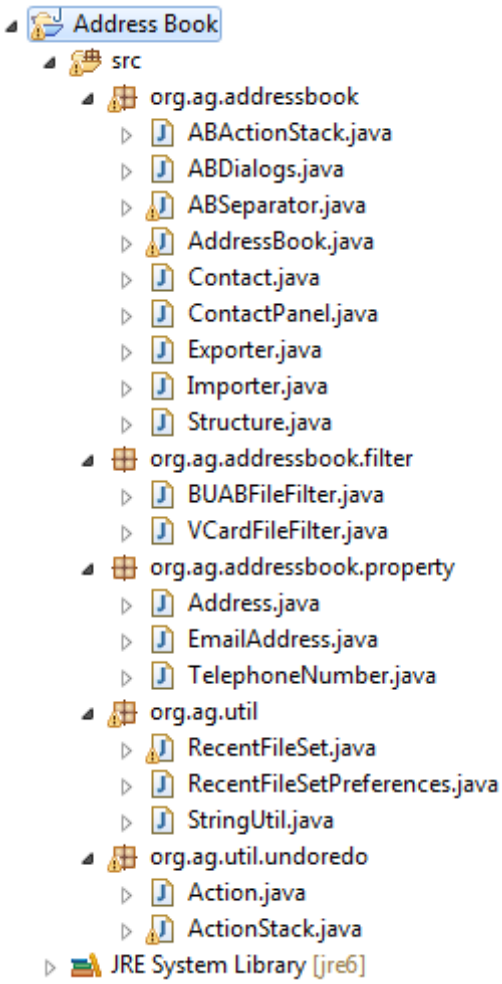
Ashley Gwinnell
Student Number: 4204611

Programming 2
Assignment 1

Address Book: Source Code

Index:



```
package org.ag.addressbook;
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.GridLayout;
import java.awt.Insets;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.InputEvent;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;
import java.io.File;
import java.io.IOException;
import java.util.ArrayList;

import javax.swing.DefaultListModel;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JList;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JSeparator;
import javax.swing.JSplitPane;
import javax.swing.JTextField;
import javax.swing.KeyStroke;
import javax.swing.ListSelectionModel;
import javax.swing.UIManager;
import javax.swing.border.MatteBorder;
import javax.swing.event.ListSelectionEvent;
import javax.swing.event.ListSelectionListener;

import org.ag.util.RecentFileSet;
import org.ag.util.undoredo.Action;
import org.ag.util.undoredo.ActionStack;

/**
 * Address Book
 *  TODO:
 *   - UI
 *    - ContactPanel to be aligned at top.
 *   - help file.
 *    - search is separated by commas.
 *    - open is different from import.
 *   - link up to facebook / netbin.
 *   - clean up import of duplicate contacts so it shows "duplicate contact 1 of 6".
 *
 * @version 0.6
 * @author Ashley Gwinnell
 */
public class AddressBook
{
      private JFrame frame;
      private DefaultListModel model;
      private JList list;
      private File currentlyOpenedFile;
      private ContactPanel contactPanel;

      private ABActionStack actionStack;
      private int savedAtStackLocation = -1;
```

```
	private RecentFileSet recentFileSet;

	private JMenuItem m_undo;
	private JMenuItem m_redo;
	private JButton tb_undo;
	private JButton tb_redo;

	private JTextField tf_search;
	private String filter = "";

	private ArrayList<Contact> contacts = new ArrayList<Contact>();
	private ArrayList<Contact> filteredContacts = new ArrayList<Contact>();
	private ArrayList<Contact> unfilteredContacts = new ArrayList<Contact>();

	/**
	 * Create a new Address book window.
	 */
	public AddressBook()
	{
		try {
			UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
		}
		catch (Exception e) { }

		JFrame.setDefaultLookAndFeelDecorated(false);

		frame = new JFrame();
		frame.setTitle("Address Book - Untitled Document");
		frame.setSize(750, 560);
		frame.setMinimumSize(new Dimension(750, 560));
		frame.setIconImage(new ImageIcon("files/FrameIcon.png").getImage());
		frame.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
		frame.setLocationRelativeTo(null);

		recentFileSet = new RecentFileSet(5);

		JMenuBar menubar = new JMenuBar();

			JMenu menu_file = new JMenu("File");
			menu_file.setMnemonic(KeyEvent.VK_F);

				JMenuItem menuitem_new = new JMenuItem("New");
				menuitem_new.setMnemonic(KeyEvent.VK_N);

	menuitem_new.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_N,
InputEvent.CTRL_DOWN_MASK));
				menuitem_new.addActionListener(new ActionListener() { public
void actionPerformed(ActionEvent e) {
					newFile();
				}});
				menuitem_new.setIcon(new ImageIcon("files/Doc-Add.png"));
				menu_file.add(menuitem_new);

				menu_file.add(new JSeparator());

				JMenuItem menuitem_open = new JMenuItem("Open File");
				menuitem_open.setMnemonic(KeyEvent.VK_O);

	menuitem_open.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_O,
InputEvent.CTRL_DOWN_MASK));
				menuitem_open.setIcon(new ImageIcon("files/Folder.png"));
				menuitem_open.addActionListener(new ActionListener() { public
void actionPerformed(ActionEvent e) {
					open();
				}});
				menu_file.add(menuitem_open);

				JMenu menu_recentfiles = new JMenu("Open Recent File ");
```

```
                        menu_recentfiles.setIcon(new ImageIcon("files/Folder.png"));
                        final ArrayList<String> recentfiles = this.recentFileSet.get();
                        for (int i = 0; i < recentfiles.size(); i++) {
                              final int j = i;
                              JMenuItem menuitem_recentfile = new JMenuItem((i + 1) + ")
" + recentfiles.get(i));
                              menuitem_recentfile.addActionListener(new ActionListener()
{
                                    public void actionPerformed(ActionEvent e) {
                                          open(new File(recentfiles.get(j)));
                                    }
                              });
                              menu_recentfiles.add(menuitem_recentfile);
                        }
                        if (recentfiles.size() == 0) {
                              menu_recentfiles.setEnabled(false);
                        }
                        menu_file.add(menu_recentfiles);

                        menu_file.add(new JSeparator());

                        JMenuItem menuitem_save = new JMenuItem("Save");
                        menuitem_save.setMnemonic(KeyEvent.VK_S);

      menuitem_save.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_S,
InputEvent.CTRL_DOWN_MASK));
                        menuitem_save.addActionListener(new ActionListener() { public
void actionPerformed(ActionEvent e) {
                              save();
                        }});
                        menuitem_save.setIcon(new ImageIcon("files/Save.png"));
                        menu_file.add(menuitem_save);

                        JMenuItem menuitem_saveas = new JMenuItem("Save As");
                        menuitem_saveas.setMnemonic(KeyEvent.VK_A);

      menuitem_saveas.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_S,
InputEvent.CTRL_DOWN_MASK + InputEvent.SHIFT_DOWN_MASK));
                        menuitem_saveas.addActionListener(new ActionListener() { public
void actionPerformed(ActionEvent e) {
                              saveAs();
                        }});
                        menuitem_saveas.setIcon(new ImageIcon("files/SaveAs.png"));
                        menu_file.add(menuitem_saveas);

                        menu_file.add(new JSeparator());

                        JMenuItem menuitem_import = new JMenuItem("Import Contacts From
File");
                        menuitem_import.setMnemonic(KeyEvent.VK_I);

      menuitem_import.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_I,
InputEvent.CTRL_DOWN_MASK));
                        menuitem_import.addActionListener(new ActionListener() { public
void actionPerformed(ActionEvent e) {
                              importContacts();
                        }});
                        menuitem_import.setIcon(new ImageIcon("files/Datbase-Add.png"));
                        menu_file.add(menuitem_import);

                        JMenu menu_import_recentfiles = new JMenu("Import Contacts From
