```
 1: // $Id: jfmt.java,v 1.1 2010-12-13 17:26:24-08 - - $
 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:
18:    static void format (Scanner infile) {
19:       for (int linenr = 1; infile.hasNextLine (); ++linenr) {
20:          String line = infile.nextLine ();
21:          out.printf ("line %3d: [%s]%n", linenr, line);
22:          List<String> words = new LinkedList<String> ();
23:          for (String word: line.split ("\\s+")) {
24:             if (word.length () == 0) continue;
25:             out.printf ("...[%s]%n", word);
26:             words.add (word);
27:          }
28:          out.printf ("list:");
29:          for (String word: words) out.printf (" %s", word);
30:          out.printf ("%n");
31:       }
32:    }
33:
34:    public static void main (String[] args) {
35:       if (args.length < 1) {
36:          out.printf ("FILE: -%n");
37:          format (new Scanner (in));
38:       }else {
39:          for (int argix = 0; argix < args.length; ++argix) {
40:             String filename = args[argix];
41:             if (filename.equals ("-")) {
42:                out.printf ("FILE: -%n");
43:                format (new Scanner (in));
44:             }else {
45:                try {
46:                   Scanner infile = new Scanner (new File (filename));
47:                   out.printf ("FILE: %s%n", filename);
48:                   format (infile);
49:                   infile.close ();
50:                }catch (IOException error) {
51:                   auxlib.warn (error.getMessage ());
52:                }
53:             }
54:          }
55:       }
56:    }
57:
58: }
```

```
 1: // $Id: auxlib.java,v 1.1 2010-12-13 17:26:24-08 - - $
 2: //
 3: // NAME
 4: //    auxlib - Auxiliary miscellanea for handling system interaction.
 5: //
 6: // DESCRIPTION
 7: //    Auxlib has system access functions that can be used by other
 8: //    classes to print appropriate messages and keep track of
 9: //    the program name and exit codes.  It assumes it is being run
10: //    from a jar and gets the name of the program from the classpath.
11: //    Can not be instantiated.
12: //
13:
14: import static java.lang.System.*;
15: import static java.lang.Integer.*;
16:
17: public final class auxlib{
18:    public static final String PROGNAME =
19:                   basename (getProperty ("java.class.path"));
20:    public static final int EXIT_SUCCESS = 0;
21:    public static final int EXIT_FAILURE = 1;
22:    public static int exitvalue = EXIT_SUCCESS;
23:
24:    //
25:    // private ctor - prevents class from new instantiation.
26:    //
27:    private auxlib () {
28:       throw new UnsupportedOperationException ();
29:    }
30:
31:    //
32:    // basename - strips the dirname and returns only the basename.
33:    //            See:  man -s 3c basename
34:    //
35:    public static String basename (String pathname) {
36:       if (pathname == null || pathname.length () == 0) return ".";
37:       String[] paths = pathname.split ("/");
38:       for (int index = paths.length - 1; index >= 0; --index) {
39:          if (paths[index].length () > 0) return paths[index];
40:       }
41:       return "/";
42:    }
43:
44:    //
45:    // Functions:
46:    //    whine   - prints a message with a given exit code.
47:    //    warn    - prints a stderr message and sets the exit code.
48:    //    die     - calls warn then exits.
49:    // Combinations of arguments:
50:    //    objname - name of the object to be printed (optional)
51:    //    message - message to be printed after the objname,
52:    //              either a Throwable or a String.
53:    //
54:    public static void whine (int exitval, Object... args) {
55:       exitvalue = exitval;
56:       err.printf ("%s", PROGNAME);
57:       for (Object argi : args) err.printf (": %s", argi);
58:       err.printf ("%n");
59:    }
60:    public static void warn (Object... args) {
61:       whine (EXIT_FAILURE, args);
62:    }
63:    public static void die (Object... args) {
64:       warn (args);
```

```
65:       exit ();
66:    }
67:
68:    //
69:    // usage_exit - prints a usage message and exits.
