```
 1: // $Id: jxref.java,v 1.11 2012-01-19 19:43:07-08 - - $
 2:
 3: import java.io.*;
 4: import java.util.Iterator;
 5: import java.util.Map.Entry;
 6: import java.util.NoSuchElementException;
 7: import java.util.Scanner;
 8: import java.util.regex.Matcher;
 9: import java.util.regex.Pattern;
10: import static java.lang.System.*;
11:
12: class jxref {
13:    // Static program constants.
14:    private static final String STDIN_FILENAME = "-";
15:    private static final String JARNAME = get_jarname();
16:    private static final int EXIT_SUCCESS = 0;
17:    private static final int EXIT_FAILURE = 1;
18:
19:    // Static exit status variable.
20:    private static int exit_status = EXIT_SUCCESS;
21:
22:    // A basename is the final component of a pathname.
23:    // If a java program is run from a jar, the classpath is the
24:    // pathname of the jar.
25:    private static String get_jarname() {
26:       String jarpath = getProperty ("java.class.path");
27:       int lastslash = jarpath.lastIndexOf ('/');
28:       if (lastslash < 0) return jarpath;
29:       return jarpath.substring (lastslash + 1);
30:    }
31:
```

```
32:
33:    private static final String WORD_REGEX = "\\w+([-'.:/]\\w+)*";
34:    private static final Pattern WORD_PATTERN
35:                         = Pattern.compile (WORD_REGEX);
36:    private static void xref_file (String filename, Scanner file) {
37:       err.printf ("TRACE: filename = %s%n", filename);
38:       listmap map = new listmap();
39:       for (int linenr = 1; file.hasNextLine(); ++linenr) {
40:          String line = file.nextLine();
41:          err.printf ("TRACE: %s: %4d: %s%n", filename, linenr, line);
42:          Matcher match = WORD_PATTERN.matcher (line);
43:          while (match.find()) {
44:             String word = match.group();
45:             err.printf ("TRACE: word = %s%n", word);
46:          }
47:       }
48:       for (Entry<String, intqueue> entry: map) {
49:          err.printf ("STUB: %s (%s, %s)%n", entry,
50:                      entry.getKey(), entry.getValue());
51:       }
52:    }
53:
54:    // For each filename, open the file, cross reference it, and print.
55:    private static void xref_filename (String filename) {
56:       if (filename.equals (STDIN_FILENAME)) {
57:          xref_file (filename, new Scanner (System.in));
58:       }else {
59:          try {
60:             Scanner file = new Scanner (new File (filename));
61:             xref_file (filename, file);
62:             file.close();
63:          }catch (IOException error) {
64:             exit_status = EXIT_FAILURE;
65:             err.printf ("%s: %s%n", JARNAME, error.getMessage());
66:          }
67:       }
68:    }
69:
70:    // Main function scans arguments to cross reference files.
71:    public static void main (String[] args) {
72:       if (args.length == 0) {
73:          // No files specified on the command line.
74:          xref_filename (STDIN_FILENAME);
75:       }else {
76:          // Iterate over each filename given on the command line.
