```
 1: // $Id: jfmt.java,v 1.1 2012-01-05 18:38:44-08 - - $
 2: //
 3: // Starter code for the jfmt utility.  This program is similar
 4: // to the example code jcat.java, which iterates over all of its
 5: // input files, except that this program shows how to use
 6: // String.split to extract non-whitespace sequences of characters
 7: // from each line.
 8: //
 9:
10: import java.io.*;
11: import java.util.LinkedList;
12: import java.util.List;
13: import java.util.Scanner;
14: import static java.lang.System.*;
15:
16: class jfmt {
17:    // Static variables keeping the general status of the program.
18:    public static final String JAR_NAME = get_jarname ();
19:    public static final int EXIT_SUCCESS = 0;
20:    public static final int EXIT_FAILURE = 1;
21:    public static int exit_status = EXIT_SUCCESS;
22:
23:    // A basename is the final component of a pathname.
24:    // If a java program is run from a jar, the classpath is the
25:    // pathname of the jar.
26:    static String get_jarname () {
27:       String jarpath = getProperty ("java.class.path");
28:       int lastslash = jarpath.lastIndexOf ('/');
29:       if (lastslash < 0) return jarpath;
30:       return jarpath.substring (lastslash + 1);
31:    }
32:
```

```
33:
34:    // Formats a single file.
35:    static void format (Scanner infile) {
36:       // Read each line from the opened file, one after the other.
37:       // Stop the loop at end of file.
38:       for (int linenr = 1; infile.hasNextLine (); ++linenr) {
39:          String line = infile.nextLine ();
40:          out.printf ("line %3d: [%s]%n", linenr, line);
41:
42:          // Create a LinkedList of Strings.
43:          List<String> words = new LinkedList<String> ();
44:
45:          // Split the line into words around white space and iterate
46:          // over the words.
47:          for (String word: line.split ("\\s+")) {
48:
49:             // Skip an empty word if such is found.
50:             if (word.length () == 0) continue;
51:             out.printf ("...[%s]%n", word);
52:             // Append the word to the end of the linked list.
53:             words.add (word);
54:
55:          }
56:          out.printf ("list:");
57:
58:          // Use iterator syntax to print out all of the words.
59:          for (String word: words) out.printf (" %s", word);
60:          out.printf ("%n");
61:       }
62:    }
63:
```

```
64:
65:    // Main function scans arguments and opens/closes files.
66:    public static void main (String[] args) {
67:       if (args.length == 0) {
68:          // There are no filenames given on the command line.
69:          out.printf ("FILE: -%n");
70:          format (new Scanner (in));
71:       }else {
72:          // Iterate over each filename given on the command line.
73:          for (int argix = 0; argix < args.length; ++argix) {
74:             String filename = args[argix];
75:             if (filename.equals ("-")) {
76:                // Treat a filename of "-" to mean System.in.
77:                out.printf ("FILE: -%n");
78:                format (new Scanner (in));
79:             }else {
80:                // Open the file and read it, or error out.
81:                try {
82:                   Scanner infile = new Scanner (new File (filename));
83:                   out.printf ("FILE: %s%n", filename);
84:                   format (infile);
85:                   infile.close ();
86:                }catch (IOException error) {
87:                   exit_status = EXIT_FAILURE;
88:                   err.printf ("%s: %s%n", JAR_NAME,
89:                               error.getMessage ());
90:                }
91:             }
92:          }
93:       }
94:       exit (exit_status);
95:    }
96:
97: }
```

```
 1: # $Id: Makefile,v 1.1 2012-01-05 19:21:49-08 - - $
 2:
 3: JAVASRC    = jfmt.java
 4: SOURCES    = ${JAVASRC} Makefile README
 5: MAINCLASS  = jfmt
 6: CLASSES    = ${JAVASRC:.java=.class}
 7: JARCLASSES = ${CLASSES}
 8: JARFILE    = jfmt
 9: SUBMITDIR  = cmps012b-wm.w12 asg1
10: LISTING    = asg1j-jfmt.code.ps
11:
12: all : ${JARFILE}
13:
14: ${JARFILE} : ${CLASSES}
15:         echo Main-class: ${MAINCLASS} >Manifest
16:         jar cvfm ${JARFILE} Manifest ${JARCLASSES}
17:         - rm Manifest
18:         chmod +x ${JARFILE}
19:
20: %.class : %.java
21:         cid + $<
22:         javac $<
23:         checksource $<
24:
25: clean :
26:         - rm ${JARCLASSES} test.output
27:
28: spotless : clean
29:         - rm ${JARFILE}
30:
31: ci : ${SOURCES}
32:         cid + ${SOURCES} test.input
33:         checksource ${SOURCES}
34:
35: lis : all
36:         ${JARFILE} test.input >test.output
37:         mkpspdf ${LISTING} ${SOURCES} test.input test.output
38:
39: submit : ${SOURCES} ci
40:         submit ${SUBMITDIR} ${SOURCES}
41:
42: again : ${SOURCES}
43:         gmake --no-print-directory spotless lis
44:
```

```
1: $Id: README,v 1.1 2012-01-05 19:20:00-08 - - $
```

```
 1: This is a small file that can be used to test your program.
