```
 1  // joi/7/jfiles/JFile.java
 2  //
 3  //
 4  // Copyright 2003 Bill Campbell and Ethan Bolker
 5
 6  import java.util.Date;
 7  import java.io.File;
 8
 9  /**
10   * A JFile object models a file in a hierarchical file system.
11   * <p>
12   * Extend this abstract class to create particular kinds of JFiles,
13   * e.g.:<br>
14   * Directory  -
15   *    a JFile that maintains a list of the files it contains.<br>
16   * TextFile  -
17   *    a JFile containing text you might want to read.<br>
18   *
19   * @see Directory
20   * @see TextFile
21   *
22   * @version 7
23   */
24
25  public abstract class JFile
26  {
27      /**
28       * The separator used in pathnames.
29       */
30
31      public static final String separator = File.separator;
32
33      private String    name;       // a JFile knows its name
34      private User      owner;      // the owner of this file
35      private Date      createDate; // when this file was created
36      private Date      modDate;    // when this file was last modified
37      private Directory parent;     // the Directory containing this file
38
39      /**
40       * Construct a new JFile, set owner, parent, creation and
41       * modification dates. Add this to parent (unless this is the
42       * root Directory).
43       *
44       * @param name     the name for this file (in its parent directory).
45       * @param creator  the owner of this new file.
46       * @param parent   the Directory in which this file lives.
47       */
48
49      protected JFile( String name, User creator, Directory parent )
50      {
51          this.name   = name;
52          this.owner  = creator;
53          this.parent = parent;
54          if (parent != null) {
55              parent.addJFile( name, this );
56          }
```



```
57          createDate  = modDate = new Date(); // set dates to now
58      }
59
60      /**
61       * The name of the file.
62       *
63       * @return the file's name.
64       */
65
66      public String getName()
67      {
68          return name;
69      }
70
71      /**
72       * The full path to this file.
73       *
74       * @return the path name.
75       */
76
77      public String getPathName()
78      {
79          if (this.isRoot()) {
80              return separator;
81          }
82          if (parent.isRoot()) {
83              return separator + getName();
84          }
85          return parent.getPathName() + separator + getName();
86      }
87
88      /**
89       * The size of the JFile
90       * (as defined by the child class)..
91       *
92       * @return the size.
93       */
94
95      public abstract int getSize();
96
97      /**
98       * Suffix used for printing file names
99       * (as defined by the child class).
100      *
101      * @return the file's suffix.
102      */
103
104     public abstract String getSuffix();
105
106     /**
107      * Set the owner for this file.
108      *
109      * @param owner the new owner.
110      */
111
112     public void setOwner( User owner )
```

```
113      {
114          this.owner = owner;
115      }
116
117      /**
118       * The file's owner.
119       *
120       * @return the owner of the file.
121       */
122
123      public User getOwner()
124      {
125          return owner;
126      }
127
128      /**
129       * The date and time of the file's creation.
130       *
131       * @return the file's creation date and time.
132       */
133
134      public String getCreateDate()
135      {
136          return createDate.toString();
137      }
138
139      /**
140       * Set the modification date to "now".
141       */
142
143      protected void setModDate()
144      {
145          modDate = new Date();
146      }
147
148      /**
149       * The date and time of the file's last modification.
150       *
151       * @return the date and time of the file's last modification.
152       */
153
154      public String getModDate()
155      {
156          return modDate.toString();
157      }
158
159      /**
160       * The Directory containing this file.
161       *
162       * @return the parent directory.
163       */
164
165      public Directory getParent()
166      {
167          return parent;
168      }
```



```
169
170      /**
171       * A JFile whose parent is null is defined to be the root
172       * (of a tree).
173       *
174       * @return true when this JFile is the root.
175       */
176
177      public boolean isRoot()
178      {
179          return (parent == null);
180      }
181
182      /**
183       * How a JFile represents itself as a String.
184       * That is,
185       * <pre>
186       *    owner   size    modDate    name+suffix
187       * </pre>
188       *
189       * @return the String representation.
190       */
191
192      public String toString()
193      {
194          return getOwner() + "\t" +
195              getSize()  + "\t" +
196              getModDate()  + "\t" +
197              getName() + getSuffix();
198      }
199  }
```