```java
package rolodex;

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.util.*;
import mvp.*;

/****
 *
 * Class RolodexTextEditor is a purely textual view of the <a href =
 * "Rolodex.html"> Rolodex model </a> class.  It presents a UNIX-like
 * (Notepad-like) plain text editor in which the textual version of the
 * complete rolodex is displayed and may be edited in a restricted fashion.
 * The RolodexTextEditor window is a java JTextArea.
 *                                                                     <p>
 * It is noteworthy this textual rolodex view uses exactly the same <a href=
 * "Rolodex.html"> Rolodex model API </a> as the GUI rolodex view in <a href=
 * "RolodexMenuUI.html"> RolodexMenuUI. </a>
 *                                                                     <p>
 * It is also noteworthy that this view is an <em>observer</em> of its
 * companion Rolodex model.  This means that this' update method will be called
 * whenever the Rolodex has changed.  The determination of when the model
 * object has actually changed is made by the model itself, when it calls the
 * setChanged method.  In the Rolodex Model, setChanged is called in <a href=
 * "Rolodex.html#add(rolodex.Card)"> Rolodex.add </a>, <a href=
 * "Rolodex.html#delete(rolodex.Id)"> Rolodex.delete </a>, and <a href=
 * "Rolodex.html#change(rolodex.Id, rolodex.Card)"> Rolodex.change, </a> q.q.v.
 * The determination of when to notify observers is made in View objects that
 * call Model objects that may (potentially) change the model.  In the Rolodex
 * Tool, notifying View classes are the button listeners that confirm one of
 * the Rolodex changing operations, viz., <a href=
 * "OKAddButtonListener.html#actionPerformed(java.awt.event.ActionEvent)">
 * OKAddButtonListener.actionPerformed, </a> <a href=
 * "OKDeleteButtonListener.html#actionPerformed(java.awt.event.ActionEvent)">
 * OKDeleteButtonListener.actionPerformed, </a> and <a href=
 * "OKChangeButtonListener.html#actionPerformed(java.awt.event.ActionEvent)">
 * OKChangeButtonListener.actionPerformed, </a> q.q.v.
 *
 */

public class RolodexTextEditor extends View {

    /**
     * Construct this with the given Screen, RolodexMultiViewMenuUI grandparent
     * view, and Rolodex model.
     */
    public RolodexTextEditor(Screen s, RolodexMultiViewMenuUI rmui,
            Rolodex r) {

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

        /*
         * Set the grandparent view.
         */
        this.rmui = rmui;

        /*
         * Construct the text area.
         */
        jta = new JTextArea(
            RolodexMultiViewMenuUI.TEXT_ED_HEIGHT,
            RolodexMultiViewMenuUI.TEXT_ED_WIDTH);

        /*
         * Make this an observer of the companion Rolodex model.
         */
        r.addObserver(this);
    }

    /**
     * Compose the window of this by adding the JTextArea and commit button.
     */
    public void compose() {

        /*
         * Make the window.
         */
        window = new mvp.Window();

        /*
         * Set the font to something "terminal looking".
         */
        jta.setFont(new Font("Courier", Font.PLAIN, 12));

        /*
         * Add some canned text for demo purposes.
         */
        jta.append(
            "Name: Gene Fisher\n" +
            "Id: 563928591\n" +
            "Age: 48\n" +
            "Sex: M\n" +
            "Addr: Bldg 14, Room 210\n" +
            "\n" +
            "Name: Ellen Stier\n" +
            "Id: 890194211\n" +
            "Age: 42\n" +
            "Sex: F\n" +
            "Addr: Bldg 14, Room 252\n"
        );

        /*
         * Make a vertical box with text area and a commit button; add the box
         * to the window's content pane.
         */
        Box b = new Box(BoxLayout.Y_AXIS);
        b.add(jta);
```

```
        JButton jb = new JButton("Commit Changes");
        jb.setAlignmentX(JFrame.CENTER_ALIGNMENT);
        b.add(jb);
        window.getContentPane().add(b);

        /*
         * Don't forget to set the title and pack the window.
         */
        window.setTitle("Rolodex Text Editor");
        window.pack();

    }

    /**
     * Respond to a change in the companion model.  The current implementation
     * is simply a stub.  The complete implementation would display the updated
     * Rolodex in the text area.  This could be accomplished brute force by
     * completely erasing and redisplaying the text area with a full
     * enumeration of the Rolodex cards.  Alternatively, a more sophisticated
     * form of incremental update could be performed that only changes the
     * affected card.  In the latter case, callers of this update method could
     * oblige by passing in the card that was affected, plus an indication of
     * the operation performed.  Since the current specification of the Rolodex
     * Model only provides single-card updating operations (i.e., add, delete,
     * change), incremental update can be guaranteed only to affect only one
     * card.
     */
    public void update(Observable o, Object arg) {
        System.out.println("In RolodexTextEditor.update");
    }


    /** The representation of this is a JTextArea. */
    protected JTextArea jta;

    /** The grandparent view. */
    protected RolodexMultiViewMenuUI rmui;

}
```