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

import caltool.file.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.lang.reflect.*;
import mvp.*;

/****
 *
 * Class FileMenu is the pulldown menu view of the <a href =
 * "../file/File.html" File </a> model class.  The FileMenu widget is a Java
 * JMenu.  Anonymous instances of JMenuItem are defined for each item in the
 * menu.
 *
 * @author Gene Fisher (gfisher@calpoly.edu)
 * @version 6feb04
 *
 */
public class FileMenu extends View {

    /**
     * Construct this with the given File model.
     */
    public FileMenu(Screen screen, File file, FileUI fileUI) {

        /*
         * Call the parent constructor.
         */
        super(screen, file);

        /*
         * Create a local reference to super.model for non-casting convenience.
         * This way, the model can be referenced as "file" instead of
         * "(File)model".
         */
        this.file = file;

    }

    /**
     * Compose this by inserting each of its nine menu items into the pulldown
     * menu.  The items are New, Open, Close, Close All, Save, Save As, Save
     * All, Print, and Exit.  JSeparators are placed after the ’Close All’,
     * ’Save All’, and ’Print’ items.
     */
    public Component compose() {

        /*
         * Make the widget of this the JMenu.
         */
        widget = new JMenu("File");

        /*
         * Create a local conveience ref to super.widget.
         */
        menu = (JMenu) widget;

        /*
         * Add each of the items to the menu, with separators in the
         * appropriate places (per the requirements).
         */
        addNewItem();
        addOpenItem();
        menu.add(new JSeparator());

        addCloseItem();
        addCloseAllItem();
        menu.add(new JSeparator());

        addSaveItem();
        addSaveAsItem();
        addSaveAllItem();
        menu.add(new JSeparator());

        addLoadSettingsItem();
        addSaveSettingsItem();
        menu.add(new JSeparator());

        addPageSetupItem();
        addPrintItem();
        menu.add(new JSeparator());

        addExitItem();
        menu.add(new JSeparator());
        addSystemTestItems();

        return widget;

    }


    /*-*
     * Protected methods
     */

    /**
     * Add the ’New’ menu item.  Its action listener invokes the File.fileNew
     * method.
     *
     * This and all other menu items are created using the following pattern:
     *                                                                    <pre>
     *     JMenu.add(new JMenuItem("<em>Item name</em>")).addActionListener(
     *         new ActionListener() {
     *         public void actionPerformed(ActionEvent e) {
     *             <em>Model.method()</em>
     *                                                                   </pre>
     */
    protected void addNewItem() {

        /*
         * Use the standard menu item pattern for the item.
         */
        menu.add(new JMenuItem("New")).addActionListener(
            new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                    /*
                     * Call the stubbed-out model method.
                     */
                    file.fileNew();
                }
            }
        );
    }

    /**
     * Add the ’Open ...’ menu item.  Its action listener invokes the File.open
     * method.  See the description of the <a href= FileMenu.html#addNewItem>
     * addNewItem </a> method for further info.
     */
    protected void addOpenItem() {

        final JMenuItem item;          // Used by JFileChooser for placement

        menu.add(
          item = new JMenuItem("Open ...")).addActionListener(
            new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                    JFileChooser chooser = new JFileChooser();
                    java.io.File f;
                    chooser.showOpenDialog(item);
                    if ((f = chooser.getSelectedFile()) != null) {
                        file.open(f.getName());
                    }
                }
            }
        );
    }

    /**
     * Add the ’Close’ menu item.  Its action listener invokes the File.close
     * method.  See the description of the <a href= FileMenu.html#addNewItem>
     * addNewItem </a> method for further info.
     */
    protected void addCloseItem() {
        menu.add(new JMenuItem("Close")).addActionListener(
            new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                    file.close();
                }
            }
        );
    }

    /**
     * Add the ’Close All menu item.  Its action listener invokes the
     * File.closeAll method.  See the description of the <a href=
     * FileMenu.html#addNewItem> addNewItem </a> method for further info.
     */
    protected void addCloseAllItem() {
        menu.add(new JMenuItem("Close All")).addActionListener(
            new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                    file.closeAll();
                }
            }
        );
    }

    /**
     * Add the ’Save’ menu item.  Its action listener invokes the File.save
     * method.  See the description of the <a href= FileMenu.html#addNewItem>
     * addNewItem </a> method for further info.
     */
    protected void addSaveItem() {
        menu.add(new JMenuItem("Save")).addActionListener(
            new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                    file.save();
                }
            }
        );
    }

