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.sequence;  
018  
019/\*\*  
020 \* This interface should be implemented by user object to walk  
021 \* through {@link EditScript EditScript} objects.  
022 \* <p>  
023 \* Users should implement this interface in order to walk through  
024 \* the {@link EditScript EditScript} object created by the comparison  
025 \* of two sequences. This is a direct application of the visitor  
026 \* design pattern. The {@link EditScript#visit EditScript.visit}  
027 \* method takes an object implementing this interface as an argument,  
028 \* it will perform the loop over all commands in the script and the  
029 \* proper methods of the user class will be called as the commands are  
030 \* encountered.  
031 \* </p>  
032 \* <p>  
033 \* The implementation of the user visitor class will depend on the  
034 \* need. Here are two examples.  
035 \* </p>  
036 \* <p>  
037 \* The first example is a visitor that build the longest common  
038 \* subsequence:  
039 \* </p>  
040 \* <pre>  
041 \* import org.apache.commons.collections4.comparators.sequence.CommandVisitor;  
042 \*  
043 \* import java.util.ArrayList;  
044 \*  
045 \* public class LongestCommonSubSequence implements CommandVisitor {  
046 \*  
047 \* public LongestCommonSubSequence() {  
048 \* a = new ArrayList();  
049 \* }  
050 \*  
051 \* public void visitInsertCommand(Object object) {  
052 \* }  
053 \*  
054 \* public void visitKeepCommand(Object object) {  
055 \* a.add(object);  
056 \* }  
057 \*  
058 \* public void visitDeleteCommand(Object object) {  
059 \* }  
060 \*  
061 \* public Object[] getSubSequence() {  
062 \* return a.toArray();  
063 \* }  
064 \*  
065 \* private ArrayList a;  
066 \*  
067 \* }  
068 \* </pre>  
069 \* <p>  
070 \* The second example is a visitor that shows the commands and the way  
071 \* they transform the first sequence into the second one:  
072 \* </p>  
073 \* <pre>  
074 \* import org.apache.commons.collections4.comparators.sequence.CommandVisitor;  
075 \*  
076 \* import java.util.Arrays;  
077 \* import java.util.ArrayList;  
078 \* import java.util.Iterator;  
079 \*  
080 \* public class ShowVisitor implements CommandVisitor {  
081 \*  
082 \* public ShowVisitor(Object[] sequence1) {  
083 \* v = new ArrayList();  
084 \* v.addAll(Arrays.asList(sequence1));  
085 \* index = 0;  
086 \* }  
087 \*  
088 \* public void visitInsertCommand(Object object) {  
089 \* v.insertElementAt(object, index++);  
090 \* display("insert", object);  
091 \* }  
092 \*  
093 \* public void visitKeepCommand(Object object) {  
094 \* ++index;  
095 \* display("keep ", object);  
096 \* }  
097 \*  
098 \* public void visitDeleteCommand(Object object) {  
099 \* v.remove(index);  
100 \* display("delete", object);  
101 \* }  
102 \*  
103 \* private void display(String commandName, Object object) {  
104 \* System.out.println(commandName + " " + object + " ->" + this);  
105 \* }  
106 \*  
107 \* public String toString() {  
108 \* StringBuffer buffer = new StringBuffer();  
109 \* for (Iterator iter = v.iterator(); iter.hasNext();) {  
110 \* buffer.append(' ').append(iter.next());  
111 \* }  
112 \* return buffer.toString();  
113 \* }  
114 \*  
115 \* private ArrayList v;  
116 \* private int index;  
117 \*  
118 \* }  
119 \* </pre>  
120 \*  
121 \* @since 4.0  
122 \*/  
123public interface CommandVisitor<T> {  
124  
125 /\*\*  
126 \* Method called when an insert command is encountered.  
127 \*  
128 \* @param object object to insert (this object comes from the second sequence)  
129 \*/  
130 void visitInsertCommand(T object);  
131  
132 /\*\*  
133 \* Method called when a keep command is encountered.  
134 \*  
135 \* @param object object to keep (this object comes from the first sequence)  
136 \*/  
137 void visitKeepCommand(T object);  
138  
139 /\*\*  
140 \* Method called when a delete command is encountered.  
141 \*  
142 \* @param object object to delete (this object comes from the first sequence)  
143 \*/  
144 void visitDeleteCommand(T object);  
145  
146}