```
 1  /*-----------------------------------------------------------------
 2  This file has been modified for use in CISC 323.  Some code has
 3  been removed to reduce the length of the inspection task.  Defects
 4  may have been added.  Please do not assume all defects found are
 5  the fault of the original author.  The choice of this class for an
 6  inspection exercise does not imply any value judgement about the
 7  quality of the original code.  It was chosen simply because it is
 8  real-world code and freely available under the GNU license.
 9  -----------------------------------------------------------------*/
10
11  // $Id: StringUtils.java,v 1.2 2001/12/07 11:41:24 ramsdell Exp $
12
13  // String processing functions.
14
15  /*
16   * Copyright 1999 by John D. Ramsdell
17   *
18   * This program is free software; you can redistribute it and/or
19   * modify it under the terms of the GNU Lesser General Public License
20   * as published by the Free Software Foundation; either version 2
21   * of the License, or (at your option) any later version.
22   *
23   * This program is distributed in the hope that it will be useful,
24   * but WITHOUT ANY WARRANTY; without even the implied warranty of
25   * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
26   * GNU Lesser General Public License for more details.
27   *
28   * You should have received a copy of the GNU Lesser General Public License
29   * along with this program; if not, write to the Free Software
30   * Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
31   */
32
33  import java.io.File;
34
35  /**
36   * String processing functions.
37   * These functions are used by the makefile loader to implement
38   * its string processing functions.
39   * The results computed by a string processing function is appended
40   * to its list argument to produce the final result.
41   * @see edu.neu.ccs.jmk.GlobalTable
42   * @version May 1999
43   * @author John D. Ramsdell
44   */
45  public class StringUtils
46  {
47
48    // No instances of this class should be constructed.
49    private StringUtils() { }
50
51    /**
52     * The patsubst function matches the pattern with the strings
53     * in the input.  A string that does not match is copied unchanged
54     * to the output.  A string that does match is replaced by a string
55     * generated by replacing wild cards in the replacement with the match.
56     * When the replacement does not contain a wild card, the match is
57     * concatenated with the replacement.
58     * @see edu.neu.ccs.jmk.Matcher
59     */
60    static StringList patsubst(String pattern,
61                               String replacement,
62                               StringList input,
63                               StringList list) {
```

```
64        if (input == null)
65          return list;
66        else {
67          StringList head = new StringList("");
68          StringList last = head;
69          Matcher matcher = new Matcher(pattern);
70          for (; input != null; input = input.getRest()) {
71            String string = input.getString();
72            String match = matcher.match(string);
73            if (match != null) {
74              if (Matcher.isPattern(replacement))
75                string = Matcher.subst(match, replacement);
76              else
77                string = match + replacement;
78            }
79            StringList temp = new StringList(string);
80            last.setRest(temp);
81            last = temp;
82          }
83          last.setRest(list);
84          return head.getRest();
85        }
86      }
87
88      /**
89       * Applies stringSubst to each element of the input.
90       */
91      static StringList subst(String pattern,
92                              String replacement,
93                              StringList input,
94                              StringList list) {
95        if (input == null)
96          return list;
97        else {
98          StringList head = new StringList("");
99          StringList last = head;
100         for (; input != null; input = input.getRest()) {
101           String string = input.getString();
102           string = stringSubst(pattern, replacement, string);
103           StringList temp = new StringList(string);
104           last.setRest(temp);
105           last = temp;
106         }
107         last.setRest(list);
108         return head.getRest();
109       }
110     }
111
112     /**
113      * Substitutes the replacement for every non-overlapping occurrence
114      * of the pattern in the string.
115      */
116     static String stringSubst(String pattern,
117                               String replacement,
118                               String string) {
119       int i = string.indexOf(pattern);
120       if (i >= 0) {               // Pattern found.
121         StringBuffer sb = new StringBuffer();
122         do {
123           sb.append(string.substring(0, i));
124           sb.append(replacement);
125           string = string.substring(i + pattern.length());
126           i = string.indexOf(pattern);
```

```
127         }
128         while (i >= 0);
129         sb.append(string);
130         string = sb.toString();
131       }
132       return string;
133     }
134
135   }
```