```
 1: // $Id: jxref.java,v 1.9 2016-01-12 17:37:33-08 - - $
 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:    private static final String STDIN_FILENAME = "-";
14:    private static final String REGEX = "\\w+([-'.:/]\\w+)*";
15:    private static final Pattern PATTERN = Pattern.compile(REGEX);
16:
17:    private static void xref_file (String filename, Scanner file){
18:       misc.trace ("filename", filename);
19:       listmap map = new listmap();
20:       for (int linenr = 1; file.hasNextLine(); ++linenr) {
21:          String line = file.nextLine();
22:          misc.trace (filename, linenr, line);
23:          Matcher match = PATTERN.matcher (line);
24:          while (match.find()) {
25:             String word = match.group();
26:             misc.trace ("word", word);
27:             //FIXME
28:          }
29:       }
30:       for (Entry<String, intqueue> entry: map) {
31:          misc.trace ("STUB", entry,
32:                      entry.getKey(), entry.getValue());
33:          //FIXME
34:       }
35:    }
36:
```

```
37:
38:    // For each filename, open the file, cross reference it,
39:    // and print.
40:    private static void xref_filename (String filename) {
41:       if (filename.equals (STDIN_FILENAME)) {
42:          xref_file (filename, new Scanner (System.in));
43:       }else {
44:          try {
45:             Scanner file = new Scanner (new File (filename));
46:             xref_file (filename, file);
47:             file.close();
48:          }catch (IOException error) {
49:             misc.warn (error.getMessage());
50:          }
51:       }
52:    }
53:
54:    // Main function scans arguments to cross reference files.
55:    public static void main (String[] args) {
56:       if (args.length == 0) {
57:          xref_filename (STDIN_FILENAME);
58:       }else {
59:          for (String filename: args) {
60:             xref_filename (filename);
61:          }
62:       }
63:       exit (misc.exit_status);
64:    }
65:
66: }
67:
```

```
 1: // $Id: listmap.java,v 1.5 2013-10-16 17:10:32-07 - - $
 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 class node implements Entry<String,intqueue> {
11:       String key;
12:       intqueue queue = new intqueue();
13:       node link;
14:       public String getKey() {
15:          return key;
16:       }
17:       public intqueue getValue() {
18:          return queue;
19:       }
20:       public intqueue setValue (intqueue queue) {
21:          throw new UnsupportedOperationException();
22:       }
23:    }
24:    private node head = null;
25:
26:    public listmap() {
27:       // Not needed, since head defaults to null anyway.
28:    }
29:
30:    public void insert (String key, int linenr) {
31:       misc.trace ("insert", key, linenr);
32:       //FIXME
33:    }
34:
35:    public Iterator<Entry<String,intqueue>> iterator() {
36:       return new iterator();
37:    }
38:
```

```
39:
40:    private class iterator
41:            implements Iterator<Entry<String,intqueue>> {
42:       node curr = head;
43:
44:       public boolean hasNext() {
45:          return curr != null;
46:       }
47:
48:       public Entry<String,intqueue> next() {
49:          if (curr == null) throw new NoSuchElementException();
50:          node next = curr;
51:          curr = curr.link;
52:          return next;
53:       }
54:
55:       public void remove() {
56:          throw new UnsupportedOperationException();
57:       }
58:
59:    }
60:
61: }
```

```
 1: // $Id: intqueue.java,v 1.4 2013-10-16 17:10:32-07 - - $
 2:
 3: import java.util.Iterator;
 4: import java.util.NoSuchElementException;
 5:
 6: class intqueue implements Iterable<Integer> {
 7:
 8:    private class node {
 9:       int linenr;
10:       node link;
11:    }
12:    private int count = 0;
13:    private node front = null;
14:    private node rear = null;
15:
16:    public void insert (int number) {
17:       ++count;
18:       misc.trace (count);
19:       //FIXME
20:    }
21:
22:    public boolean empty() {
23:       return count == 0;
24:    }
25:
26:    public int getcount() {
27:       return count;
28:    }
29:
30:    public Iterator<Integer> iterator() {
31:       return new iterator();
32:    }
33:
34:    private class iterator implements Iterator<Integer> {
35:       node curr = front;
36:
37:       public boolean hasNext() {
38:          return curr != null;
39:       }
40:
41:       public Integer next() {
42:          if (curr == null) throw new NoSuchElementException();
43:          Integer next = curr.linenr;
44:          curr = curr.link;
45:          return next;
46:       }
47:
48:       public void remove() {
49:          throw new UnsupportedOperationException();
50:       }
51:    }
52:
53: }
54:
```

```
 1: // $Id: misc.java,v 1.7 2013-10-11 19:24:18-07 - - $
 2:
 3: import static java.lang.System.*;
 4:
 5: class misc {
 6:    public static final int EXIT_SUCCESS = 0;
 7:    public static final int EXIT_FAILURE = 1;
 8:    public static final String program_name =
 9:                   basename (getProperty ("java.class.path"));
10:    public static int exit_status = EXIT_SUCCESS;
11:
12:    // constructor - prevents instantiation: only static fns.