70:    //
71:    public static void usage_exit (String optsargs) {
72:       exitvalue = EXIT_FAILURE;
73:       err.printf ("Usage: %s %s%n", PROGNAME, optsargs);
74:       exit ();
75:    }
76:
77:    //
78:    // exit - calls exit with the appropriate code.
79:    //        This function should be called instead of returning
80:    //        from the main function.
81:    //
82:    public static void exit () {
83:       System.exit (exitvalue);
84:    }
85:
86:    //
87:    // identity - returns the default Object.toString value
88:    //            Useful for debugging.
89:    //
90:    public static String identity (Object object) {
91:       return object == null ? "(null)"
92:            : object.getClass().getName() + "@"
93:            + toHexString (identityHashCode (object));
94:    }
95:
96: }
```

```
 1: # $Id: Makefile,v 1.1 2010-12-13 17:26:24-08 - - $
 2:
 3: JAVASRC    = jfmt.java auxlib.java
 4: SOURCES    = ${JAVASRC} Makefile README pfmt.perl
 5: MAINCLASS  = jfmt
 6: CLASSES    = ${JAVASRC:.java=.class}
 7: JARCLASSES = ${CLASSES}
 8: JARFILE    = jfmt
 9: LISTING    = ../asg1j-jfmt.code.ps
10: SUBMITDIR  = cmps012b-wm.s10 asg1
11:
12: all : ${JARFILE}
13:         cid + ${SOURCES}
14:         checksource ${SOURCES}
15:
16: ${JARFILE} : ${CLASSES}
17:         echo Main-class: ${MAINCLASS} >Manifest
18:         jar cvfm ${JARFILE} Manifest ${JARCLASSES}
19:         - rm Manifest
20:         chmod +x ${JARFILE}
21:
22: %.class : %.java
23:         javac $<
24:
25: clean :
26:         - rm ${JARCLASSES}
27:
28: spotless : clean
29:         - rm ${JARFILE}
30:
31: ci : ${SOURCES}
32:         cid + ${SOURCES}
33:         checksource ${SOURCES}
34:
35: lis : ${SOURCES}
36:         mkpspdf ${LISTING} ${SOURCES}
37:
38: submit : ${SOURCES}
39:         submit ${SUBMITDIR} ${SOURCES}
40:         testsubmit ${SUBMITDIR} ${SOURCES}
41:
42: again : ${SOURCES}
43:         gmake --no-print-directory spotless all lis
44:
```

```
 1: #!/usr/bin/perl
 2: # $Id: pfmt.perl,v 1.1 2010-12-13 17:26:24-08 - - $
 3: use strict;
 4: use warnings;
 5:
 6: $0 =~ s|^(.*/)?([^/]+)/*$|$2|;
 7: $/ = "";
 8: my $exit_status = 0;
 9: END {exit $exit_status}
10: sub note(@) {print STDERR "@_"};
11: $SIG{'__WARN__'} = sub {note @_; $exit_status = 1};
12: $SIG{'__DIE__'} = sub {warn @_; exit};
13:
14: my $linelen = 65;
15: if (@ARGV and $ARGV[0] =~ m/^-(.+)/) {
16:    $linelen = $1;
17:    die "Usage: $0 [-width] [filename...]\n" if $linelen =~ m/\D/;
18:    shift @ARGV
19: }
20:
21: push @ARGV, "-" unless @ARGV;
22: for my $filename (@ARGV) {
23:    open my $file, "<$filename" or warn "$0: $filename: $!\n" and next;
24:    while (my @words = split " ", <$file>) {
25:       for (;;) {
26:          last unless my $line = shift @words;
27:          while (@words and my $newline = "$line $words[0]") {
28:             last if (length $newline) > $linelen;
29:             $line = $newline;
30:             shift @words;
31:          }
32:          print $line, "\n";
33:       }
34:       print "\n";
35:    }
36:    close $file;
37: }
38:
```