointerException if either argument is null  
053 \* @since 3.2  
054 \*/  
055 public static <E> Closure<E> ifClosure(final Predicate<? super E> predicate, final Closure<? super E> trueClosure) {  
056 return IfClosure.<E>ifClosure(predicate, trueClosure, NOPClosure.<E>nopClosure());  
057 }  
058  
059 /\*\*  
060 \* Factory method that performs validation.  
061 \*  
062 \* @param <E> the type that the closure acts on  
063 \* @param predicate predicate to switch on  
064 \* @param trueClosure closure used if true  
065 \* @param falseClosure closure used if false  
066 \* @return the <code>if</code> closure  
067 \* @throws NullPointerException if any argument is null  
068 \*/  
069 public static <E> Closure<E> ifClosure(final Predicate<? super E> predicate,  
070 final Closure<? super E> trueClosure,  
071 final Closure<? super E> falseClosure) {  
072 if (predicate == null) {  
073 throw new NullPointerException("Predicate must not be null");  
074 }  
075 if (trueClosure == null || falseClosure == null) {  
076 throw new NullPointerException("Closures must not be null");  
077 }  
078 return new IfClosure<>(predicate, trueClosure, falseClosure);  
079 }  
080  
081 /\*\*  
082 \* Constructor that performs no validation.  
083 \* Use <code>ifClosure</code> if you want that.  
084 \* <p>  
085 \* This constructor creates a closure that performs no action when  
086 \* the predicate is false.  
087 \*  
088 \* @param predicate predicate to switch on, not null  
089 \* @param trueClosure closure used if true, not null  
090 \* @since 3.2  
091 \*/  
092 public IfClosure(final Predicate<? super E> predicate, final Closure<? super E> trueClosure) {  
093 this(predicate, trueClosure, NOPClosure.nopClosure());  
094 }  
095  
096 /\*\*  
097 \* Constructor that performs no validation.  
098 \* Use <code>ifClosure</code> if you want that.  
099 \*  
100 \* @param predicate predicate to switch on, not null  
101 \* @param trueClosure closure used if true, not null  
102 \* @param falseClosure closure used if false, not null  
103 \*/  
104 public IfClosure(final Predicate<? super E> predicate, final Closure<? super E> trueClosure,  
105 final Closure<? super E> falseClosure) {  
106 super();  
107 iPredicate = predicate;  
108 iTrueClosure = trueClosure;  
109 iFalseClosure = falseClosure;  
110 }  
111  
112 /\*\*  
113 \* Executes the true or false closure according to the result of the predicate.  
114 \*  
115 \* @param input the input object  
116 \*/  
117 @Override  
118 public void execute(final E input) {  
119 if (iPredicate.evaluate(input)) {  
120 iTrueClosure.execute(input);  
121 } else {  
122 iFalseClosure.execute(input);  
123 }  
124 }  
125  
126 /\*\*  
127 \* Gets the predicate.  
128 \*  
129 \* @return the predicate  
130 \* @since 3.1  
131 \*/  
132 public Predicate<? super E> getPredicate() {  
133 return iPredicate;  
134 }  
135  
136 /\*\*  
137 \* Gets the closure called when true.  
138 \*  
139 \* @return the closure  
140 \* @since 3.1  
141 \*/  
142 public Closure<? super E> getTrueClosure() {  
143 return iTrueClosure;  
144 }  
145  
146 /\*\*  
147 \* Gets the closure called when false.  
148 \*  
149 \* @return the closure  
150 \* @since 3.1  
151 \*/  
152 public Closure<? super E> getFalseClosure() {  
153 return iFalseClosure;  
154 }  
155  
156}