```
 1: // $Id: jxref.java,v 1.1 2014-01-16 17:44:47-08 - - $
 2:
 3: import java.io.*;
 4: import java.util.Scanner;
 5: import static java.lang.System.*;
 6:
 7: class jxref {
 8:    static final String STDIN_NAME = "-";
 9:
10:    static class printer implements visitor <String, queue <Integer>> {
11:       public void visit (String key, queue <Integer> value) {
12:          out.printf ("%s %s", key, value);
13:          for (int linenr: value) out.printf (" %d", linenr);
14:          out.printf ("%n");
15:       }
16:    }
17:
18:    static void xref_file (String filename, Scanner scan) {
19:       treemap <String, queue <Integer>> map =
20:             new treemap <String, queue <Integer>> ();
21:       for (int linenr = 1; scan.hasNextLine (); ++linenr) {
22:          for (String word: scan.nextLine().split ("\\W+")) {
23:             if (word.matches ("^\\d*$")) continue;
24:             out.printf ("%s: %d: %s%n", filename, linenr, word);
25:          }
26:       }
27:       visitor <String, queue <Integer>> print_fn = new printer ();
28:       map.do_visit (print_fn);
29:    }
30:
31:    public static void main (String[] args) {
32:       if (args.length == 0) {
33:          xref_file (STDIN_NAME, new Scanner (in));
34:       }else {
35:          for (int argi = 0; argi < args.length; ++argi) {
36:             String filename = args[argi];
37:             if (filename.equals (STDIN_NAME)) {
38:                xref_file (STDIN_NAME, new Scanner (in));
39:             }else {
40:                try {
41:                   Scanner scan = new Scanner (new File (filename));
42:                   xref_file (filename, scan);
43:                   scan.close ();
44:                }catch (IOException error) {
45:                   auxlib.warn (error.getMessage ());
46:                }
47:             }
48:          }
49:       }
50:       auxlib.exit ();
51:    }
52:
53: }
54:
```

```
 1: // $Id: auxlib.java,v 1.2 2014-02-07 17:06:33-08 - - $
 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:    //
46:    // Functions:
47:    //    whine   - prints a message with a given exit code.
48:    //    warn    - prints a stderr message and sets the exit code.
49:    //    die     - calls warn then exits.
50:    // Combinations of arguments:
51:    //    objname - name of the object to be printed (optional)
52:    //    message - message to be printed after the objname,
53:    //              either a Throwable or a String.
54:    //
55:    public static void whine (int exitval, Object... args) {
56:       exitvalue = exitval;
57:       err.printf ("%s", PROGNAME);
58:       for (Object argi : args) err.printf (": %s", argi);
59:       err.printf ("%n");
60:    }
61:    public static void warn (Object... args) {
62:       whine (EXIT_FAILURE, args);
63:    }
64:    public static void die (Object... args) {
65:       warn (args);
66:       exit ();
67:    }
68:
69:    //
70:    // usage_exit - prints a usage message and exits.
71:    //
72:    public static void usage_exit (String optsargs) {
73:       exitvalue = EXIT_FAILURE;
74:       err.printf ("Usage: %s %s%n", PROGNAME, optsargs);
75:       exit ();
76:    }
77:
78:    //
79:    // exit - calls exit with the appropriate code.
80:    //        This function should be called instead of returning
81:    //        from the main function.
82:    //
83:    public static void exit () {
84:       System.exit (exitvalue);
85:    }
86:
87:    //
88:    // identity - returns the default Object.toString value
89:    //            Useful for debugging.