13:    private misc() {
14:       throw new UnsupportedOperationException();
15:    }
16:
17:    // basename - strips the dirname and returns the basename.
18:    //            See:  man -s 3c basename
19:    public static String basename (String pathname) {
20:       if (pathname == null || pathname.length() == 0) return ".";
21:       String[] paths = pathname.split ("/");
22:       return paths.length == 0 ? "." : paths[paths.length - 1];
23:    }
24:
25:    // trace - print a trace message to stderr
26:    public static void trace (Object... args) {
27:       StackTraceElement elt =
28:                    Thread.currentThread().getStackTrace()[2];
29:       err.printf ("%s[%d]", elt.getMethodName(),
30:                    elt.getLineNumber());
31:       for (Object arg: args) err.printf (": %s", arg);
32:       err.printf ("%n");
33:    }
34:
35:    // warn - print a warning and set exit status to failure.
36:    public static void warn (Object... args) {
37:       err.printf ("%s", program_name);
38:       for (Object arg: args) err.printf (": %s", arg);
39:       err.printf ("%n");
40:       exit_status = EXIT_FAILURE;
41:    }
42:
43:    // die - print a warning and exit program.
44:    public static void die (Object... args) {
45:       warn (args);
46:       exit (exit_status);
47:    }
48:
49: }
```

```
# $Id: Makefile,v 1.6 2015-01-26 12:39:24-08 - - $

JAVASRC    = jxref.java listmap.java intqueue.java misc.java
SOURCES    = ${JAVASRC} Makefile README
ALLSOURCES = ${SOURCES} pxref.perl
MAINCLASS  = jxref
CLASSES    = ${JAVASRC:.java=.class}
JARCLASSES = ${CLASSES} intqueue\$$*.class listmap\$$*.class
JARFILE    = jxref
LISTING    = Listing.ps
SUBMITDIR  = cmps012b-wm.w15 asg2

all : ${JARFILE}

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

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

clean :
        - rm -vf ${JARCLASSES} Manifest

spotless : clean
        - rm -vf ${JARFILE} ${LISTING} ${LISTING:.ps=.pdf}

ci : ${ALLSOURCES}
        - checksource ${ALLSOURCES}
        cid + ${ALLSOURCES}

lis : ${ALLSOURCES}
        mkpspdf -s12 ${LISTING} ${ALLSOURCES}

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

again : ${ALLSOURCES}
        gmake --no-print-directory spotless ci all lis
```

```
   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 2013-01-24 19:22:48-08 - - $
```

```
 1: #!/usr/bin/perl
 2: # $Id: pxref.perl,v 1.2 2013-10-11 19:24:18-07 - - $
 3: use strict;
 4: use warnings;
 5:
 6: $0 =~ s|^.*/||;
 7: my $exit_status = 0;
 8: END {exit $exit_status}
 9: sub note(@) {print STDERR "@_"};
10: $SIG{'__WARN__'} = sub {note @_; $exit_status = 1};
11: $SIG{'__DIE__'} = sub {warn @_; exit};
12:
13: my $sep = ":" x 32;
14: push @ARGV, "-" unless @ARGV;
15:
16: for my $filename (@ARGV) {
17:    my %xref;
18:    open my $file, "<$filename"
19:         or warn "$0: $filename: $!\n" and next;
20:    while (defined (my $line = <$file>)) {
21:       push @{$xref{$1}}, $.
22:            while $line =~ s|^.*?(\w+([-'.:/]\w+)*)||;
23:    }
24:    close $file;
25:    print "$sep\n$filename\n$sep\n";
26:    printf "%s [%d] %s\n", $_, @{$xref{$_}} + 0,
27:           join " ", @{$xref{$_}}
28:           for sort keys %xref;
29: }
30:
```