Recent File ");
                        menu_import_recentfiles.setIcon(new ImageIcon("files/Datbase-
Add.png"));
                        final ArrayList<String> import_recentfiles =
this.recentFileSet.get();
                        for (int i = 0; i < recentfiles.size(); i++) {
                              final int j = i;
```

```
                              JMenuItem menuitem_recentfile = new JMenuItem((i + 1) + ")
" + recentfiles.get(i));
                              menuitem_recentfile.addActionListener(new ActionListener()
{
                                    public void actionPerformed(ActionEvent e) {
                                          importContacts(new File[] {new
File(recentfiles.get(j))});
                                    }
                              });
                              menu_import_recentfiles.add(menuitem_recentfile);
                        }
                        if (import_recentfiles.size() == 0) {
                              menu_import_recentfiles.setEnabled(false);
                        }
                        menu_file.add(menu_import_recentfiles);

                        menu_file.add(new JSeparator());

                        JMenuItem menuitem_exit = new JMenuItem("Exit ");
                        menuitem_exit.setMnemonic(KeyEvent.VK_E);

      menuitem_exit.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_W,
InputEvent.CTRL_DOWN_MASK));
                        menuitem_exit.addActionListener(new ActionListener() {
                              public void actionPerformed(ActionEvent e) {
                                    close();
                              }
                        });
                        menuitem_exit.setIcon(new ImageIcon("files/Delete.png"));
                        menu_file.add(menuitem_exit);

                  menubar.add(menu_file);

                  JMenu menu_edit = new JMenu("Edit");
                  menu_edit.setMnemonic(KeyEvent.VK_E);
                        m_undo = new JMenuItem("Undo");
                        m_undo.setEnabled(false);
                        m_undo.setMnemonic(KeyEvent.VK_U);
                        m_undo.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_Z,
InputEvent.CTRL_DOWN_MASK + InputEvent.ALT_DOWN_MASK));
                        m_undo.addActionListener(new ActionListener() {
                              public void actionPerformed(ActionEvent e) {
                                    actionStack.pop();
                              }
                        });
                        m_undo.setIcon(new ImageIcon("files/Left.png"));
                        menu_edit.add(m_undo);
                        m_redo = new JMenuItem("Redo");
                        m_redo.setEnabled(false);
                        m_redo.setMnemonic(KeyEvent.VK_R);
                        m_redo.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_Y,
InputEvent.CTRL_DOWN_MASK + InputEvent.ALT_DOWN_MASK));
                        m_redo.addActionListener(new ActionListener() {
                              public void actionPerformed(ActionEvent e) {
                                    actionStack.push();
                              }
                        });
                        m_redo.setIcon(new ImageIcon("files/Right.png"));
                        menu_edit.add(m_redo);
                  menubar.add(menu_edit);

                  JMenu menu_help = new JMenu("Help");
                  menu_help.setMnemonic(KeyEvent.VK_H);

                        JMenuItem menuitem_about = new JMenuItem("About");
                        menuitem_about.setMnemonic(KeyEvent.VK_A);

      menuitem_about.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_F1, 0));
```

```
                        menuitem_about.addActionListener(new ActionListener() { public
void actionPerformed(ActionEvent e) {
                              ABDialogs.createAndShowAboutDialog(getFrame());
                        }});
                        menuitem_about.setIcon(new ImageIcon("files/Info.png"));
                        menu_help.add(menuitem_about);


                  menubar.add(menu_help);

            frame.setJMenuBar(menubar);

            frame.addWindowListener(new WindowListener() {
                  public void windowActivated(WindowEvent e) { }
                  public void windowClosed(WindowEvent e) { }
                  public void windowDeactivated(WindowEvent e) { }
                  public void windowDeiconified(WindowEvent e) { }
                  public void windowIconified(WindowEvent e) { }
                  public void windowOpened(WindowEvent e) { }
                  public void windowClosing(WindowEvent e) {
                        close();
                  }
            });

            JPanel toolBar = new JPanel();
            toolBar.setPreferredSize(new Dimension(toolBar.getWidth(), 36));
            toolBar.setLayout(null);
                  int toolbar_separator_num = 0;
                  int toolbar_item_num = 0;
                  int toolbar_item_width = 32;



                  JButton tb_new = new JButton(new ImageIcon("files/Doc-Add.png"));
                  tb_new.setToolTipText("New Document");
                  tb_new.setBounds((toolbar_item_num*toolbar_item_width) + 2 +
(7*toolbar_separator_num), 2, toolbar_item_width, toolbar_item_width);
                  tb_new.setMargin(new Insets(1,1,1,1));
                  tb_new.addActionListener(new ActionListener() { public void
actionPerformed(ActionEvent e) {
                        newFile();
                  } });
                  toolBar.add(tb_new);
                  toolbar_item_num++;

                  JButton tb_open = new JButton(new ImageIcon("files/Folder.png"));
                  tb_open.setToolTipText("Open Document");
                  tb_open.setBounds((toolbar_item_num*toolbar_item_width) + 2 +
(7*toolbar_separator_num), 2,toolbar_item_width, toolbar_item_width);
                  tb_open.setMargin(new Insets(1,1,1,1));
                  tb_open.addActionListener(new ActionListener() { public void
actionPerformed(ActionEvent e) {
                        open();
                  }});
                  toolBar.add(tb_open);
                  toolbar_item_num++;

                  JButton tb_save = new JButton(new ImageIcon("files/Save.png"));
                  tb_save.setToolTipText("Save Document");
                  tb_save.setBounds((toolbar_item_num*toolbar_item_width) + 2 +
(7*toolbar_separator_num), 2,toolbar_item_width, toolbar_item_width);
                  tb_save.setMargin(new Insets(1,1,1,1));
                  tb_save.addActionListener(new ActionListener() { public void
actionPerformed(ActionEvent e) {
