```java
// $Id: jfmt.java,v 1.2 2013-09-24 14:38:16-07 - - $
//
// Starter code for the jfmt utility.  This program is similar
// to the example code jcat.java, which iterates over all of its
// input files, except that this program shows how to use
// String.split to extract non-whitespace sequences of characters
// from each line.
//

import java.io.*;
import java.util.LinkedList;
import java.util.List;
import java.util.Scanner;
import static java.lang.System.*;

class jfmt {
   // Static variables keeping the general status of the program.
   public static final String JAR_NAME = get_jarname();
   public static final int EXIT_SUCCESS = 0;
   public static final int EXIT_FAILURE = 1;
   public static int exit_status = EXIT_SUCCESS;

   // A basename is the final component of a pathname.
   // If a java program is run from a jar, the classpath is the
   // pathname of the jar.
   static String get_jarname() {
      String jarpath = getProperty ("java.class.path");
      int lastslash = jarpath.lastIndexOf ('/');
      if (lastslash < 0) return jarpath;
      return jarpath.substring (lastslash + 1);
   }


   // Formats a single file.
   static void format (Scanner infile) {
      // Read each line from the opened file, one after the other.
      // Stop the loop at end of file.
      for (int linenr = 1; infile.hasNextLine(); ++linenr) {
         String line = infile.nextLine();
         out.printf ("line %3d: [%s]%n", linenr, line);

         // Create a LinkedList of Strings.
         List<String> words = new LinkedList<String>();

         // Split the line into words around white space and iterate
         // over the words.
         for (String word: line.split ("\\s+")) {

            // Skip an empty word if such is found.
            if (word.length() == 0) continue;
            out.printf ("...[%s]%n", word);
            // Append the word to the end of the linked list.
            words.add (word);

         }
         out.printf ("list:");

         // Use iterator syntax to print out all of the words.
         for (String word: words) out.printf (" %s", word);
         out.printf ("%n");
      }
   }


   // Main function scans arguments and opens/closes files.
   public static void main (String[] args) {
      if (args.length == 0) {
         // There are no filenames given on the command line.
         out.printf ("FILE: -%n");
         format (new Scanner (in));
      }else {
         // Iterate over each filename given on the command line.
         for (int argix = 0; argix < args.length; ++argix) {
            String filename = args[argix];
            if (filename.equals ("-")) {
               // Treat a filename of "-" to mean System.in.
               out.printf ("FILE: -%n");
               format (new Scanner (in));
            }else {
               // Open the file and read it, or error out.
               try {
                  Scanner infile = new Scanner (new File (filename));
                  out.printf ("FILE: %s%n", filename);
                  format (infile);
                  infile.close();
               }catch (IOException error) {
                  exit_status = EXIT_FAILURE;
                  err.printf ("%s: %s%n", JAR_NAME,
                              error.getMessage());
               }
            }
         }
      }
      exit (exit_status);
   }

}
```

```
# $Id: Makefile,v 1.2 2013-09-24 17:12:34-07 - - $

JAVASRC    = jfmt.java
SOURCES    = ${JAVASRC} Makefile README
MAINCLASS  = jfmt
CLASSES    = ${JAVASRC:.java=.class}
JARCLASSES = ${CLASSES}
JARFILE    = jfmt
SUBMITDIR  = cmps012b-wm.f13 asg1
LISTING    = Listing.ps

all : ${JARFILE}

${JARFILE} : ${CLASSES}
        echo Main-class: ${MAINCLASS} >Manifest
        jar cvfm ${JARFILE} Manifest ${JARCLASSES}
        - rm Manifest
        chmod +x ${JARFILE}

%.class : %.java
        cid + $<
        javac $<
        checksource $<

clean :
        - rm ${JARCLASSES} test.output

spotless : clean
        - rm ${JARFILE}

ci : ${SOURCES}
        cid + ${SOURCES} test.input
        checksource ${SOURCES}

lis : all
        ${JARFILE} test.input >test.output
        mkpspdf ${LISTING} ${SOURCES} test.input test.output

submit : ${SOURCES} ci
        submit ${SUBMITDIR} ${SOURCES}

again : ${SOURCES}
        gmake --no-print-directory spotless lis

```

```
$Id: README,v 1.1 2013-09-24 14:36:52-07 - - $
```

```
This is a small file that can be used to test your program.
Also see the files in the .score subdirectory.
The initial program does not do much but dump the input in debug
format.
You can test your program by running:
   pfmt.perl .... >output1
   jfmt .... >output2
   diff output1 output2
where .... are various arguments that might be supplied.
When you run diff, if your program works, it should not print anything.
$Id: test.input,v 1.1 2013-09-24 14:36:52-07 - - $
```

```
FILE: test.input
line   1: [This is a small file that can be used to test your program.]
...[This]
...[is]
...[a]
...[small]
...[file]
...[that]
...[can]
...[be]
...[used]
...[to]
...[test]
...[your]
...[program.]
list: This is a small file that can be used to test your program.
line   2: [Also see the files in the .score subdirectory.]
...[Also]
...[see]
...[the]
...[files]
...[in]
...[the]
...[.score]
...[subdirectory.]
list: Also see the files in the .score subdirectory.
line   3: [The initial program does not do much but dump the input in debug]
...[The]
...[initial]
...[program]
...[does]
...[not]
...[do]
...[much]
...[but]
...[dump]
...[the]
...[input]
...[in]
...[debug]
list: The initial program does not do much but dump the input in debug
line   4: [format.]
...[format.]
list: format.
line   5: [You can test your program by running:]
...[You]
...[can]
...[test]
...[your]
...[program]
...[by]
...[running:]
list: You can test your program by running:
line   6: [   pfmt.perl .... >output1]
...[pfmt.perl]
...[....]
...[>output1]
list: pfmt.perl .... >output1
line   7: [   jfmt .... >output2]
...[jfmt]
...[....]
```

```
...[>output2]
list: jfmt .... >output2
line   8: [   diff output1 output2]
...[diff]
...[output1]
...[output2]
list: diff output1 output2
line   9: [where .... are various arguments that might be supplied.]
...[where]
...[....]
...[are]
...[various]
...[arguments]
...[that]
...[might]
...[be]
...[supplied.]
list: where .... are various arguments that might be supplied.
line  10: [When you run diff, if your program works, it should not print anything.]
...[When]
...[you]
...[run]
...[diff,]
...[if]
...[your]
...[program]
...[works,]
...[it]
...[should]
...[not]
...[print]
...[anything.]
list: When you run diff, if your program works, it should not print anything.
line  11: [$Id: test.input,v 1.1 2013-09-24 14:36:52-07 - - $]
...[$Id:]
...[test.input,v]
...[1.1]
...[2013-09-24]
...[14:36:52-07]
...[-]
...[-]
...[$]
list: $Id: test.input,v 1.1 2013-09-24 14:36:52-07 - - $
```