

```

/**
 * Class Name      : MainFrame.java
 * Version         : Build 01.10
 * Date of Change  : 2000 MAR 11
 * Author         : Don Baldwin
 * Copyright       : 2000 ASR Strategic Resources
 * Description     : The GUI and event listeners
 *                  for the main frame.
 *
 * Version History :
 *
 * Version         Date           Author
 * Build 01.10     2000 MAR 11     Feature enhancement
 *                  build. Improved GUI.
 * Build 01.00     2000 MAR 10     Anders Nyström
 *                  Started the class
 *
 * Known Errors: None
 */

```

```

import java.util.Vector;
import java.util.Enumeration;

```

```

import java.awt.List;
import java.awt.Label;
import java.awt.TextField;
import java.awt.Frame;
import java.awt.TextArea;
import java.awt.Button;
import java.awt.Panel;
import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.BorderLayout;
import java.awt.PopupMenu;
import java.awt.MenuItem;
import java.awt.Choice;

```

```

import java.awt.event.MouseListener;
import java.awt.event.ItemListener;
import java.awt.event.ActionListener;
import java.awt.event.WindowListener;
import java.awt.event.ActionEvent;
import java.awt.event.WindowEvent;
import java.awt.event.ItemEvent;
import java.awt.event.MouseEvent;

```

```

import java.sql.SQLException;

```

```

public class MainFrame extends Frame
{
    private static MainFrame objMainFrame = null;

    private MainFrameButtonListener listenButton;
    private MainFrameWindowListener listenWindow;
    private MainFrameMouseListener listenMouse;
    private MainFrameItemListener listenItem;

    private Vector vecAddresses = null;
    private long longCurrentId = 0;
    private boolean isNewAddress = false;

    private PopupMenu popUpMenu;

    private Button btnNew;

```

```

private Button btnEdit;
private Button btnDelete;
private Button btnSearch;
private Button btnReset;

private Panel panelNorth;
private Panel panelSouth;
private Panel panelCenter;

private TextField fldSearch;
private TextField fldInput;
private TextField fldName;
private TextArea areaWhoList;

private List lstDbInfo;
private Choice choiceTable;

/**
 * The only way to get the main frame, we only need one.
 */
public static MainFrame getMainFrame()
{
    if(objMainFrame == null)
    {
        objMainFrame = new MainFrame("Address Book");
    }

    return objMainFrame;
}

/**
 * Hide the constructor, only accessible from getMainFrame().
 */
private MainFrame(String aStrTitle)
{
    super(aStrTitle);
    setSize(500,350);
    setLayout(new BorderLayout());

// The graphical setup for the application main frame.
    fldSearch = new TextField(20);
    fldInput = new TextField(50);
    fldName = new TextField(20);
    areaWhoList = new TextArea(10,30);

    panelNorth = new Panel();
    panelSouth = new Panel();
    panelCenter = new Panel();

    btnNew = new Button("New");
    btnEdit = new Button("Edit");
    btnDelete = new Button("Delete");
    btnSearch = new Button("Search");
    btnReset = new Button("Show All");

    choiceTable = new Choice();
    lstDbInfo = new List();

    panelNorth.add(fldSearch);
    panelNorth.add(choiceTable);
    panelNorth.add(btnSearch);
    panelNorth.add(btnReset);
    add("North",panelNorth);

```

```

panelSouth.add(btnNew);
panelSouth.add(btnEdit);
panelSouth.add(btnDelete);
add("South",panelSouth);

add("Center",areaWhoList);
add("West",lstDbInfo);

// Add all table names to the choice list
fillTableChoices();

// Add action listeners and other listeners
addListeners();

// Read all addresses into the registry
Registry.readAllAddresses();

// Put all addresses in the registry into the list.
fetchInformation();
}

/**
 * Wrapper method to show the main frame.
 */
public void showMainFrame()
{
    setVisible(true);
}

/**
 * Will enable the frame and clear the address info in the
 * right window.
 */
public void enableMainFrame()
{
    areaWhoList.setText("");
    setEnabled(true);
}

/**
 * Just disable the frame. Not really necessary but
 * it looks good. ;)
 */
public void disableMainFrame()
{
    setEnabled(false);
}

/**
 * When we return from the edit frame for instance,
 * we clear the search field and reset the choice list.
 */
public void refreshMainFrame()
{
    fldSearch.setText("");
    choiceTable.select("Firstname");
}

/**
 * Simple method to add all the listeners to their
 * different components.
 */
private void addListeners()
{