                        save();
                  }});
                  toolBar.add(tb_save);
                  toolbar_item_num++;
```

```
                JButton tb_saveas = new JButton(new ImageIcon("files/SaveAs.png"));
                tb_saveas.setToolTipText("Save Document As Copy");
                tb_saveas.setBounds((toolbar_item_num*toolbar_item_width) + 2 +
(7*toolbar_separator_num), 2,toolbar_item_width, toolbar_item_width);
                tb_saveas.setMargin(new Insets(1,1,1,1));
                tb_saveas.addActionListener(new ActionListener() { public void
actionPerformed(ActionEvent e) {
                      saveAs();
                }});
                toolBar.add(tb_saveas);
                toolbar_item_num++;

                ABSeparator s2 = new ABSeparator();
                s2.setBounds((toolbar_item_num*toolbar_item_width) + 5 +
(7*toolbar_separator_num), 6, 1, toolbar_item_width - 8);
                toolBar.add(s2);
                toolbar_separator_num++;

                JButton tb_import = new JButton(new ImageIcon("files/Datbase-
Add.png"));
                tb_import.setToolTipText("Import Contacts From A File");
                tb_import.setBounds(    (toolbar_item_num*toolbar_item_width) + 2 +
(7*toolbar_separator_num),
                                                    2,
                                                    toolbar_item_width,
                                                    toolbar_item_width);
                tb_import.setMargin(new Insets(1,1,1,1));
                tb_import.addActionListener(new ActionListener() {
                      public void actionPerformed(ActionEvent e) {
                            importContacts();
                      }
                });
                toolBar.add(tb_import);
                toolbar_item_num++;

                ABSeparator ss = new ABSeparator();
                ss.setBounds((toolbar_item_num*toolbar_item_width) + 5 +
(7*toolbar_separator_num), 6, 1, toolbar_item_width - 8);
                toolBar.add(ss);
                toolbar_separator_num++;

                tb_undo = new JButton(new ImageIcon("files/Left.png"));
                tb_undo.setToolTipText("Undo");
                tb_undo.setBounds((toolbar_item_num*toolbar_item_width) + 2 +
(7*toolbar_separator_num), 2,toolbar_item_width, toolbar_item_width);
                tb_undo.setMargin(new Insets(1,1,1,1));
                tb_undo.addActionListener(new ActionListener() { public void
actionPerformed(ActionEvent e) {
                      actionStack.pop();
                }});
                tb_undo.setEnabled(false);
                toolBar.add(tb_undo);
                toolbar_item_num++;

                tb_redo = new JButton(new ImageIcon("files/Right.png"));
                tb_redo.setToolTipText("Redo");
                tb_redo.setBounds((toolbar_item_num*toolbar_item_width) + 2 +
(7*toolbar_separator_num), 2,toolbar_item_width, toolbar_item_width);
                tb_redo.setMargin(new Insets(1,1,1,1));
                tb_redo.addActionListener(new ActionListener() { public void
actionPerformed(ActionEvent e) {
                      actionStack.push();
                }});
                tb_redo.setEnabled(false);
                toolBar.add(tb_redo);
                toolbar_item_num++;
```

```
                ABSeparator s = new ABSeparator();
                s.setBounds((toolbar_item_num*toolbar_item_width) + 5 +
(7*toolbar_separator_num), 6, 1, toolbar_item_width - 8);
                toolBar.add(s);
                toolbar_separator_num++;

                JButton tb_add = new JButton(new ImageIcon("files/User.png"));
                tb_add.setToolTipText("Add Contact");
                tb_add.setBounds( (toolbar_item_num*toolbar_item_width) + 2 +
(7*toolbar_separator_num),
                                                2,
                                                toolbar_item_width,
                                                toolbar_item_width);
                tb_add.setMargin(new Insets(1,1,1,1));
                final AddressBook ab = this;
                tb_add.addActionListener(new ActionListener() {
                      public void actionPerformed(ActionEvent e) {
                            ABDialogs.createAndShowAddDialog(ab);
                      }
                });
                toolBar.add(tb_add);
                toolbar_item_num++;

                JButton tb_del = new JButton(new ImageIcon("files/User-Del.png"));
                tb_del.setToolTipText("Remove Contact(s)");
                tb_del.setBounds( (toolbar_item_num*toolbar_item_width) + 2 +
(7*toolbar_separator_num),
                                                2,
                                                toolbar_item_width,
                                                toolbar_item_width);
                tb_del.setMargin(new Insets(1,1,1,1));
                tb_del.addActionListener(new ActionListener() {
                      public void actionPerformed(ActionEvent e) {
                            if (list.getSelectedIndex() != -1) {
                                  getActionStack().push(new Action() {
                                        public int index = list.getSelectedIndex();
                                        public Contact c = contacts.get(index);
                                        public void doAction() {
                                              contacts.remove(c);
                                              refreshList();
                                              int current_index = 0;
                                              list.setSelectedIndex(index);
current_index = index;
                                              if (index >= model.size()) {
                                                    list.setSelectedIndex(index - 1);
current_index = index - 1;
                                                    if (index - 1 >= model.size()) {
                                                          list.setSelectedIndex(index -
2); current_index = index - 2;
                                                          if (index - 2 >=
model.size()) {

      list.setSelectedIndex(0); current_index = 0;
                                                          }
                                                    }
                                              }
                                              if (model.size() > 0) {

      contactPanel.repopulate(contacts.get(current_index));
                                              }
                                        }
                                        public String getRedoText() {
                                              return "Delete " + c.getName();