    /**
     * Add the ’Save As ...’ menu item.  Its action listener displays a file
     * chooser to select the file on which to save, then invokes the
     * File.saveAs method.  See the description of the <a href=
     * FileMenu.html#addNewItem> addNewItem </a> method for further info.
     */
    protected void addSaveAsItem() {
        final JMenuItem item;          // Used by JFileChooser for placement

        menu.add(
          item = new JMenuItem("Save As ...")).addActionListener(
            new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                    (new JFileChooser()).showSaveDialog(item);
                    file.saveAs(null);
                }
            }
        );
    }

    /**
     * Add the ’Save All’ menu item.  Its action listener invokes the
     * File.saveAll method.  See the description of the <a href=
     * FileMenu.html#addNewItem> addNewItem </a> method for further info.
     */
    protected void addSaveAllItem() {
        menu.add(new JMenuItem("Save All")).addActionListener(
            new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                    file.saveAll();
                }
            }
        );
    }

    /**
     * Add the 'Load Settings' menu item.  Its action listener displays a load
     * settings dialog, from which the File.loadSettings method is invoked.
     * See the description of the <a href= FileMenu.html#addNewItem> addNewItem
     * </a> method for further info.
     */
    protected void addLoadSettingsItem() {
        final JMenuItem item;           // Used by JFileChooser for placement

        menu.add(
          item = new JMenuItem("Load Settings ...")).addActionListener(
            new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                    /*
                     * For the stubbed design, just display a quickie dialog
                     * that will be refined later.
                     */
                    JOptionPane.showInputDialog("... Load Settings stuff goes here ...");
                }
            }
        );
    }

    /**
     * Add the 'Load Settings' menu item.  Its action listener displays a save
     * settings dialog, from which the File.saveSettings method is invoked.
     * FileMenu.html#addNewItem> addNewItem </a> method for further info.
     */
    protected void addSaveSettingsItem() {
        final JMenuItem item;           // Used by JFileChooser for placement

        menu.add(
          item = new JMenuItem("Save Settings ...")).addActionListener(
            new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                    /*
                     * For the stubbed design, just display a quickie dialog
                     * that will be refined later.
                     */
                    JOptionPane.showInputDialog("... Save Settings stuff goes here ...");
                }
            }
        );
    }

    /**
     * Add the 'Page Setup ...' menu item.  Its action listener displays a Page
     * Setup dialog, from which the File.pageSetup method is invoked.  See the
     * description of the <a href= FileMenu.html#addNewItem> addNewItem </a>
     * method for further info.
     */
    protected void addPageSetupItem() {
        final JMenuItem item;           // Used by JFileChooser for placement

        menu.add(new JMenuItem("Page Setup ...")).addActionListener(
            new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                    /*
                     * For the stubbed design, just display a quickie dialog
                     * that will be refined later.
                     */
                    JOptionPane.showInputDialog("... Page Setup stuff goes here ...");
                }
            }
        );
    }

    /**
     * Add the 'Print ...' menu item.  Its action listener invokes the
     * File.print method.  See the description of the <a href=
     * FileMenu.html#addNewItem> addNewItem </a> method for further info.
     */
    protected void addPrintItem() {
        final JMenuItem item;           // Used by JFileChooser for placement

        menu.add(new JMenuItem("Print ...")).addActionListener(
            new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                    /*
                     * For the stubbed design, just display a quickie dialog
                     * that will be refined later.
                     */
                    JOptionPane.showInputDialog("Choose a printer: ");
                    file.print(null);
                }
            }
        );
    }

    /**
     * Add the 'Exit' menu item.  Its action listener invokes the File.exit
     * method.  See the description of the <a href= FileMenu.html#addNewItem>
     * addNewItem </a> method for further info.
     */
    protected void addExitItem() {
        menu.add(new JMenuItem("Exit")).addActionListener(
            new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                    file.exit();
                }
            }
        );
    }
    /**
     * Add menu items to perform any system tests that may be useful during
     * development.  This submenu goes away in the production version of the
     * system.
     */
    protected void addSystemTestItems() {
        JMenu menu = new JMenu("System Tests");
        menu.add(menu);

        /*
         * Add the ’Dump User Cal’ menu item.
         */
        menu.add(new JMenuItem("Dump User Cal")).addActionListener(
            new ActionListener() {
                public void actionPerformed(ActionEvent ev) {
                    file.dumpUserCal();
                }

            }
        );

    }


    /*-*
     * Data fields
     */

    /** Local reference to super.model for convenience to avoid casting
     * everywhere; i.e., file is set equal to (File)super.model once in the
     * constructor.
     */
    File file;

    /** Local reference to (JMenu)super.widget for convenience to avoid
     * casting; i.e., menu is set equal to (File)super.widget once in the
     * constructor. */
    JMenu menu;

}
```