77:          for (int argi = 0; argi < args.length; ++argi) {
78:             xref_filename (args[argi]);
79:          }
80:       }
81:       exit (exit_status);
82:    }
83:
84: }
85:
```

```
 1: // $Id: listmap.java,v 1.13 2012-01-19 19:43:07-08 - - $
 2:
 3: import java.util.Iterator;
 4: import java.util.Map.Entry;
 5: import java.util.NoSuchElementException;
 6: import static java.lang.System.*;
 7:
 8: class listmap implements Iterable<Entry<String, intqueue>> {
 9:
10:    private node head = null;
11:    private class node implements Entry<String, intqueue> {
12:       String key;
13:       intqueue queue = new intqueue();
14:       node link;
15:       public String getKey() {
16:          return key;
17:       }
18:       public intqueue getValue() {
19:          return queue;
20:       }
21:       public intqueue setValue (intqueue queue) {
22:          throw new UnsupportedOperationException();
23:       }
24:    }
25:
26:    public listmap() {
27:       err.printf ("TRACE: new %s%n", this);
28:    }
29:
30:    public void insert (String key, int linenr) {
31:       err.printf ("STUB: insert (%s, %s)%n", key, linenr);
32:    }
33:
34:    public Iterator<Entry<String, intqueue>> iterator() {
35:       return new itor();
36:    }
37:
38:    private class itor implements Iterator<Entry<String, intqueue>> {
39:       node curr = head;
40:
41:       public boolean hasNext() {
42:          return curr != null;
43:       }
44:
45:       public Entry<String, intqueue> next() {
46:          if (curr == null) throw new NoSuchElementException();
47:          node next = curr;
48:          curr = curr.link;
49:          return next;
50:       }
51:
52:       public void remove() {
53:          throw new UnsupportedOperationException();
54:       }
55:
56:    }
57:
58: }
```

```
 1: // $Id: intqueue.java,v 1.4 2012-01-19 19:43:07-08 - - $
 2:
 3: import java.util.Iterator;
 4: import java.util.NoSuchElementException;
 5:
 6: class intqueue implements Iterable<Integer> {
 7:
 8:    private int count = 0;
 9:    private node front = null;
10:    private node rear = null;
11:    private class node {
12:       int linenr;
13:       node link;
14:    }
15:
16:    public void insert (int number) {
17:       ++count;
18:    }
19:
20:    public boolean empty() {
21:       return count == 0;
22:    }
23:
24:    public int getcount() {
25:       return count;
26:    }
27:
28:    public Iterator<Integer> iterator() {
29:       return new itor();
30:    }
31:
32:    private class itor implements Iterator<Integer> {
33:       node curr = front;
34:
35:       public boolean hasNext() {
36:          return curr != null;
37:       }
38:
39:       public Integer next() {
40:          if (curr == null) throw new NoSuchElementException();
41:          Integer next = curr.linenr;
42:          curr = curr.link;
43:          return next;
44:       }
45:
46:       public void remove() {
47:          throw new UnsupportedOperationException();
48:       }
49:    }
50:
51: }
52:
```

```
 1: # $Id: Makefile,v 1.3 2012-01-19 19:14:44-08 - - $
 2:
 3: JAVASRC    = jxref.java listmap.java intqueue.java
 4: SOURCES    = ${JAVASRC} Makefile README
 5: MAINCLASS  = jxref
 6: CLASSES    = ${JAVASRC:.java=.class}
 7: JARCLASSES = ${CLASSES} \
 8:              intqueue\$$1.class listmap\$$1.class \
 9:              intqueue\$$itor.class listmap\$$itor.class \
10:              intqueue\$$node.class listmap\$$node.class
11: JARFILE    = jxref
12: LISTING    = ../asg2j-jxref.code.ps
13: SUBMITDIR  = cmps012b-wm.w12 asg2
14:
15: all : ${JARFILE}
16:
17: ${JARFILE} : ${CLASSES}
18:         echo Main-class: ${MAINCLASS} >Manifest
19:         jar cvfm ${JARFILE} Manifest ${JARCLASSES}
20:         chmod +x ${JARFILE}
21:
22: %.class : %.java
23:         - checksource $<
24:         - cid + $<
25:         javac $<
26:
27: clean :
28:         - rm ${JARCLASSES} Manifest
29:
30: spotless : clean
31:         - rm ${JARFILE}
32:
33: ci : ${SOURCES}
34:         - checksource ${SOURCES}
35:         cid + ${SOURCES}
36:
37: lis : ${SOURCES}
38:         mkpspdf ${LISTING} ${SOURCES}
39:
40: submit : ${SOURCES}
41:         submit ${SUBMITDIR} ${SOURCES}
42:
43: again : ${SOURCES}
44:         gmake --no-print-directory spotless ci all lis
45:
```

```
 1: This directory contains starter code for your project and a
 2: Makefile which can be used to build it.  Begin by copying this
 3: directory int your private volume before beginning work.
 4:
 5: The Perl program is not part of your project, but is a reference
 6: implementation.  Your program should produce the same output,
 7: except possibly for minor variations in the format of the error
 8: messages.
 9:
10: $Id: README,v 1.1 2010-04-13 13:21:40-07 - - $
```