                                        }
                                        public String getUndoText() {
                                              return "Undelete " + c.getName();
                                        }
                                        public void undoAction() {
```

```
                                        contacts.add(c);
                                        refreshList();
                                    }
                                });
                            }
                        }
                    });
                    toolBar.add(tb_del);
                    toolbar_item_num++;



                    ABSeparator s3 = new ABSeparator();
                    s3.setBounds((toolbar_item_num*toolbar_item_width) + 5 +
(7*toolbar_separator_num), 6, 1, toolbar_item_width - 8);
                    toolBar.add(s3);
                    toolbar_separator_num++;

                    int xOffset = 0;
                    JPanel pnl_search = new JPanel();
                    pnl_search.setBounds(   (toolbar_item_num*toolbar_item_width) + 2 +
(7*toolbar_separator_num),
                                            2,
                                            toolbar_item_width + 200,
                                            toolbar_item_width);
                    pnl_search.setLayout(null);
                    pnl_search.setBackground(Color.white);
                        JLabel lbl_icon = new JLabel(new ImageIcon("files/Search.png"));
                        lbl_icon.setBounds(0, 0, 32, pnl_search.getHeight());
                        lbl_icon.setBorder(new MatteBorder(1, 1, 1, 0,
Color.LIGHT_GRAY));
                        pnl_search.add(lbl_icon);

                        tf_search = new JTextField("");
                        tf_search.setBounds(30,0, 200 + toolbar_item_width - 30,
pnl_search.getHeight());
                        tf_search.setBackground(Color.white);
                        tf_search.setBorder(new MatteBorder(1, 0, 1, 1,
Color.LIGHT_GRAY));
                        tf_search.addKeyListener(new KeyListener() {
                            public void keyPressed(KeyEvent e) {}
                            public void keyReleased(KeyEvent e) {
                                setFilter(tf_search.getText());
                            }
                            public void keyTyped(KeyEvent e) {}
                        });
                        pnl_search.add(tf_search);
                    toolBar.add(pnl_search);
                    toolbar_item_num++;
                    xOffset += 200;

                    ABSeparator s4 = new ABSeparator();
                    s4.setBounds((toolbar_item_num*toolbar_item_width) + 5 +
(7*toolbar_separator_num) + xOffset, 6, 1, toolbar_item_width - 8);
                    toolBar.add(s4);
                    toolbar_separator_num++;

                    JButton tb_about = new JButton(new ImageIcon("files/Info.png"));
                    tb_about.setToolTipText("About Application");
                    tb_about.setBounds(     (toolbar_item_num*toolbar_item_width) + 2 +
(7*toolbar_separator_num) + xOffset, 2, toolbar_item_width, toolbar_item_width);
                    tb_about.setMargin(new Insets(1,1,1,1));
                    tb_about.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                            ABDialogs.createAndShowAboutDialog(getFrame());
                        }
                    });
                    toolBar.add(tb_about);
```

```
                toolbar_item_num++;

                JButton tb_exit = new JButton(new ImageIcon("files/Delete.png"));
                tb_exit.setToolTipText("Exit Application");
                tb_exit.setBounds(      (toolbar_item_num*toolbar_item_width) + 2 +
(7*toolbar_separator_num) + xOffset, 2, toolbar_item_width, toolbar_item_width);
                tb_exit.setMargin(new Insets(1,1,1,1));
                tb_exit.addActionListener(new ActionListener() {
                      public void actionPerformed(ActionEvent e) {
                            close();
                      }
                });
                toolBar.add(tb_exit);
                toolbar_item_num++;

                frame.add(toolBar, BorderLayout.NORTH);


          JSplitPane pane = new JSplitPane(JSplitPane.HORIZONTAL_SPLIT);
          pane.setDividerLocation(pane.getSize().width
                                      - pane.getInsets().right
                                      - pane.getDividerSize()
                                      - 180);


                JPanel listPanel = new JPanel();
                listPanel.setLayout(new GridLayout(1,1));

                model = new DefaultListModel();
                list = new JList(model);
                list.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
                list.addListSelectionListener(new ListSelectionListener() {
                      public void valueChanged(ListSelectionEvent e) {
                            try {
                                  int i = list.getSelectedIndex();
                                  if (getFilter().equals("")) {
                                        contactPanel.repopulate(contacts.get(i));
                                  } else {

      contactPanel.repopulate(filteredContacts.get(i));
                                  }
                                  frame.validate();
                                  // contactPanel.setVisibility(true);
                            } catch (ArrayIndexOutOfBoundsException ex) {
                                  //JOptionPane.showMessageDialog(frame, "")
                            }
                      }
                });

                listPanel.add(list);

                JScrollPane listPanelScroll = new JScrollPane(listPanel);
                listPanelScroll.setMinimumSize(new Dimension(180, 0));

      listPanelScroll.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
                pane.setLeftComponent(listPanelScroll);

                contactPanel = new ContactPanel(this);
                JScrollPane contactPanelScroll = new JScrollPane(contactPanel);
                contactPanelScroll.setMinimumSize(new Dimension(547, 0));


contactPanelScroll.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
                contactPanelScroll.setBorder(null);
                pane.setRightComponent(contactPanelScroll);

          frame.add(pane, BorderLayout.CENTER);
```

```
            actionStack = new ABActionStack(this);
            actionStack.setUI(m_undo, m_redo, tb_undo, tb_redo);


            frame.setVisible(true);
      }

      /**
       * This will set the search filter string and refresh the list.
