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 not null.  
025 \*  
026 \* @since 3.0  
027 \*/  
028public final class NotNullPredicate<T> implements Predicate<T>, Serializable {  
029  
030 /\*\* Serial version UID \*/  
031 private static final long serialVersionUID = 7533784454832764388L;  
032  
033 /\*\* Singleton predicate instance \*/  
034 @SuppressWarnings("rawtypes")  
035 public static final Predicate INSTANCE = new NotNullPredicate<>();  
036  
037 /\*\*  
038 \* Factory returning the singleton instance.  
039 \*  
040 \* @param <T> the type that the predicate queries  
041 \* @return the singleton instance  
042 \* @since 3.1  
043 \*/  
044 public static <T> Predicate<T> notNullPredicate() {  
045 return INSTANCE;  
046 }  
047  
048 /\*\*  
049 \* Restricted constructor.  
050 \*/  
051 private NotNullPredicate() {  
052 super();  
053 }  
054  
055 /\*\*  
056 \* Evaluates the predicate returning true if the object does not equal null.  
057 \*  
058 \* @param object the object to evaluate  
059 \* @return true if not null  
060 \*/  
061 @Override  
062 public boolean evaluate(final T object) {  
063 return object != null;  
064 }  
065  
066 private Object readResolve() {  
067 return INSTANCE;  
068 }  
069  
070}