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.Factory;  
022import org.apache.commons.collections4.FunctorException;  
023  
024/\*\*  
025 \* Factory implementation that always throws an exception.  
026 \*  
027 \* @since 3.0  
028 \*/  
029public final class ExceptionFactory<T> implements Factory<T>, Serializable {  
030  
031 /\*\* Serial version UID \*/  
032 private static final long serialVersionUID = 7179106032121985545L;  
033  
034 /\*\* Singleton predicate instance \*/  
035 @SuppressWarnings("rawtypes") // the static instance works for all types  
036 public static final Factory INSTANCE = new ExceptionFactory<>();  
037  
038 /\*\*  
039 \* Factory returning the singleton instance.  
040 \*  
041 \* @param <T> the type the factory creates  
042 \* @return the singleton instance  
043 \* @since 3.1  
044 \*/  
045 public static <T> Factory<T> exceptionFactory() {  
046 return INSTANCE;  
047 }  
048  
049 /\*\*  
050 \* Restricted constructor.  
051 \*/  
052 private ExceptionFactory() {  
053 super();  
054 }  
055  
056 /\*\*  
057 \* Always throws an exception.  
058 \*  
059 \* @return never  
060 \* @throws FunctorException always  
061 \*/  
062 @Override  
063 public T create() {  
064 throw new FunctorException("ExceptionFactory invoked");  
065 }  
066  
067 private Object readResolve() {  
068 return INSTANCE;  
069 }  
070  
071}