 2: Also see the files in the .score subdirectory.
 3: The initial program does not do much but dump the input in debug
 4: format.
 5: You can test your program by running:
 6:    pfmt.perl .... >output1
 7:    jfmt .... >output2
 8:    diff output1 output2
 9: where .... are various arguments that might be supplied.
10: When you run diff, if your program works, it should not print anything.
11: $Id$
```

```
 1: FILE: test.input
 2: line   1: [This is a small file that can be used to test your program.]
 3: ...[This]
 4: ...[is]
 5: ...[a]
 6: ...[small]
 7: ...[file]
 8: ...[that]
 9: ...[can]
10: ...[be]
11: ...[used]
12: ...[to]
13: ...[test]
14: ...[your]
15: ...[program.]
16: list: This is a small file that can be used to test your program.
17: line   2: [Also see the files in the .score subdirectory.]
18: ...[Also]
19: ...[see]
20: ...[the]
21: ...[files]
22: ...[in]
23: ...[the]
24: ...[.score]
25: ...[subdirectory.]
26: list: Also see the files in the .score subdirectory.
27: line   3: [The initial program does not do much but dump the input in debug]
28: ...[The]
29: ...[initial]
30: ...[program]
31: ...[does]
32: ...[not]
33: ...[do]
34: ...[much]
35: ...[but]
36: ...[dump]
37: ...[the]
38: ...[input]
39: ...[in]
40: ...[debug]
41: list: The initial program does not do much but dump the input in debug
42: line   4: [format.]
43: ...[format.]
44: list: format.
45: line   5: [You can test your program by running:]
46: ...[You]
47: ...[can]
48: ...[test]
49: ...[your]
50: ...[program]
51: ...[by]
52: ...[running:]
53: list: You can test your program by running:
54: line   6: [   pfmt.perl .... >output1]
55: ...[pfmt.perl]
56: ...[....]
57: ...[>output1]
58: list: pfmt.perl .... >output1
59: line   7: [   jfmt .... >output2]
60: ...[jfmt]
61: ...[....]
62: ...[>output2]
63: list: jfmt .... >output2
64: line   8: [   diff output1 output2]
```

```
   65: ...[diff]
   66: ...[output1]
   67: ...[output2]
   68: list: diff output1 output2
   69: line   9: [where .... are various arguments that might be supplied.]
   70: ...[where]
   71: ...[....]
   72: ...[are]
   73: ...[various]
   74: ...[arguments]
   75: ...[that]
   76: ...[might]
   77: ...[be]
   78: ...[supplied.]
   79: list: where .... are various arguments that might be supplied.
   80: line  10: [When you run diff, if your program works, it should not print anythin
g.]
   81: ...[When]
   82: ...[you]
   83: ...[run]
   84: ...[diff,]
   85: ...[if]
   86: ...[your]
   87: ...[program]
   88: ...[works,]
   89: ...[it]
   90: ...[should]
   91: ...[not]
   92: ...[print]
   93: ...[anything.]
   94: list: When you run diff, if your program works, it should not print anything.
   95: line  11: [$Id$]
   96: ...[$Id$]
   97: list: $Id$
```