```
 1  // joi/3/textfiles/TextFile.java
 2  //
 3  //
 4  // Copyright 2003 Bill Campbell and Ethan Bolker
 5
 6  import java.util.Date;
 7
 8  /**
 9   * A TextFile mimics the sort of text file that one finds
10   * on a computer's file system.  It has an owner,
11   * a create date (when the file was created),
12   * a modification date (when the file was last modified),
13   * and String contents.
14   *
15   * @version 3
16   */
17
18  public class TextFile
19  {
20      // Private Implementation
21
22      private String owner;       // Who owns the file.
23      private Date   createDate;  // When the file was created.
24      private Date   modDate;     // When the file was last modified.
25      private String contents;    // The text stored in the file.
26
27      // Public Interface
28
29      /**
30       * Construct a new TextFile with given owner and
31       * contents; set the creation and modification dates.
32       *
33       * @param owner the user who owns the file.
34       * @param contents the file's initial contents.
35       */
36
37      public TextFile( String owner, String contents )
38      {
39          this.owner    = owner;
40          this.contents = contents;
41          createDate    = new Date(); // date and time now
42          modDate       = createDate;
43      }
44
45      /**
46       * Replace the contents of the file.
47       *
48       * @param contents the new contents.
49       */
50
51      public void setContents( String contents )
52      {
53          this.contents = contents;
54          modDate = new Date();
55      }
56
```



```
57      /**
58       * The contents of a file.
59       *
60       * @return String contents of the file.
61       */
62
63      public String getContents()
64      {
65          return contents;
66      }
67
68      /**
69       * Append text to the end of the file.
70       *
71       * @param  text the text to be appended.
72       */
73
74      public void append( String text )
75      {
76          this.setContents( contents + text );
77      }
78
79      /**
80       * Append a new line of text to the end of the file.
81       *
82       * @param  text the text to be appended.
83       */
84
85      public void appendLine( String text )
86      {
87          this.setContents(contents + '\n' + text);
88      }
89
90      /**
91       * The size of a file.
92       *
93       * @return the integer size of the file
94       *  (the number of characters in its String contents)
95       */
96
97      public int getSize()
98      {
99          int charCount;
100         charCount = contents.length();
101         return charCount;
102     }
103
104     /**
105      * The data and time of the file's creation.
106      *
107      * @return the file's creation date and time.
108      */
109
110     public String getCreateDate()
111     {
112         return createDate.toString();
```

```
113      }
114
115      /**
116       * The date and time of the file's last modification.
117       *
118       * @return the date and time of the file's last modification.
119       */
120
121      public String getModDate()
122      {
123          return modDate.toString();
124      }
125
126      /**
127       * The file's owner.
128       *
129       * @return the owner of the file.
130       */
131
132      public String getOwner()
133      {
134          return owner;
135      }
136
137      /**
138       * A definition of main(), used only for testing this class.
139       *
140       * Executing
141       * <pre>
142       *     %> java TextFile
143       * </pre>
144       * produces the output:
145       * <pre>
146       * TextFile myTextFile contains 13 characters.
147       * Created by bill, Sat Dec 29 14:02:37 EST 2001
148       * Hello, world.
149       *
150       * append new line "How are you today?"
151       * Hello, world.
152       * How are you today?
153       * TextFile myTextFile contains 32 characters.
154       * Modified Sat Dec 29 14:02:38 EST 2001
155       * </pre>
156       */
157
158      public static void main( String[] args )
159      {
160          Terminal terminal = new Terminal();
161          TextFile myTextFile = new TextFile( "bill", "Hello, world." );
162
163          terminal.println( "TextFile myTextFile contains " +
164                            myTextFile.getSize() + " characters.");
165          terminal.println( "Created by " +  myTextFile.getOwner() +
166                            ", " +
167                            myTextFile.getCreateDate() );
168          terminal.println( myTextFile.getContents() );
```



```
169          terminal.println();
170
171          terminal.println( "append new line \"How are you today?\"" );
172          myTextFile.appendLine( "How are you today?" );
173          terminal.println( myTextFile.getContents() );
174          terminal.println( "TextFile myTextFile contains " +
175                            myTextFile.getSize() + " characters." );
176          terminal.println( "Modified "  +
177                            myTextFile.getModDate() );
178      }
179  }
```

```
 1  // joi/4/textfiles/Directory.java
 2  //
 3  //
 4  // Copyright 2003 Ethan Bolker and Bill Campbell
 5
 6  // This draft contains just stubs for the methods.
 7  // You can invoke them all, but none will do anything.
 8
 9  /**
10   * Directory of TextFiles.
11   *
12   * @version 4
13   */
14
15  public class Directory
16  {
17      /**
18       * Construct a Directory.
19       */
20
21      public Directory( )
22      {
23      }
24
25      /**
26       * The size of a directory is the number of TextFiles it contains.
27       *
28       * @return the number of TextFiles.
29       */
30
31      public int getSize()
32      {
33          return 0;
34      }
35
36      /**
37       * Add a TextFile to this Directory. Overwrite if a TextFile
38       * of that name already exists.
39       *
40       * @param name the name under which this TextFile is added.
41       * @param afile the TextFile to add.
42       */
43
44      public void addTextFile(String name, TextFile afile)
45      {
46      }
47
48      /**
49       * Get a TextFile in this Directory, by name .
50       *
51       * @param filename the name of the TextFile to find.
52       * @return the TextFile found, null if none.
53       */
54
55      public TextFile retrieveTextFile( String filename )
56      {
```



```
57          return null;
58      }
59
60      /**
61       * Get the contents of this Directory as an array of
62       * the file names, each of which is a String.
63       *
64       * @return the array of names.
65       */
66
67      public String[] getFileNames()
68      {
69          // pseudocode for an implementation:
70          // declare an array of String
71          // create that array with as many spaces as there
72          //   are TextFile's in this Directory
73          // loop through the keys of the TreeMap of TextFiles,
74          // adding each String key to the array
75          // return the array
76
77          // the next line is there because we have to return
78          // _something_ in order to satisfy the compiler
79          return new String[0];
80      }
81
82      /**
83       *  main, for unit testing.
84       *
85       * The command
86       * <pre>
87       *   java Directory
88       * </pre>
89       * should produce output
90       * <pre>
91       * bill     17      Sun Jan 06 19:40:13 EST 2003    diary
92       * eb       12      Sun Jan 06 19:40:13 EST 2003    greeting
93       * </pre>
94       * (with current dates, of course).
95       */
96
97      public static void main( String[] args )
98      {
99          Directory dir = new Directory();
100         dir.addTextFile("greeting", new TextFile("eb", "Hello, world"));
101         dir.addTextFile("diary", new TextFile("bill", "Writing Directory"
102         // now list TextFiles in dir to get output specified
103     }
104 }
```