       * @param text
       */
      public void setFilter(String text)
      {
            this.filter = text.toUpperCase();
            this.refreshList();
      }

      /**
       * return the search filter.
       * @return
       */
      public String getFilter() {
            return filter;
      }

      /**
       * Clear any existing contacts in the address book.
       * will ask the user if there are unsaved changes.
       */
      public void newFile() {
            if (this.isChangedAndUnsaved()) {
                  int response =
ABDialogs.createAndShowUnsavedChangesDialog(getFrame());

                  if (response == JOptionPane.CANCEL_OPTION) {
                        return;
                  } else if (response == JOptionPane.YES_OPTION) {
                        this.save();
                        if (this.save() == false) {
                              return;
                        }
                  }
            }
            this.clearList();
            this.currentlyOpenedFile = null;
            this.savedAtStackLocation = -1;
            this.actionStack.clear();
            this.actionStack.refreshUI();

            frame.setTitle("Address Book - Untitled Document");
            this.actionStack.refreshUI();
            refreshList();
      }

      /**
       * Opens the "import file" dialog where the user picks one or
       * more files to import.
       */
      public void importContacts() {
            File[] files = ABDialogs.createAndShowOpenFileSelector(getFrame(), true,
"Import Contacts From File: ");
            if (files != null) {
                  this.importContacts(files);
            }
      }

      /**
       * Imports all the contacts from the files into hte address book.
```

```
	 * Will ask the user what to do on duplicate contacts.
	 * @param files A collective of files to attempt to import.
	 */
	public void importContacts(File[] files)
	{
		int all_response = -1;
		boolean duplicateThisPass = false;
		for (int i = 0; i < files.length; i++) {

			if (!Importer.isValidExtension(files[i])) {
				ABDialogs.createAndShowInvalidExtensionDialog(getFrame(),
files[i]);
				continue;
			}
			Importer importer = new Importer(files[i]);
			ArrayList<Contact> contacts = importer.load();
			for (int j = 0; j < contacts.size(); j++) {
				for (int k = 0; k < this.contacts.size(); k++) {
					if
(contacts.get(j).getNameForSorting().equals(this.contacts.get(k).getNameForSorting()))
{
						duplicateThisPass = true;
						int response;
						if (all_response != -1) {
							response = all_response;
						} else {
							// ask the user what to do with the duplicate
contact!
							response =
ABDialogs.createAndShowImportDialog(this, this.frame, "Duplicate Contacts Found: ",
this.contacts.get(k), contacts.get(j), files[i]);
						}

						// DO TO ALL.
						if (response == ABDialogs.IMPORT_DIALOG_COMBINE_ALL)
{
							all_response =
ABDialogs.IMPORT_DIALOG_COMBINE_ONE;
						} else if (response ==
ABDialogs.IMPORT_DIALOG_REPLACE_ALL) {
							all_response =
ABDialogs.IMPORT_DIALOG_REPLACE_ONE;
						} else if (response ==
ABDialogs.IMPORT_DIALOG_KEEP_ALL) {
							// do nothing for all. :3
							all_response =
ABDialogs.IMPORT_DIALOG_KEEP_ONE;
						}

						// DO SINGULAR
						if (response == ABDialogs.IMPORT_DIALOG_COMBINE_ONE
|| all_response == ABDialogs.IMPORT_DIALOG_COMBINE_ONE) {
							// combine contacts.
							this.contacts.get(k).combine(contacts.get(j));
						} else if (response ==
ABDialogs.IMPORT_DIALOG_REPLACE_ONE || all_response ==
ABDialogs.IMPORT_DIALOG_REPLACE_ONE) {
							this.contacts.set(k, contacts.get(j));
						} else if (response ==
ABDialogs.IMPORT_DIALOG_KEEP_ONE || all_response == ABDialogs.IMPORT_DIALOG_KEEP_ONE) {
							// do nothing!
						}
						break;
					}
				}
				if (!duplicateThisPass) {
					this.contacts.add(contacts.get(j));
				}
```

```
                        duplicateThisPass = false;
                  }
            }
            this.refreshList();
      }

      /**
       * Close the Address Book safely asking the user what to
       * do with their unsaved changes.
       */
      public void close() {
            if (this.isChangedAndUnsaved()) {
                  int response =
ABDialogs.createAndShowUnsavedChangesDialog(getFrame());

                  if (response == JOptionPane.CANCEL_OPTION) {
                        return;
                  } else if (response == JOptionPane.YES_OPTION) {
                        this.save();
                        if (this.save() == false) {
                              return;
                        }
                  }
                  System.exit(0);
            } else {
                  System.exit(0);
            }
      }

      /**
       * Opens the "open file" dialog where the user picks
       * one file to open.
       */
      public void open()
      {
            File[] files = ABDialogs.createAndShowOpenFileSelector(getFrame(), false,
"Open File:");
            if (files != null) {
                  this.open(files[0]);
            }
      }

      /**
       * Opens the file specified and oimports all of the contacts in the file
       * into the address book. Will stop working with the previous file.
       * Will clear the existing address book entries.
       * @param f The file to open.
       */
      public void open(File f) {
            if (!Importer.isValidExtension(f)) {
                  ABDialogs.createAndShowInvalidExtensionDialog(getFrame(), f);
                  return;
            }
            if (this.isChangedAndUnsaved()) {
                  int response =
ABDialogs.createAndShowUnsavedChangesDialog(getFrame());

                  if (response == JOptionPane.CANCEL_OPTION) {
                        return;
                  } else if (response == JOptionPane.YES_OPTION) {
                        if (this.save() == false) {
                              return;
                        }
                  }
            }
            this.clearList();
            this.actionStack.clear();
            this.savedAtStackLocation = 0;
```

```
            this.currentlyOpenedFile = f;
            this.actionStack.refreshUI();

            Importer i = new Importer(f);
            frame.setTitle("Address Book - " + f.getAbsolutePath());
            this.addToList(i.load());
            if (!this.recentFileSet.isRecentFile(f.getAbsolutePath())) {
                  this.recentFileSet.add(f.getAbsolutePath());
            }

            this.refreshList();
      }


      /**
       * Determines whether the address book has unsaved changes.
