```
Loading vc-cvs...
package caltool.file;

import mvp.*;
import caltool.*;
import caltool.caldb.*;
import java.util.*;
import java.io.*;

/****
 *
 * Class File is the model class for the Calendar Tool file handling.  It
 * contains methods for all of the operations defined on the File menu, which
 * constitute the functional command group for file handling.
 *
 * @author Gene Fisher (gfisher@calpoly.edu)
 * @version 6feb04
 *
 */
public class File extends Model {

    /**
     * Construct this with the given companion view and the parent CalendarDB
     * model.  The CalendarDB is provided for its service methods that access
     * the Calendar Tool workspace.
     */
    public File(View view, CalendarDB calDB, CalendarTool calTool) {
        super(null);
        this.calDB = calDB;
        this.calTool = calTool;
    }

    /*-*
     * Derived methods
     */

    /**
     * Add a new empty calendar to the workspace and make it current.
     */
    public void fileNew() {
        System.out.println("In File.fileNew");
    }

    /**
     * Open an existing calendar file of the given name and put the data from
     * that file in the workspace.
     */
    public void open(String filename) {
        System.out.println("In File.open with filename arg = " + filename);

    }

    /**
     * Close the current calendar if it does not require saving.  If saving is
     * required, ask the user what to do.
     */
    public void close() {
        System.out.println("In File.close");
    }

    /**
     * Close the all open calendars if they do not require saving.  If saving
     * is required, ask the user what to do.
     */
    public void closeAll() {
        System.out.println("In File.closeAll");
    }

    /**
     * If the current calendar in the workspace requires saving, save it.
     */
    public void save() {
        try {
            calTool.setView(null);
            FileOutputStream outFile = new FileOutputStream("caltool.dat");
            ObjectOutputStream outStream = new ObjectOutputStream(outFile);
            outStream.writeObject(calDB);
//          calTool.getFile().setView(null);
//          outStream.writeObject(calTool.getFile());
            outStream.writeObject(calTool.getEdit());
//          outStream.writeObject(calTool.getSchedule());
            outStream.writeObject(calTool.getCalView());
            outStream.writeObject(calTool.getAdmin());
            outStream.writeObject(calTool.getOptions());
            outStream.writeObject(calTool.getHelp());
        }
        catch (FileNotFoundException fnfe) {
            System.out.println("File not found.");
        }
        catch (IOException ioe) {
            System.out.println("In file save:");
            ioe.printStackTrace();
        }
    }

    /**
     * Save the current calendar in a file of the given name.
     */
    public void saveAs(String filename) {
        System.out.println("In File.saveAs");
    }

    /**
     * For each open calendar in the workspace, save it if it requires saving.
     */
    public void saveAll() {
        System.out.println("In File.saveAll");
    }

    /**
     * Save the current workspace configuration, including the positions of all
     * open view windows.
```

```java
     */
    public void saveConfig() {
        System.out.println("In File.saveConfig");
    }

    /**
     * Set the local files directory in which standard Calendar Tool files are stored.
     */
    public void localFiles() {
        System.out.println("In File.localFiles");
    }

    /**
     * Install the given page setup info.
     */
    public void print(/* page setup info goes here */) {
        System.out.println("In File.pageSetup");
    }

    /**
     * Print the current calendar per the given print specs.
     */
    public void print(PrintSpecs printSpecs) {
        System.out.println("In File.print");
    }

    /**
     * Exit the Calendar Tool.  If saving is required for any open calendars,
     * ask the user what to do.
     */
    public void exit() {
        System.out.println("In File.exit.");
        System.exit(0);
    }

    /**
     * Temporary system test method to dump out the current user calendar to
     * stdout.
     */
    public void dumpUserCal() {
        System.out.println(calDB.getCurrentCalendar().toString());
    }


    /*-*
     * Data fields
     */

    /** The CalendarDB, containing the data to be stored onto files and into
        which file data are read. */
    CalendarDB calDB;

    /** Temp ref to top-level tool for serialization testing purposes */
    CalendarTool calTool;

}
```