```
 1  // joi/7/juno/Directory.java
 2  //
 3  //
 4  // Copyright 2003 Ethan Bolker and Bill Campbell
 5
 6  import java.util.*;
 7
 8  /**
 9   * Directory of JFiles.
10   *
11   * A Directory is a JFile that maintains a
12   * table of the JFiles it contains.
13   *
14   * @version 7
15   */
16
17  public class Directory extends JFile
18  {
19      private TreeMap jfiles;  // table for JFiles in this Directory
20
21      /**
22       * Construct a Directory.
23       *
24       * @param name the name for this Directory (in its parent Directory)
25       * @param creator the owner of this new Directory
26       * @param parent the Directory in which this Directory lives.
27       */
28
29      public Directory( String name, User creator, Directory parent)
30      {
31          super( name, creator, parent );
32          jfiles = new TreeMap();
33      }
34
35      /**
36       * The size of a Directory is the number of JFiles it contains.
37       *
38       * @return the Directory's size.
39       */
40
41      public int getSize()
42      {
43          return jfiles.size();
44      }
45
46      /**
47       * Suffix used for printing Directory names;
48       * we define it as the (system dependent)
49       * name separator used in path names.
50       *
51       * @return the suffix for Directory names.
52       */
53
54      public String getSuffix()
55      {
56          return  JFile.separator;
```



```
57      }
58
59      /**
60       * Add a JFile to this Directory. Overwrite if a JFile
61       * of that name already exists.
62       *
63       * @param name the name under which this JFile is added.
64       * @param afile the JFile to add.
65       */
66
67      public void addJFile(String name, JFile afile)
68      {
69          jfiles.put( name, afile );
70          setModDate();
71      }
72
73      /**
74       * Get a JFile in this Directory, by name .
75       *
76       * @param filename the name of the JFile to find.
77       * @return the JFile found.
78       */
79
80      public JFile retrieveJFile( String filename )
81      {
82          JFile aFile = (JFile)jfiles.get( filename );
83          return aFile;
84      }
85
86      /**
87       * Remove a JFile in this Directory, by name .
88       *
89       * @param filename the name of the JFile to remove
90       */
91
92      public void removeJFile( String filename )
93      {
94          jfiles.remove( filename );
95      }
96
97      /**
98       * Get the contents of this Directory as an array of
99       * the file names, each of which is a String.
100      *
101      * @return the array of names.
102      */
103
104     public String[] getFileNames()
105     {
106         return (String[])jfiles.keySet().toArray( new String[0] );
107     }
108 }
```