

```

1 // fo1/10/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 10
23 */
24
25 public abstract class JFile
26     implements java.io.Serializable
27 {
28     /**
29      * The separator used in pathnames.
30      */
31
32     public static final String separator = File.separator;
33
34     private String name; // a JFile knows its name
35     private User owner; // the owner of this file
36     private Date createDate; // when this file was created
37     private Date modDate; // when this file was last modified
38     private Directory parent; // the Directory containing this file
39
40     /**
41      * Construct a new JFile, set owner, parent, creation and
42      * modification dates. Add this to parent (unless this is the
43      * root Directory).
44      *
45      * @param name the name for this file (in its parent directory).
46      * @param creator the owner of this new file.
47      * @param parent the Directory in which this file lives.
48      */
49
50     protected JFile( String name, User creator, Directory parent )
51     {
52         this.name = name;
53         this.owner = creator;
54         this.parent = parent;
55         if (parent != null) {
56             parent.addJFile( name, this );

```

```

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

```

```

113 public void setOwner( User owner )
114 {
115     this.owner = owner;
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 /**
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
178 public boolean isRoot()
179 {
180     return (parent == null);
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
193 public String toString()
194 {
195     return getOwner() + "\t" +
196         getSize() + "\t" +
197         getModDate() + "\t" +
198         getName() + getSuffix();
199 }
200 }

```