```
 1: // $Id: airport.java,v 1.2 2013-01-31 17:00:19-08 - - $
 2:
 3: //
 4: // Starter code for the airport utility.
 5: //
 6:
 7: import java.io.*;
 8: import java.util.Scanner;
 9: import static java.lang.System.*;
10:
11: class airport {
12:    private static final String STDIN_FILENAME = "-";
13:
14:    public static treemap load_database (String database_name) {
15:       treemap tree = new treemap ();
16:       try {
17:          Scanner database = new Scanner (new File (database_name));
18:          for (int linenr = 1; database.hasNextLine(); ++linenr) {
19:             String line = database.nextLine();
20:             if (line.matches ("^\\s*(#.*)?$")) continue;
21:             String[] keyvalue = line.split (":");
22:             if (keyvalue.length != 2) {
23:                misc.warn (database_name, linenr, "invalid line");
24:                continue;
25:             }
26:             tree.put (keyvalue[0], keyvalue[1]);
27:          }
28:          database.close();
29:       }catch (IOException error) {
30:          misc.warn (error.getMessage());
31:       }
32:       return tree;
33:    }
34:
35:    public static void main (String[] args) {
36:       treemap tree = load_database (args[0]);
37:       Scanner stdin = new Scanner (in);
38:       while (stdin.hasNextLine()) {
39:          String airport = stdin.nextLine().toUpperCase().trim();
40:          String airport_name = tree.get (airport);
41:          if (airport_name == null) {
42:             out.printf ("%s: no such airport%n", airport);
43:          }else {
44:             out.printf ("%s = %s%n", airport, airport_name);
45:          }
46:       }
47:       tree.debug_tree ();
48:       exit (misc.exit_status);
49:    }
50:
51: }
```

```
 1: // $Id: treemap.java,v 1.1 2012-02-07 15:43:17-08 - - $
 2:
 3: // Development version of treemap.
 4: // To be deleted and replaced by an actual implementation that
 5: // does *NOT* use java.util.TreeMap.
 6:
 7: import static java.lang.System.*;
 8:
 9: class treemap {
10:
11:    class tree {
12:       String key;
13:       String value;
14:       tree left;
15:       tree right;
16:    }
17:    tree root = null;
18:
19:    java.util.TreeMap <String, String> tree
20:          = new java.util.TreeMap <String, String> ();
21:
22:    public String get (String key) {
23:       return tree.get (key);
24:    }
25:
26:    public String put (String key, String value) {
27:       return tree.put (key, value);
28:    }
29:
30:    public void debug_tree () {
31:       debug_tree_recur (root, 0);
32:    }
33:
34:    private void debug_tree_recur (tree node, int depth) {
35:    }
36:
37: }
```

```
 1: // $Id: misc.java,v 1.1 2013-01-31 17:00:19-08 - - $
 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 allowed.
13:    private misc() {
14:       throw new UnsupportedOperationException();
15:    }
16:
17:    // basename - strips the dirname and returns only 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 = Thread.currentThread().getStackTrace()[2];
28:       err.printf ("%s[%d]", elt.getMethodName(), elt.getLineNumber());
29:       for (Object arg: args) err.printf (": %s", arg);
30:       err.printf ("%n");
31:    }
32:
33:    // warn - print a warning and set exit status to failure.
34:    public static void warn (Object... args) {
35:       err.printf ("%s", program_name);
36:       for (Object arg: args) err.printf (": %s", arg);
37:       err.printf ("%n");
38:       exit_status = EXIT_FAILURE;
39:    }
40:
41:    // die - print a warning and exit program.
42:    public static void die (Object... args) {
43:       warn (args);
44:       exit (exit_status);
45:    }
46:
47: }
```

```
 1: # $Id: Makefile,v 1.5 2015-01-30 15:23:45-08 - - $
 2:
 3: JAVASRC    = airport.java treemap.java misc.java
 4: SOURCES    = ${JAVASRC} Makefile README
 5: MAINCLASS  = airport
 6: CLASSES    = ${JAVASRC:.java=.class}
 7: JARCLASSES = ${CLASSES} treemap\$$tree.class
 8: JARFILE    = airport
 9: LISTING    = Listing.ps
10:
11: all : ${JARFILE}
12:
13: ${JARFILE} : ${CLASSES}
14:         echo Main-class: ${MAINCLASS} >Manifest
15:         jar cvfm ${JARFILE} Manifest ${JARCLASSES}
16:         - rm Manifest
17:         chmod +x ${JARFILE}
18:
19: %.class : %.java
20:         - cid $<
21:         javac $<
22:
23: clean :
24:         - rm ${JARCLASSES}
25:
26: spotless : clean
27:         - rm ${JARFILE}
28:
29: ci : ${SOURCES}
30:         - checksource ${SOURCES}
31:         cid + ${SOURCES}
32:
33: lis : ${SOURCES}
34:         mkpspdf ${LISTING} ${SOURCES}
35:
36: submit : ${SOURCES}
37:         submit cmps012b-wm.w15 asg3 ${SOURCES}
38:
39: again : ${SOURCES}
40:         gmake --no-print-directory spotless ci all lis
41:
```

```
1: $Id: README,v 1.1 2012-02-07 15:43:17-08 - - $
2: Replace this name with your name and username
3: and that of your partner if your are doing pair programming.
```