       * @return whether the address book has unsaved changes.
       */
      public boolean isChangedAndUnsaved() {
            if (savedAtStackLocation != -1) {
                  if (savedAtStackLocation != this.actionStack.getTop()) {
                        return true;
                  }
            }
            return false;
      }


      /**
       * Saves the address book data to the currently opened file.
       * @return true on success, false on failure.
       */
      public boolean save() {
            if (this.currentlyOpenedFile == null) {
                  // no file is opened, so have to create one to save to!
                  this.saveAs();
            } else {
                  int response = this.checkForInformationNotStorableInBUAB();
                  if (response == JOptionPane.NO_OPTION) {
                        return false;
                  }
                  // file is opened.
                  if (!
this.recentFileSet.isRecentFile(this.currentlyOpenedFile.getAbsolutePath())) {

      this.recentFileSet.add(this.currentlyOpenedFile.getAbsolutePath());
                  }
                  frame.setTitle("Address Book - " +
this.currentlyOpenedFile.getAbsolutePath());
                  savedAtStackLocation = this.actionStack.getTop();
                  if (!this.currentlyOpenedFile.exists()) {
                        try {
                              this.currentlyOpenedFile.createNewFile();
                        } catch (IOException e1) {
                              JOptionPane.showMessageDialog(frame, "Cannot save
file.\r\nMake sure it is not in use by any other programs and try again.", "Error: ",
JOptionPane.ERROR_MESSAGE);
                              return false;
                        }
                  }

                  Exporter e = new Exporter(this.currentlyOpenedFile, this.contacts);
                  e.write();

            }
            return true;
      }
```

```
	/**
	 * Saves the address book data to a file specified by the user with a dialog.
	 */
	public void saveAs() {
		if (contacts.size() == 0) {
			JOptionPane.showMessageDialog(frame, "You are saving an empty address
book document.", "Warning:", JOptionPane.WARNING_MESSAGE);
		}
		File f = ABDialogs.createAndShowSaveFileSelector(frame);
		if (f != null) {
			this.currentlyOpenedFile = f;
			this.save();
		}
	}

	/**
	 * Checks whether the current address book data can be stored in a BUAB file.
	 * @return -1 if data is storable in buab and the JOptionPane response otherwise.
	 */
	public int checkForInformationNotStorableInBUAB() {
		if
(Importer.getFileExtension(this.currentlyOpenedFile).equals(Importer.BUAB)) {
			boolean valid = true;
			for (int i = 0; i < this.contacts.size(); i++) {
				if (this.contacts.get(i).hasInformationNotStorableInBUAB()) {
					valid = false;
				}
			}
			if (!valid) {
				int response = JOptionPane.showConfirmDialog(
														this.frame,
														"Some of your Address Book
Data cannot be stored in the BUAB format.\n" +
														"We advise that you save in
VCard format.\n" +
														"Are you sure that you want
to continue (and lose data)?",
														"Warning:",
														JOptionPane.YES_NO_OPTION,
														JOptionPane.WARNING_MESSAGE);
				return response;
			}
		}
		return -1;

	}

	/**
	 * Clears the current working list and contacts.
	 */
	public void clearList() {
		model.clear();
		contacts.clear();
	}

	/**
	 * Adds contacts to the address books UI.
	 * @param list
	 */
	public void addToList(ArrayList<Contact> list) {
		for (int i = 0; i < list.size(); i++) {
			contacts.add(list.get(i));
			model.addElement(" " + list.get(i).getForenames() + " " +
list.get(i).getSurname());
		}
	}

	/**
```

```
	 * Refreshes the address book list interface taking
	 * into account the currently set filter.
	 */
	public void refreshList() {
		model.clear();

		for (int i = 0; i < this.contacts.size(); i++) {
			for (int j = this.contacts.size()-1; j > i; j--) {
				if
(this.contacts.get(i).getNameForSorting().compareTo(this.contacts.get(j).getNameForSort
ing()) > 0) {
					Contact c = this.contacts.get(j);
					this.contacts.set(j, this.contacts.get(i));
					this.contacts.set(i, c);
				}
			}
		}

		if (this.filter.equals("")) {
			// no filter.
			//this.contacts.addAll(this.filteredContacts);
			for (int i = 0; i < this.contacts.size(); i++) {
				model.addElement(" " + this.contacts.get(i).getForenames() + " "
+ this.contacts.get(i).getSurname());
			}
		} else {
			// apply filter
			filteredContacts.clear();
			unfilteredContacts.clear();
			String[] filter_items = filter.split(",");
			for (int i = 0; i < this.contacts.size(); i++) {
				for (int j = 0; j < filter_items.length; j++) {
					if (!this.filteredContacts.contains(this.contacts.get(i)))
{
						if
(this.contacts.get(i).toSearchString().toUpperCase().contains(filter_items[j])) {

	this.filteredContacts.add(this.contacts.get(i));
						} else {

	this.unfilteredContacts.add(this.contacts.get(i));
						}
					}
				}
			}
			ArrayList<Contact> remove = new ArrayList<Contact>();
			for (int i = 0; i < this.filteredContacts.size(); i++) {
				for (int j = 0; j < filter_items.length; j++) {
					if (!
this.filteredContacts.get(i).toSearchString().toUpperCase().contains(filter_items[j]))
{
						remove.add(this.filteredContacts.get(i));
						break;
					}
				}
			}
			this.filteredContacts.removeAll(remove);
			for (int i = 0; i < this.filteredContacts.size(); i++) {
			//
	System.out.println(this.filteredContacts.get(i).toSearchString().toUpperCase());
				model.addElement(" " +
this.filteredContacts.get(i).getForenames() + " " +
this.filteredContacts.get(i).getSurname());
			}
		}

		if (this.contacts.size() == 0) {
			this.contactPanel.repopulate(null);
```

```
            }
        }

        /**
         * Get the contact panel
         * @return the address books contact panel.
         */
        public ContactPanel getContactPanel() {
            return contactPanel;
        }

        /**
         * get the undo/redo stack.
