```
 1: // $Id: jxref.java,v 1.9 2012-10-31 17:06:20-07 - - $
 2:
 3: import java.io.*;
 4: import java.util.Iterator;
 5: import java.util.Scanner;
 6: import java.util.regex.Matcher;
 7: import java.util.regex.Pattern;
 8: import static java.lang.System.*;
 9:
10: class jxref {
11:    static final String WORD_REGEX =  "\\w+([-'.:/]\\w+)*";
12:    static final Pattern WORD_PATTERN = Pattern.compile (WORD_REGEX);
13:
14:    // Pseudo-typedef in Java, q.v., typedef in C.
15:    static class string_cqueue_map
16:                 extends treemap <String, counted_queue <Integer>> {
17:    }
18:
19:    static class print_visitor
20:                 implements visitor <String, counted_queue <Integer>> {
21:       boolean want_queue_printed;
22:       print_visitor (boolean count_only) {
23:          want_queue_printed = ! count_only;
24:       }
25:       public void visit (String key, counted_queue<Integer> queue) {
26:          out.printf ("%s [%d]", key, queue.count());
27:          if (want_queue_printed) {
28:             for (int line_nr: queue) out.printf (" %d", line_nr);
29:          }
30:          out.printf ("%n");
31:       }
32:    }
33:
```

```
34:
35:    static void scanfile (String filename, Scanner file, options opts) {
36:       out.printf ("filename = %s%n", filename);
37:       string_cqueue_map tree = new string_cqueue_map();
38:       for (int linenr = 1; file.hasNextLine(); ++linenr) {
39:          String line = file.nextLine();
40:          out.printf ("%s: %d: %s%n", filename, linenr, line);
41:          Matcher matcher = WORD_PATTERN.matcher (line);
42:          while (matcher.find()) {
43:             String word = matcher.group();
44:             counted_queue <Integer> queue = tree.get (word);
45:             out.printf ("word = %s%n", word);
46:          }
47:       }
48:       tree.visit_all (new print_visitor (opts.count_only));
49:    }
50:
51:    public static void main (String[] args) {
52:       options opts = new options (args);
53:       for (String filename : opts.filenames) {
54:          if (filename.equals ("-")) {
55:             scanfile ("-", new Scanner (in), opts);
56:          }else {
57:             try {
58:                Scanner scan = new Scanner (new File (filename));
59:                scanfile (filename, scan, opts);
60:                scan.close();
61:             }catch (IOException error) {
62:                messages.warn (error.getMessage());
63:             }
64:          }
65:       }
66:       exit (messages.exit_status);
67:    }
68:
69: }
```

```
1: // $Id: visitor.java,v 1.1 2012-10-31 13:12:50-07 - - $
2:
3: interface visitor <key_t, value_t> {
4:    public void visit (key_t key, value_t value);
5: }
6:
```

```
 1: // $Id: messages.java,v 1.1 2012-10-31 13:12:50-07 - - $
 2:
 3: import static java.lang.System.*;
 4:
 5: class messages {
 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:    //
13:    // constructor - prevents instantiation: only static fns allowed.
14:    //
15:    private messages() {
16:       throw new UnsupportedOperationException();
17:    }
18:
19:    //
20:    // basename - strips the dirname and returns only the basename.
21:    //            See:  man -s 3c basename
22:    //
23:    public static String basename (String pathname) {
24:       if (pathname == null || pathname.length () == 0) return ".";
25:       String[] paths = pathname.split ("/");
26:       for (int index = paths.length - 1; index >= 0; --index) {
27:          if (paths[index].length () > 0) return paths[index];
28:       }
29:       return "/";
30:    }
31:
32:    //
33:    // warn - print a warning and set exit status to failure.
34:    //
35:    public static void warn (Object... args) {
36:       exit_status = EXIT_FAILURE;
37:       err.printf ("%s", program_name);
38:       for (Object arg: args) err.printf (": %s", arg);
39:       err.printf ("%n");
40:    }
41:
42:    //
43:    // die - print a warning and exit program.