```

```

listenWindow    = new MainFrameWindowListener();
listenButton   = new MainFrameButtonListener();
listenMouse     = new MainFrameMouseListener();
listenItem      = new MainFrameItemListener();

addWindowListener(listenWindow);

lstDbInfo.addItemListener(listenItem);
lstDbInfo.addMouseListener(listenMouse);

btnNew.addActionListener(listenButton);
btnEdit.addActionListener(listenButton);
btnDelete.addActionListener(listenButton);
btnSearch.addActionListener(listenButton);
btnReset.addActionListener(listenButton);

}

/**
 * This will fill the choice list with all table names
 * in the database.
 */
private void fillTableChoices()
{
    try
    {
        Enumeration    allTableNames;

        allTableNames = Address.getAllTableNames();

        while(allTableNames.hasMoreElements())
        {
            String      strTableName;

            strTableName = (String)allTableNames.nextElement();

            choiceTable.add(strTableName);
        }
    }
    catch(SQLException excSQL)
    {
        System.out.println(excSQL.getMessage());
    }
}

/**
 * This will update the address list so it
 * matches the registry's address list.
 */
public void fetchInformation()
{
    Vector    vecAddresses;
    Enumeration    allAddresses;

    vecAddresses = Registry.getAddresses();
    allAddresses = vecAddresses.elements();

    lstDbInfo.removeAll();

    while(allAddresses.hasMoreElements())
    {
        Address    objAddress;
        long    longId;
        String    strName;
    }
}

```

```

        objAddress = (Address)allAddresses.nextElement();
        longId      = objAddress.getId();
        strName     = objAddress.getFirstLastName();

        // We display the id, to separate people
        // with the same name easily.
        lstDbInfo.add(longId + ", " + strName);
    }

    areaWhoList.setText("");
}

/*
 * The inner listener class for the window's actions.
 */
private class MainFrameWindowListener implements WindowListener
{
    public void windowOpened(WindowEvent e){}
    public void windowClosed(WindowEvent e){}
    public void windowIconified(WindowEvent e){}
    public void windowDeiconified(WindowEvent e){}
    public void windowActivated(WindowEvent e){}
    public void windowDeactivated(WindowEvent e){}

    /**
     * Called if the user clicks on the close window button or
     * closes the window in another way.
     */
    public void windowClosing(WindowEvent e)
    {
        dispose();
        System.exit(0);
    }

}

/**
 * The inner listener class for the buttons' actions.
 */
private class MainFrameButtonListener implements ActionListener
{
    /**
     * Called when the user clicks a button or chooses
     * a popup-menu alternative.
     */
    public void actionPerformed(ActionEvent evtAction)
    {
        Object    objSource;
        String    strCommand;

        objSource = evtAction.getSource();
        strCommand = evtAction.getActionCommand();

        /**
         * The user clicks on the new button.
         */
        if(objSource == btnNew)
        {
            EditFrame    objEditFrame;

            objEditFrame = EditFrame.getEditFrame();

            objEditFrame.newPost();
        }
    }
}

```

```

    }

    // The user clicks on the edit button OR clicks
    // the menu option edit on the popup menu.
    else if(strCommand.equals("Edit"))
    {
        Address objAddress;
        EditFrame objEditFrame;

        objEditFrame = EditFrame.getEditFrame();

        // If nothing is selected in the list, ignore.
        if(lstDbInfo.getSelectedIndex() < 0)
        {
            return;
        }

        objAddress = Registry.getAddress(longCurrentId);
        objEditFrame.updatePost(objAddress);
    }

    // The user clicks on the delete button OR clicks
    // the menu option delete on the popup menu.
    else if(strCommand.equals("Delete"))
    {

        // If nothing is selected in the list, ignore.
        if(lstDbInfo.getSelectedIndex() < 0)
        {
            return;
        }
        try
        {
            MainFrame objMainFrame;

            objMainFrame = MainFrame.getMainFrame();

            Address.deleteAddress(longCurrentId);
            Registry.removeAddress(longCurrentId);

            objMainFrame.fetchInformation();
        }
        catch(SQLException excSQL)
        {
            System.out.println("Exception deleting post: "+excSQL);
        }
    }

    // The user clicks on the search button.
    else if(objSource == btnSearch)
    {
        Vector vecAddresses = null;

        String strValue;
        String strTable;

        strValue = fldSearch.getText();
        strTable = choiceTable.getSelectedItemId();

        // Nothing typed in the search textfield? Ignore.
        if(strValue.equals(""))
        {
            return;
        }
    }