         * @return the undo/redo action stack.
         */
        public ActionStack getActionStack() {
            return actionStack;
        }

        /**
         * get the address books frame.
         * @return the address book's frame.
         */
        public JFrame getFrame() {
            return frame;
        }

        /**
         * The address books stack location
         * @return -1 if document is not open.
         */
        public int getSavedAtStackLocation() {
            return savedAtStackLocation;
        }

        /**
         * get the file that is currently "open".
         * @return the file that is currently "open".
         */
        public File getCurrentlyOpenedFile() {
            return currentlyOpenedFile;
        }

        /**
         * a list of all of the contacts that are currently in the address book.
         * @return a list of all of the contacts that are currently in the address book.
         */
        public ArrayList<Contact> getContacts() {
            return contacts;
        }

        /**
         * Entry point to the program with any command line arguments.
         * @param args Command line parameters.
         */
        public static void main(String[] args)
        {
            AddressBook a = new AddressBook();
        }
}
```

```
package org.ag.addressbook;

import java.util.ArrayList;

import org.ag.addressbook.property.Address;
import org.ag.addressbook.property.EmailAddress;
import org.ag.addressbook.property.TelephoneNumber;
```

```java
/**
 * Contact represents a real-world person with their details.
 * The properties of the contact allow export to the vCard standard
 * and probably LDAP.
 *
 * @author Ashley Gwinnell
 */
public class Contact
{
	private String forenames;
	private String surname;
	private String prefixes;
	private String suffixes;

	private ArrayList<EmailAddress> emailAddresses = new ArrayList<EmailAddress>();
	private ArrayList<TelephoneNumber> telephoneNumbers = new
ArrayList<TelephoneNumber>();
	private ArrayList<Address> addresses = new ArrayList<Address>();

	/**
	 * Create a new contact with no information specified.
	 * Use getters/setters.
	 */
	public Contact() {
		this.forenames = new String();
		this.surname = new String();
		this.prefixes = new String();
		this.suffixes = new String();
	}

	/**
	 * Sets the name of the contact.
	 * @param fullname name of the contact.
	 */
	public void setName(String fullname) {
		String[] names = fullname.split(" ");
		if (names.length == 1) {
			this.setForenames(fullname);
			return;
		}
		for (int i = 0; i < names.length-1; i++) {
			this.addForename(names[i]);
		}
		this.setSurname(names[names.length-1]);
	}
	/**
	 * Gets the name of the contact.
	 * @return the name of the contact.
	 */
	public String getName() {
		return (prefixes.trim() + " " + forenames.trim() + " " + surname.trim() + "
" + suffixes.trim()).trim();
	}

	/**
	 * Gets the name of the contact for sorting in the JList.
	 * @return the name of the contact for sorting in the JList.
	 */
	public String getNameForSorting() {
		return surname.trim().toUpperCase() + ", " + forenames.trim().toUpperCase();
	}

	/**
	 * Gets the contacts forenames.
	 * @return the contact's forenames.
	 */
	public String getForenames() {
```

```
		return forenames.trim();
	}

	/**
	 * Gets hte contacts forenames separated by the parameter given.
	 * @param delimiter the forename separation string.
	 * @return the contact's forenames separated by delimiter.
	 */
	public String getForenames(String delimiter) {
		String name = "";
		String[] names = forenames.trim().split(" ");
		for (int i = 0; i < names.length; i++) {
			name += names[i] + delimiter;
		}
		return name;
	}

	/**
	 * Set the contact's forename string.
	 * @param forenames the contact's forename string.
	 */
	public void setForenames(String forenames) {
		this.forenames = forenames.trim();
	}

	/**
	 * Add a name to the contact's forename string.
	 * @param forename a name to add to the contacts forenames.
	 */
	public void addForename(String forename) {
		this.forenames += forename.trim() + " ";
	}

	/**
	 * Gets the contact's surname.
	 * @return the contact's surname.
	 */
	public String getSurname() {
		return surname;
	}

	/**
	 * Sets the contact's surname.
	 * @param surname the contact's surname specified.
	 */
	public void setSurname(String surname) {
		this.surname = surname.trim();
	}

	private String getForenames(String splitDelimiter, String joinDelimiter, int
startIndex, int length) {
		String name = "";
		String[] names = this.forenames.trim().split(splitDelimiter);
		for (int i = startIndex; i < startIndex+length; i++) {
			name += names[i] + joinDelimiter;
		}
		return name;
	}

	/**
	 * Given name is another name for a contacts first name, singular.
	 * This is used in the vCard implementation.
	 * @return Given name is another name for a contacts first name, singular.
	 */
	public String getGivenName() {
		return this.getForenames(" ", "", 0, 1);
	}
```

```
	/**
	 * Get's the contact's surname aka family name in the vCard specification.
	 * @return the contact's surname aka family name
	 */
	public String getFamilyName() {
		return this.surname.trim();
	}

	/**
	 * Gets the contact's additional forenames.
	 * Additional forenames are names that are neither forename or surname,
	 * such as middle names.
	 * @return the contact's additional forenames.
	 */
	public String getAdditionalNames() {
		return this.getForenames(" ", ",", 1, this.forenames.trim().split("
").length-1);
	}

	/**
	 * The contact's honorable suffixies, eg. BSc.
	 * @return The contact's honorable suffixies
	 */
	public String getSuffixes() {
		return suffixes;
	}

	/**
	 * Add an honorable suffix to a contact.
	 * @param suffix the honorable suffix.
	 */
	public void addSuffix(String suffix) {
		this.addSuffix(suffix, ",");
	}

	/**
	 * Add an honorable suffix to a contact using a joinString.
	 * @param suffix the honorable suffix
	 * @param joinString the join string, typically a comma.
	 */
	public void addSuffix(String suffix, String joinString) {
		this.suffixes += suffix + joinString;
	}

	/**
	 * Get the honorable prefixes for the contact, eg. Dr.
	 * @return
	 */
	public String getPrefixes() {
		return prefixes;
	}

	/**
	 * Add an honorable prefix to a contact.
	 * @param prefix the honorable prefix.