44:    //
45:    public static void die (Object... args) {
46:       warn (args);
47:       exit (exit_status);
48:    }
49:
50: }
```

```
 1: // $Id: options.java,v 1.1 2012-10-31 13:12:50-07 - - $
 2:
 3: class options {
 4:    boolean count_only = false;
 5:    boolean dump_tree = false;
 6:    boolean fold_cases = false;
 7:    String[] filenames = null;
 8:
 9:    options (String[] args) {
10:    }
11:
12: }
```

```
 1: // $Id: queue.java,v 1.2 2012-10-31 17:11:14-07 - - $
 2:
 3: import java.util.Iterator;
 4: import java.util.NoSuchElementException;
 5:
 6: //
 7: // Linked implementation of a generic queue with iteration.
 8: //
 9:
10: class queue<item_t> implements Iterable<item_t> {
11:    private node head = null;
12:    private node tail = null;
13:
14:    private class node {
15:       item_t item;
16:       node link = null;
17:       node (item_t item_) { item = item_; }
18:    }
19:
20:    public boolean isempty () {
21:       return head == null;
22:    }
23:
24:    public void insert (item_t item) {
25:       node tmp = new node (item);
26:       if (tail == null) head = tmp;
27:                     else tail.link = tmp;
28:       tail = tmp;
29:    }
30:
31:    public item_t remove() {
32:       if (isempty()) throw new NoSuchElementException();
33:       item_t result = head.item;
34:       head = head.link;
35:       if (head == null) tail = null;
36:       return result;
37:    }
38:
39:    public Iterator<item_t> iterator() {
40:       return new iterator();
41:    }
42:
43:    protected class iterator implements Iterator<item_t> {
44:       node nextnode = head;
45:
46:       public boolean hasNext() {
47:          return nextnode != null;
48:       }
49:
50:       public item_t next() {
51:          if (nextnode == null) throw new NoSuchElementException();
52:          item_t nextitem = nextnode.item;
53:          nextnode = nextnode.link;
54:          return nextitem;
55:       }
56:
57:       public void remove() {
58:          throw new UnsupportedOperationException();
59:       }
60:    }
61:
62: }
```

```
 1: // $Id: counted_queue.java,v 1.2 2012-10-31 17:12:51-07 - - $
 2:
 3: class counted_queue<item_t> extends queue<item_t>
 4:                             implements Iterable<item_t> {
 5:    private int count = 0;
 6:
 7:    public void insert (item_t item) {
 8:       super.insert (item);
 9:       ++count;
10:    }
11:
12:    public item_t remove() {
13:       item_t result = super.remove();
14:       --count;
15:       return result;
16:    }
17:
18:    public int count() {
19:       return count;
20:    }
21:
22:    public String toString() {
23:       return "[" + count + ", " + super.toString() + "]";
24:    }
25:
26: }
```

```
 1: // $Id: treemap.java,v 1.1 2012-10-31 13:12:50-07 - - $
 2:
 3: import java.io.*;
 4: import static java.lang.System.*;
 5:
 6: class treemap <key_t extends Comparable<key_t>, value_t> {
 7:
 8:    private class bstree {
 9:       key_t key;
10:       value_t value;
11:       bstree left;
12:       bstree right;
13:    }
14:    private bstree root = null;
15:
16:    public value_t put (key_t key, value_t value) {
17:       int cmp = root.key.compareTo (key);
18:       throw new UnsupportedOperationException ();
19:    }
20:
21:    public value_t get (key_t key) {
22:       throw new UnsupportedOperationException ();
