001/\*  
002 \* Copyright 2013 The Apache Software Foundation.  
003 \*  
004 \* Licensed under the Apache License, Version 2.0 (the "License");  
005 \* you may not use this file except in compliance with the License.  
006 \* You may obtain a copy of the License at  
007 \*  
008 \* http://www.apache.org/licenses/LICENSE-2.0  
009 \*  
010 \* Unless required by applicable law or agreed to in writing, software  
011 \* distributed under the License is distributed on an "AS IS" BASIS,  
012 \* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
013 \* See the License for the specific language governing permissions and  
014 \* limitations under the License.  
015 \*/  
016package org.apache.commons.collections4.iterators;  
017  
018import java.util.Iterator;  
019import java.util.NoSuchElementException;  
020  
021import org.w3c.dom.Node;  
022import org.w3c.dom.NodeList;  
023  
024/\*\*  
025 \* An {@link Iterator} over a {@link NodeList}.  
026 \* <p>  
027 \* This iterator does not support {@link #remove()} as a {@link NodeList} does not support  
028 \* removal of items.  
029 \*  
030 \* @since 4.0  
031 \* @see NodeList  
032 \*/  
033public class NodeListIterator implements Iterator<Node> {  
034  
035 /\*\* the original NodeList instance \*/  
036 private final NodeList nodeList;  
037 /\*\* The current iterator index \*/  
038 private int index = 0;  
039  
040 /\*\*  
041 \* Convenience constructor, which creates a new NodeListIterator from  
042 \* the specified node's childNodes.  
043 \*  
044 \* @param node Node, who's child nodes are wrapped by this class. Must not be null  
045 \* @throws NullPointerException if node is null  
046 \*/  
047 public NodeListIterator(final Node node) {  
048 if (node == null) {  
049 throw new NullPointerException("Node must not be null.");  
050 }  
051 this.nodeList = node.getChildNodes();  
052 }  
053  
054 /\*\*  
055 \* Constructor, that creates a new NodeListIterator from the specified  
056 \* <code>org.w3c.NodeList</code>  
057 \*  
058 \* @param nodeList node list, which is wrapped by this class. Must not be null  
059 \* @throws NullPointerException if nodeList is null  
060 \*/  
061 public NodeListIterator(final NodeList nodeList) {  
062 if (nodeList == null) {  
063 throw new NullPointerException("NodeList must not be null.");  
064 }  
065 this.nodeList = nodeList;  
066 }  
067  
068 @Override  
069 public boolean hasNext() {  
070 return nodeList != null && index < nodeList.getLength();  
071 }  
072  
073 @Override  
074 public Node next() {  
075 if (nodeList != null && index < nodeList.getLength()) {  
076 return nodeList.item(index++);  
077 }  
078 throw new NoSuchElementException("underlying nodeList has no more elements");  
079 }  
080  
081 /\*\*  
082 \* Throws {@link UnsupportedOperationException}.  
083 \*  
084 \* @throws UnsupportedOperationException always  
085 \*/  
086 @Override  
087 public void remove() {  
088 throw new UnsupportedOperationException("remove() method not supported for a NodeListIterator.");  
089 }  
090}