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  
019import java.util.ArrayList;  
020import java.util.List;  
021  
022/\*\*  
023 \* This class gathers all the {@link EditCommand commands} needed to transform  
024 \* one objects sequence into another objects sequence.  
025 \* <p>  
026 \* An edit script is the most general view of the differences between two  
027 \* sequences. It is built as the result of the comparison between two sequences  
028 \* by the {@link SequencesComparator SequencesComparator} class. The user can  
029 \* walk through it using the <em>visitor</em> design pattern.  
030 \* </p>  
031 \* <p>  
032 \* It is guaranteed that the objects embedded in the {@link InsertCommand insert  
033 \* commands} come from the second sequence and that the objects embedded in  
034 \* either the {@link DeleteCommand delete commands} or {@link KeepCommand keep  
035 \* commands} come from the first sequence. This can be important if subclassing  
036 \* is used for some elements in the first sequence and the <code>equals</code>  
037 \* method is specialized.  
038 \* </p>  
039 \*  
040 \* @see SequencesComparator  
041 \* @see EditCommand  
042 \* @see CommandVisitor  
043 \* @see ReplacementsHandler  
044 \*  
045 \* @since 4.0  
046 \*/  
047public class EditScript<T> {  
048  
049 /\*\* Container for the commands. \*/  
050 private final List<EditCommand<T>> commands;  
051  
052 /\*\* Length of the longest common subsequence. \*/  
053 private int lcsLength;  
054  
055 /\*\* Number of modifications. \*/  
056 private int modifications;  
057  
058 /\*\*  
059 \* Simple constructor. Creates a new empty script.  
060 \*/  
061 public EditScript() {  
062 commands = new ArrayList<>();  
063 lcsLength = 0;  
064 modifications = 0;  
065 }  
066  
067 /\*\*  
068 \* Add a keep command to the script.  
069 \*  
070 \* @param command command to add  
071 \*/  
072 public void append(final KeepCommand<T> command) {  
073 commands.add(command);  
074 ++lcsLength;  
075 }  
076  
077 /\*\*  
078 \* Add an insert command to the script.  
079 \*  
080 \* @param command command to add  
081 \*/  
082 public void append(final InsertCommand<T> command) {  
083 commands.add(command);  
084 ++modifications;  
085 }  
086  
087 /\*\*  
088 \* Add a delete command to the script.  
089 \*  
090 \* @param command command to add  
091 \*/  
092 public void append(final DeleteCommand<T> command) {  
093 commands.add(command);  
094 ++modifications;  
095 }  
096  
097 /\*\*  
098 \* Visit the script. The script implements the <em>visitor</em> design  
099 \* pattern, this method is the entry point to which the user supplies its  
100 \* own visitor, the script will be responsible to drive it through the  
101 \* commands in order and call the appropriate method as each command is  
102 \* encountered.  
103 \*  
104 \* @param visitor the visitor that will visit all commands in turn  
105 \*/  
106 public void visit(final CommandVisitor<T> visitor) {  
107 for (final EditCommand<T> command : commands) {  
108 command.accept(visitor);  
109 }  
110 }  
111  
112 /\*\*  
113 \* Get the length of the Longest Common Subsequence (LCS). The length of the  
114 \* longest common subsequence is the number of {@link KeepCommand keep  
115 \* commands} in the script.  
116 \*  
117 \* @return length of the Longest Common Subsequence  
118 \*/  
119 public int getLCSLength() {  
120 return lcsLength;  
121 }  
122  
123 /\*\*  
124 \* Get the number of effective modifications. The number of effective  
125 \* modification is the number of {@link DeleteCommand delete} and  
126 \* {@link InsertCommand insert} commands in the script.  
127 \*  
128 \* @return number of effective modifications  
129 \*/  
130 public int getModifications() {  
131 return modifications;  
132 }  
133  
134}