```java
1    // joi/6/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 6
23    */
24   public abstract class JFile
25   {
26       /**
27        * The separator used in pathnames.
28        */
29       public static final String separator = File.separator;
30
31       private String     name;       // a JFile knows its name
32       private User        owner;      // the owner of this file
33       private Date        createDate; // when this file was created
34       private Date        modDate;    // when this file was last modified
35       private Directory parent;       // the Directory containing this file
36
37       /**
38        * Construct a new JFile, set owner, parent, creation and
39        * modification dates. Add this to parent (unless this is the
40        * root Directory).
41        *
42        * @param name      the name for this file (in its parent directory).
43        * @param creator   the owner of this new file.
44        * @param parent    the Directory in which this file lives.
45        */
46       protected JFile( String name, User creator, Directory parent )
47       {
48           this.name  = name;
49           this.owner = creator;
50           this.parent = parent;
51           if (parent != null) {
52               parent.addFile( name, this );
53           }
```

```java
54           createDate = modDate = new Date(); // set dates to now
55       }
56
57       /**
58        * The name of the file.
59        *
60        * @return the file's name.
61        */
62       public String getName()
63       {
64           return name;
65       }
66
67       /**
68        * The full path to this file.
69        *
70        * @return the path name.
71        */
72       public String getPathName()
73       {
74           if (this.isRoot()) {
75               return separator;
76           }
77           if (parent.isRoot()) {
78               return separator + getName();
79           }
80           return parent.getPathName() + separator + getName();
81       }
82
83       /**
84        * The size of the JFile
85        * (as defined by the child class)..
86        *
87        * @return the size.
88        */
89       public abstract int getSize();
90
91       /**
92        * Suffix used for printing file names
93        * (as defined by the child class).
94        *
95        * @return the file's suffix.
96        */
97       public abstract String getSuffix();
98
99       /**
100       * Set the owner for this file.
101       *
102       * @param owner the new owner.
103       */
104      public void setOwner( User owner )
```

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

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