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 true if the input is an instanceof  
025 \* the type stored in this predicate.  
026 \*  
027 \* @since 3.0  
028 \*/  
029public final class InstanceofPredicate implements Predicate<Object>, Serializable {  
030  
031 /\*\* Serial version UID \*/  
032 private static final long serialVersionUID = -6682656911025165584L;  
033  
034 /\*\* The type to compare to \*/  
035 private final Class<?> iType;  
036  
037 /\*\*  
038 \* Factory to create the identity predicate.  
039 \*  
040 \* @param type the type to check for, may not be null  
041 \* @return the predicate  
042 \* @throws NullPointerException if the class is null  
043 \*/  
044 public static Predicate<Object> instanceOfPredicate(final Class<?> type) {  
045 if (type == null) {  
046 throw new NullPointerException("The type to check instanceof must not be null");  
047 }  
048 return new InstanceofPredicate(type);  
049 }  
050  
051 /\*\*  
052 \* Constructor that performs no validation.  
053 \* Use <code>instanceOfPredicate</code> if you want that.  
054 \*  
055 \* @param type the type to check for  
056 \*/  
057 public InstanceofPredicate(final Class<?> type) {  
058 super();  
059 iType = type;  
060 }  
061  
062 /\*\*  
063 \* Evaluates the predicate returning true if the input object is of the correct type.  
064 \*  
065 \* @param object the input object  
066 \* @return true if input is of stored type  
067 \*/  
068 @Override  
069 public boolean evaluate(final Object object) {  
070 return iType.isInstance(object);  
071 }  
072  
073 /\*\*  
074 \* Gets the type to compare to.  
075 \*  
076 \* @return the type  
077 \* @since 3.1  
078 \*/  
079 public Class<?> getType() {  
080 return iType;  
081 }  
082  
083}