23:    }
24:
25:    public value_t remove (key_t key) {
26:       throw new UnsupportedOperationException ();
27:    }
28:
29:    public void visit_all (visitor <key_t, value_t> visitor_fn) {
30:       visit_all_inorder (root, visitor_fn);
31:    }
32:
33:    private void visit_all_inorder (bstree tree,
34:                    visitor <key_t, value_t> visitor_fn) {
35:       visitor_fn.visit (tree.key, tree.value);
36:       throw new UnsupportedOperationException ();
37:    }
38:
39:    public void debug_dump () {
40:       debug_dump_inorder (root, 0);
41:    }
42:
43:    private void debug_dump_inorder (bstree tree, int depth) {
44:       if (tree == null) return;
45:       debug_dump_inorder (tree.left, depth + 1);
46:       out.printf ("%*s%d: %s => %s%n",
47:                    depth * 3, "", depth, tree.key, tree.value);
48:       debug_dump_inorder (tree.right, depth + 1);
49:    }
50:
51: }
```

```
 1: # $Id: Makefile,v 1.3 2012-10-31 19:49:33-07 - - $
 2:
 3: JAVASRC    = jxref.java visitor.java messages.java options.java \
 4:              queue.java counted_queue.java treemap.java
 5: SOURCES    = ${JAVASRC} Makefile README
 6: ALLSOURCES = ${SOURCES} pxref.perl
 7: MAINCLASS  = jxref
 8: CLASSES    = ${JAVASRC:%.java=%.class}
 9: INNCLASSES = ${CLASSES:%.class=%\\$$*.class}
10: #LS_INNER   = `(ls ${INNCLASSES} 2>/dev/null || true)`
11: JARCLASSES = ${CLASSES} ${LS_INNER}
12: JARFILE    = jxref
13: LISTING    = ../asg3j-jxref.code.ps
14:
15: all : ${JARFILE}
16:         - checksource ${SOURCES}
17:
18: ${JARFILE} : ${CLASSES}
19:         echo Main-class: ${MAINCLASS} >Manifest
20:         jar cvfm ${JARFILE} Manifest ${JARCLASSES}
21:         chmod +x ${JARFILE}
22:         - rm Manifest
23:
24: %.class : %.java
25:         cid + $<
26:         javac -Xlint $<
27:
28: clean :
29:         - rm ${JARCLASSES} Manifest
30:
31: spotless : clean
32:         - rm ${JARFILE}
33:
34: ci : ${ALLSOURCES}
35:         cid + ${ALLSOURCES}
36:
37: lis : ${ALLSOURCES}
38:         mkpspdf ${LISTING} ${ALLSOURCES}
39:
40: again : ${ALLSOURCES}
41:         make --no-print-directory spotless ci all lis
42:
```

```
1: $Id: README,v 1.1 2012-10-31 13:12:50-07 - - $
```

```
 1: #!/usr/bin/perl
 2: # $Id: pxref.perl,v 1.4 2012-10-31 14:45:07-07 - - $
 3: use strict;
 4: use warnings;
 5: use Getopt::Std;
 6:
 7: $0 =~ s|^(.*/)?([^/]+)/*$|$2|;
 8: my $exit_status = 0;
 9: END {exit $exit_status}
10: sub note (@) {print STDERR "$0: @_"}
11: $SIG{'__WARN__'} = sub {note @_; $exit_status = 1};
12: $SIG{'__DIE__'} = sub {warn @_; exit};
13:
14: my $word_regex = qr(\w+([-'.:/]\w+)*);
15: my %opts;
16: getopts ("cdf", \%opts);
17: warn "-d option not supported in perl version\n" if $opts{'d'};
18:
19: for my $filename (@ARGV ? @ARGV : "-") {
20:    my %xref;
21:    open my $file, "<$filename" or do warn "$filename: $!\n" and next;
22:    my $sep = "\n" . ":" x 65 . "\n";
23:    print "$sep$filename$sep\n";
24:    while (defined (my $line = <$file>)) {
25:       push @{$xref{$opts{'f'} ? lc $& : $&}}, $.
26:            while $line =~ s|$word_regex||;
27:    }
28:    close $file;
29:    for my $word (sort keys %xref) {
30:       my $list = $xref{$word};
31:       printf "%s [%d]", $word, @$list + 0;
32:       print " @$list" unless $opts{'c'};
33:       print "\n";
34:    }
35: }
36:
```