90:    //
91:    public static String identity (Object object) {
92:       return object == null ? "(null)"
93:            : object.getClass().getName() + "@"
94:            + toHexString (identityHashCode (object));
95:    }
96:
97: }
```

```
 1: // $Id: treemap.java,v 1.1 2014-01-16 17:44:47-08 - - $
 2:
 3: import static java.lang.System.*;
 4:
 5: class treemap <key_t extends Comparable <key_t>, value_t> {
 6:
 7:    private class node {
 8:       key_t key;
 9:       value_t value;
10:       node left;
11:       node right;
12:    }
13:    private node root;
14:
15:    private void debug_dump_rec (node tree, int depth) {
16:       throw new UnsupportedOperationException ();
17:    }
18:
19:    private void do_visit_rec (visitor <key_t, value_t> visit_fn,
20:                               node tree) {
21:       throw new UnsupportedOperationException ();
22:    }
23:
24:    public value_t get (key_t key) {
25:       throw new UnsupportedOperationException ();
26:    }
27:
28:    public value_t put (key_t key, value_t value) {
29:       throw new UnsupportedOperationException ();
30:    }
31:
32:    public void debug_dump () {
33:       debug_dump_rec (root, 0);
34:    }
35:
36:    public void do_visit (visitor <key_t, value_t> visit_fn) {
37:       do_visit_rec (visit_fn, root);
38:    }
39:
40: }
```

```
 1: // $Id: queue.java,v 1.1 2014-01-16 17:44:47-08 - - $
 2:
 3: import java.util.Iterator;
 4: import java.util.NoSuchElementException;
 5:
 6: class queue <item_t> implements Iterable <item_t> {
 7:
 8:    private class node {
 9:       item_t item;
10:       node link;
11:    }
12:    private node head = null;
13:    private node tail = null;
14:
15:    public boolean isempty () {
16:       throw new RuntimeException ("Replace this with working code");
17:    }
18:
19:    public void insert (item_t newitem) {
20:       throw new RuntimeException ("Replace this with working code");
21:    }
22:
23:    public Iterator <item_t> iterator () {
24:       return new itor ();
25:    }
26:
27:    class itor implements Iterator <item_t> {
28:       node next = head;
29:       public boolean hasNext () {
30:          return next != null;
31:       }
32:       public item_t next () {
33:          if (! hasNext ()) throw new NoSuchElementException ();
34:          item_t result = next.item;
35:          next = next.link;
36:          return result;
37:       }
38:       public void remove () {
39:          throw new UnsupportedOperationException ();
40:       }
41:    }
42:
43: }
```

```
1: // $Id: visitor.java,v 1.1 2014-01-16 17:44:47-08 - - $
2:
3: interface visitor <key_t, value_t> {
4:    public void visit (key_t key, value_t value);
5: }
6:
```

```
 1: # $Id: Makefile,v 1.2 2014-01-16 17:46:07-08 - - $
 2:
 3: JAVASRC    = jxref.java auxlib.java treemap.java queue.java visitor.java
 4: SOURCES    = ${JAVASRC} Makefile
 5: ALLSOURCES = ${SOURCES}
 6: MAINCLASS  = jxref
 7: CLASSES    = ${patsubst %.java, %.class, ${JAVASRC}}
 8: INNCLASSES = jxref\$$printer.class treemap\$$node.class \
 9:              queue\$$itor.class queue\$$node.class
10: JARCLASSES = ${CLASSES} ${INNCLASSES}
11: JARFILE    = jxref
12: LISTING    = Listing.ps
13: SUBMITDIR  = cmps012b-wm.f10 asg3
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:         - rm Manifest
22:
23: %.class : %.java
24:         cid + $<
25:         javac -Xlint $<
26:
27: clean :
28:         - rm ${JARCLASSES} Manifest
29:
30: spotless : clean
31:         - rm ${JARFILE}
32:
33: ci : ${SOURCES}
34:         cid + ${SOURCES}
35:         checksource ${SOURCES}
36:
37: lis : ${SOURCES}
38:         mkpspdf ${LISTING} ${SOURCES}
39:
40: submit : ${SOURCES}
41:         submit ${SUBMITDIR} ${SOURCES}
42:
43: again :
44:         gmake --no-print-directory spotless ci all lis
45:
```