```
 1: // $Id: airport.java,v 1.1 2012-02-07 15:43:17-08 - - $
 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:    // Static program constants.
13:    private static final String STDIN_FILENAME = "-";
14:    private static final String JARNAME = get_jarname();
15:    private static final int EXIT_SUCCESS = 0;
16:    private static final int EXIT_FAILURE = 1;
17:
18:    // Static exit status variable.
19:    private static int exit_status = EXIT_SUCCESS;
20:
21:    // A basename is the final component of a pathname.
22:    // If a java program is run from a jar, the classpath is the
23:    // pathname of the jar.
24:    private static String get_jarname() {
25:       String jarpath = getProperty ("java.class.path");
26:       int lastslash = jarpath.lastIndexOf ('/');
27:       if (lastslash < 0) return jarpath;
28:       return jarpath.substring (lastslash + 1);
29:    }
30:
```

```
31:
32:
33:    public static treemap load_database (String database_name) {
34:       treemap tree = new treemap ();
35:       try {
36:          Scanner database = new Scanner (new File (database_name));
37:          for (int linenr = 1; database.hasNextLine(); ++linenr) {
38:             String line = database.nextLine();
39:             if (line.matches ("^\\s*(#.*)?$")) continue;
40:             String[] keyvalue = line.split (":");
41:             if (keyvalue.length != 2) {
42:                exit_status = EXIT_FAILURE;
43:                err.printf ("%s: %s:%d: invalid line",
44:                            JARNAME, database_name, linenr);
45:                continue;
46:             }
47:             tree.put (keyvalue[0], keyvalue[1]);
48:          }
49:          database.close();
50:       }catch (IOException error) {
51:          exit_status = EXIT_FAILURE;
52:          err.printf ("%s: %s%n", JARNAME, error.getMessage());
53:       }
54:       return tree;
55:    }
56:
57:    public static void main (String[] args) {
58:       treemap tree = load_database (args[0]);
59:       Scanner stdin = new Scanner (in);
60:       while (stdin.hasNextLine()) {
61:          String airport = stdin.nextLine().toUpperCase().trim();
62:          String airport_name = tree.get (airport);
63:          if (airport_name == null) {
64:             out.printf ("%s: no such airport%n", airport);
65:          }else {
66:             out.printf ("%s = %s%n", airport, airport_name);
67:          }
68:       }
69:       tree.debug_tree ();
70:       exit (exit_status);
71:    }
72:
73: }
```

```
 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: Makefile,v 1.1 2012-02-07 15:43:17-08 - - $
 2:
 3: JAVASRC    = airport.java treemap.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    = ../asg3j-airport.code.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:         - checksource $<
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.w12 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.
```