```
// joi/10/jfiles/JFile.java
//
//
// Copyright 2003 Bill Campbell and Ethan Bolker

import java.util.Date;
import java.io.File;

/**
 * A JFile object models a file in a hierarchical file system.
 * <p>
 * Extend this abstract class to create particular kinds of JFiles,
 * e.g.:<br>
 * Directory  -
 *    a JFile that maintains a list of the files it contains.<br>
 * TextFile  -
 *    a JFile containing text you might want to read.<br>
 *
 * @see Directory
 * @see TextFile
 *
 * @version 10
 */

public abstract class JFile
    implements java.io.Serializable
{
    /**
     * The separator used in pathnames.
     */

    public static final String separator = File.separator;

    private String    name;       // a JFile knows its name
    private User      owner;      // the owner of this file
    private Date      createDate; // when this file was created
    private Date      modDate;    // when this file was last modified
    private Directory parent;     // the Directory containing this file

    /**
     * Construct a new JFile, set owner, parent, creation and
     * modification dates. Add this to parent (unless this is the
     * root Directory).
     *
     * @param name     the name for this file (in its parent directory).
     * @param creator  the owner of this new file.
     * @param parent   the Directory in which this file lives.
     */

    protected JFile( String name, User creator, Directory parent )
    {
        this.name   = name;
        this.owner  = creator;
        this.parent = parent;
        if (parent != null) {
            parent.addJFile( name, this );
```



```
        }
        createDate  = modDate = new Date(); // set dates to now
    }

    /**
     * The name of the file.
     *
     * @return the file's name.
     */

    public String getName()
    {
        return name;
    }

    /**
     * The full path to this file.
     *
     * @return the path name.
     */

    public String getPathName()
    {
        if (this.isRoot()) {
            return separator;
        }
        if (parent.isRoot()) {
            return separator + getName();
        }
        return parent.getPathName() + separator + getName();
    }

    /**
     * The size of the JFile
     * (as defined by the child class)..
     *
     * @return the size.
     */

    public abstract int getSize();

    /**
     * Suffix used for printing file names
     * (as defined by the child class).
     *
     * @return the file's suffix.
     */

    public abstract String getSuffix();

    /**
     * Set the owner for this file.
     *
     * @param owner the new owner.
     */

```

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



```
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  }
```