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.iterators;  
018  
019import java.util.Enumeration;  
020import java.util.Iterator;  
021  
022/\*\*  
023 \* Adapter to make an {@link Iterator Iterator} instance appear to be an  
024 \* {@link Enumeration Enumeration} instance.  
025 \*  
026 \* @since 1.0  
027 \*/  
028public class IteratorEnumeration<E> implements Enumeration<E> {  
029  
030 /\*\* The iterator being decorated. \*/  
031 private Iterator<? extends E> iterator;  
032  
033 /\*\*  
034 \* Constructs a new <code>IteratorEnumeration</code> that will not function  
035 \* until {@link #setIterator(Iterator) setIterator} is invoked.  
036 \*/  
037 public IteratorEnumeration() {  
038 }  
039  
040 /\*\*  
041 \* Constructs a new <code>IteratorEnumeration</code> that will use the given  
042 \* iterator.  
043 \*  
044 \* @param iterator the iterator to use  
045 \*/  
046 public IteratorEnumeration(final Iterator<? extends E> iterator) {  
047 this.iterator = iterator;  
048 }  
049  
050 // Iterator interface  
051 //-------------------------------------------------------------------------  
052  
053 /\*\*  
054 \* Returns true if the underlying iterator has more elements.  
055 \*  
056 \* @return true if the underlying iterator has more elements  
057 \*/  
058 @Override  
059 public boolean hasMoreElements() {  
060 return iterator.hasNext();  
061 }  
062  
063 /\*\*  
064 \* Returns the next element from the underlying iterator.  
065 \*  
066 \* @return the next element from the underlying iterator.  
067 \* @throws java.util.NoSuchElementException if the underlying iterator has  
068 \* no more elements  
069 \*/  
070 @Override  
071 public E nextElement() {  
072 return iterator.next();  
073 }  
074  
075 // Properties  
076 //-------------------------------------------------------------------------  
077  
078 /\*\*  
079 \* Returns the underlying iterator.  
080 \*  
081 \* @return the underlying iterator  
082 \*/  
083 public Iterator<? extends E> getIterator() {  
084 return iterator;  
085 }  
086  
087 /\*\*  
088 \* Sets the underlying iterator.  
089 \*  
090 \* @param iterator the new underlying iterator  
091 \*/  
092 public void setIterator(final Iterator<? extends E> iterator) {  
093 this.iterator = iterator;  
094 }  
095  
096}