```

```

    }

    // Search the database (via the address static method) to
    // find the address(es) the match the criterias.
    try
    {
        MainFrame objMainFrame;

        objMainFrame = MainFrame.getMainFrame();
        vecAddresses = Address.searchAddresses(strTable, strValue);

        Registry.setAddresses(vecAddresses);
        objMainFrame.fetchInformation();
    }
    catch(SQLException excSQL)
    {
        System.out.println("MainFrame exception: "+excSQL);
    }
}

// The user click on the "show all" button, we read all
// database addresses into the registry's address list.
else if(objSource == btnReset)
{
    refreshMainFrame();

    Registry.readAllAddresses();

    fetchInformation();
}
}

/**
 * The inner listener class for the mouse clicks.
 */
private class MainFrameMouseListener implements MouseListener
{
    /**
     * The following four empty methods are here to legally
     * implement the mouse listener interface.
     */
    public void mouseEntered(MouseEvent me)
    {
    }
    public void mouseClicked(MouseEvent me)
    {
    }
    public void mouseReleased(MouseEvent me)
    {
    }
    public void mouseExited(MouseEvent me)
    {
    }

    /**
     * The user pressed one of the mouse buttons.
     */
    public void mousePressed(MouseEvent evtMouse)
    {
        Object objSource;

        objSource = evtMouse.getSource();
    }
}

```

```

//Right click check.
if(evtMouse.getModifiers() == 4)
{
    int        intIndex;
    boolean    isNewAddress;
    MenuItem   objMenuItem;

    // Nothing selected in the list - ignore.
    if(lstDbInfo.getSelectedIndex() < 0)
    {
        return;
    }

    // Create a new popup menu.
    popUpMenu = new PopupMenu("Modify");
    String strHeading = lstDbInfo.getSelectedItem();
    intIndex = strHeading.indexOf(',');
    strHeading = strHeading.substring(intIndex+1);

    objMenuItem = new MenuItem(strHeading);

    objMenuItem.setEnabled(false);
    popUpMenu.add(objMenuItem);
    popUpMenu.addSeparator();

    String[] menuLabels = new String[]
    {
        "Edit",
        "Delete",
    };

    for(int i = 0; i < menuLabels.length; i++)
    {
        objMenuItem = new MenuItem(menuLabels[i] );

        objMenuItem.addActionListener(listenButton);

        popUpMenu.add(objMenuItem);
    }

    // Add the popup menu to the list.
    lstDbInfo.add(popUpMenu);

    if(objSource == lstDbInfo)
    {
        int    intMouseX;
        int    intMouseY;

        intMouseX = evtMouse.getX() + 20;
        intMouseY = evtMouse.getY();

        popUpMenu.show(MainFrame.getMainFrame(),intMouseX,intMouseY);
    }
}

// Double click left mouse?
else if(evtMouse.getClickCount() == 2 &&
        evtMouse.getModifiers() == 16)
{
    Address    objAddress;
    EditFrame  objEditFrame;

    if(lstDbInfo.getSelectedIndex() < 0)

```

```

        {
            return;
        }

        objEditFrame = EditFrame.getEditFrame();

        // Find the address with the selected id, more
        // the information into the edit frame.
        objAddress = Registry.getAddress(longCurrentId);
        objEditFrame.updatePost(objAddress);
        objEditFrame.focusEditFrame();
    }
}

public class MainFrameItemListener implements ItemListener
{
    /**
     * Method that listens on the list and handles the display of personal information to the right
     */
    public void itemStateChanged(ItemEvent evtItem)
    {
        Object    objSource;
        int       intPos;
        String    strItem;
        String    strId;
        Enumeration allAddresses;

        objSource = evtItem.getSource();

        // Nothing selected in the list - ignore.
        if(lstDbInfo.getSelectedIndex() < 0)
        {
            return;
        }

        // We changed the focus in the address list, display
        // the new item to the right of it.
        if(objSource == lstDbInfo)
        {
            Address    objAddress;

            strItem    = lstDbInfo.getSelectedItem();
            intPos     = strItem.indexOf(',');
            strId      = strItem.substring(0,intPos);
            longCurrentId = Long.parseLong(strId);

            objAddress = Registry.getAddress(longCurrentId);

            areaWhoList.setText(objAddress.getMailingLabel());
        }
    }
} // END MainFrame.java

```