```java
package rolodex;

import javax.swing.*;
import java.awt.*;
import mvp.*;

/****
 *
 * Class RolodexMultiViewMenuUI is a companion view to the <a href =
 * "RolodexTool.html"> RolodexTool </a> model class.  RolodexMultiViewMenuUI
 * specializes the standard rolodex ui defined in class <a href =
 * "RolodexMenuUI.html"> RolodexMenuUI, </a> q.v.  RolodexMultiViewMenuUI has
 * all of the functionality of its parent class, plus additional three
 * additional rolodex views:
 *                                                                        <ol>
 *     <li> a command-line user interface, defined in <a href =
 *     "RolodexCommandEditor.html"> RolodexCommandEditor </a>
 *
 *     <li> a full text editor user interface, defined in <a href =
 *     "RolodexTextEditor.html"> RolodexTextEditor </a>
 *
 *     <li> a rolodex tool-bar, defined in <a href = "RolodexToolBar.html">
 *     RolodexToolBar </a>
 *                                                                       </ol>
 * In addtion, there is a new pulldown menu named "Windows" that has checkboxes
 * to enable/disable the three new views.
 *
 */

public class RolodexMultiViewMenuUI extends RolodexMenuUI {

    /**
     * Construct this with the given UI screen and RolodexTool model.  Call the
     * parent constructor and then construct the additional views.
     */
    public RolodexMultiViewMenuUI(Screen s, RolodexTool rt) {

        /*
         * Call the parent constructor.
         */
        super(s, rt);

        /*
         * Construct the windows menu.  Note that it is a pure view menu, and
         * therefore requires no model access.
         */
        wm = new WindowsMenu("Windows", this);

        /*
         * Construct of the additional rolodex views.  Note that the Rolodex is
         * the companion model for each.
         */
        rce = new RolodexCommandEditor(s, this, rt.getRolodex());
        rte = new RolodexTextEditor(s, this, rt.getRolodex());
        rtb = new RolodexToolBar(s, this);

    }

    /**
     * Call the parent compose, add the Windows menu to the menubar, and
     * compose the three new views.
     */
    public void compose() {

        /*
         * Call RolodexMenuUI.compose.
         */
        super.compose();

        /*
         * Add the Windows menu to the menubar.
         */
        jmb.add(composeWindowsMenu().getMenu());

        /*
         * Compose each of the additional views.
         */
        rce.compose();
        rte.compose();
        rtb.compose();

    }

    /**
     * Show or hide the command-line editor window.
     */
    public void showCommandEditor(boolean b) {
        if (b)
            rce.show();
        else
            rce.hide();
    }

    /**
     * Show or hide the text editor window.
     */
    public void showTextEditor(boolean b) {
        if (b)
            rte.show();
        else
            rte.hide();
    }

    /**
     * Show or hide the command-line editor window.
     */
    public void showToolBar(boolean b) {
        if (b)
            rtb.show();
        else
            rtb.hide();
    }
```

```java


    /*-*
     * Component composition functions.
     */

    /**
     * Compose and return the windows pulldown menu.
     */
    protected WindowsMenu composeWindowsMenu() {
        wm.compose();
        return wm;
    }


    /*-*
     * Data members
     */

    /** Width of the command EDITOR window */
    public static final int COMMAND_ED_WIDTH = 60;

    /** Height of the command processor window */
    public static final int COMMAND_ED_HEIGHT = 15;

    /** Width of the text editor window */
    public static final int TEXT_ED_WIDTH = 60;

    /** Height of the text editor window */
    public static final int TEXT_ED_HEIGHT = 35;

    /** Windows pulldown menu */
    protected WindowsMenu wm;

    /** Command editor view */
    RolodexCommandEditor rce;

    /** Command editor view */
    RolodexTextEditor rte;

    /** Command editor view */
    RolodexToolBar rtb;

}
```