	 */
	public void addPrefix(String prefix) {
		this.addPrefix(prefix, ",");
	}

	/**
	 * Add an honorable prefix to a contact with a joining string.
	 * @param prefix the honorable prefix
	 * @param joinString the join string typically a comma.
	 */
	public void addPrefix(String prefix, String joinString) {
		this.prefixes += prefix + joinString;
	}
```

```java
/**
 * Add an email address to a contact.
 * @param e the EmailAddress Object to add.
 */
public void addEmailAddress(EmailAddress e) {
	this.emailAddresses.add(e);
}

/**
 * Get a list of the contact's email addreses.
 * @return a list of the contact's email addresses.
 */
public ArrayList<EmailAddress> getEmailAddresses() {
	return this.emailAddresses;
}

/**
 * Remove an email address from a contact.
 * @param e the EmailAddress Object to remove.
 */
public void removeEmailAddress(EmailAddress e) {
	this.emailAddresses.remove(e);
}

/**
 * Add an address to the contact.
 * @param e the Address object to give the contact.
 */
public void addAddress(Address e) {
	this.addresses.add(e);
}

/**
 * Get a ist of the contact's Addresses.
 * @return a list of the contact's Addresses.
 */
public ArrayList<Address> getAddresses() {
	return this.addresses;
}

/**
 * Remove an Address from teh contact.
 * @param a the Address Object to remove.
 */
public void removeAddress(Address a) {
	this.addresses.remove(a);
}

/**
 * Add a TelephoneNumber to a contact.
 * @param e the TelephoneNumber to add.
 */
public void addTelephoneNumber(TelephoneNumber e) {
	this.telephoneNumbers.add(e);
}

/**
 * Get a list of the contact's TelephoneNumbers.
 * @return a list of the contact's TelephoneNumbers.
 */
public ArrayList<TelephoneNumber> getTelephoneNumbers() {
	return this.telephoneNumbers;
}

/**
 * Remove a TelephoneNumber from a contact.
 * @param a the TelephoneNumber to remove.
```

```
	 */
	public void removeTelephoneNumber(TelephoneNumber a) {
		this.telephoneNumbers.remove(a);
	}

	/**
	 * Combine two contacts into one.
	 * This is used when importing and there are duplicate contacts.
	 * @param c the Contact to combine/merge with.
	 */
	public void combine(Contact c) {
		this.addresses.addAll(c.getAddresses());
		this.telephoneNumbers.addAll(c.getTelephoneNumbers());
		this.emailAddresses.addAll(c.getEmailAddresses());
	}

	/**
	 * Determines whether the contact has information about it that
	 * cannot be stored in the BUAB file format.
	 * @return whether the contact has information about it that
	 *             cannot be stored in the BUAB file format.
	 */
	public boolean hasInformationNotStorableInBUAB() {
		if (this.emailAddresses.size() > 0
				|| this.telephoneNumbers.size() > 2
				|| this.addresses.size() > 1) {
			return true;
		} else {
			for (int i = 0; i < this.addresses.size(); i++) {
				Address a = this.addresses.get(i);
				if (	a.getPOBoxNumber().trim().length() > 0
					||	a.getExtendedAddress().trim().length() > 0
					||	a.getCity().trim().length() > 0
					||	a.getCounty().trim().length() > 0
					||	a.getCountry().trim().length() > 0
					||	a.getPostcode().trim().length() > 0) {
					return true;
				}
			}
			return false;
		}
	}


	/**
	 * Get the search String for this contact. this will be searched when using Quick
Search.
	 * @return the search String for this contact. used in quick search.
	 */
	public String toSearchString() {
		String searchString = new String("");
		searchString += prefixes + " " + forenames + " " + surname + " " + suffixes
+ " ";
		for (int i = 0; i < addresses.size(); i++) {
			searchString += addresses.get(i).toSearchString() + " ";
		}
		for (int i = 0; i < telephoneNumbers.size(); i++) {
			searchString += telephoneNumbers.get(i).toSearchString() + " ";
		}
		for (int i = 0; i < emailAddresses.size(); i++) {
			searchString += emailAddresses.get(i).toSearchString() + " ";
		}
		return searchString;
	}
}

package org.ag.addressbook;
```

```
import java.awt.Desktop;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.FocusEvent;
import java.awt.event.FocusListener;
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;
import java.net.URI;
import java.util.ArrayList;

import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;

import org.ag.addressbook.property.Address;
import org.ag.addressbook.property.EmailAddress;
import org.ag.addressbook.property.TelephoneNumber;
import org.ag.util.undoredo.Action;

/**
 * ContactPanel
 * This panel is made to refresh or "repopulate" when the selected contact changes.
 * The vast number of ArrayLists are needed to edit the correct property of the
contact.
 * @author Ashley Gwinnell
 */
public class ContactPanel extends JPanel
{
	// container?
	private AddressBook addressBook;

	// stuff for contact informationz
	private Contact currentContact;
	private JLabel lbl_fullname;

	private ArrayList<JTextField> addresses_po_box = new ArrayList<JTextField>();
	private ArrayList<JTextField> addresses_extended_address = new
ArrayList<JTextField>();
	private ArrayList<JTextArea>  addresses_street_address = new
ArrayList<JTextArea>();
	private ArrayList<JTextField> addresses_locality_city = new
ArrayList<JTextField>();
	private ArrayList<JTextField> addresses_region_state_province_county = new
ArrayList<JTextField>();
	private ArrayList<JTextField> addresses_postcode = new ArrayList<JTextField>();
	private ArrayList<JTextField> addresses_country = new ArrayList<JTextField>();
	private ArrayList<JButton> addresses_set_po_box = new ArrayList<JButton>();
	private ArrayList<JButton> addresses_set_extended_address = new
ArrayList<JButton>();

	private ArrayList<JComboBox> addresses_type = new ArrayList<JComboBox>();
	private ArrayList<JButton> addresses_delete = new ArrayList<JButton>();

	private ArrayList<JTextField> telephoneNumbers = new ArrayList<JTextField>();
	private ArrayList<JComboBox> telephoneNumbers_type = new ArrayList<JComboBox>();
	private ArrayList<JButton> telephoneNumbers_delete = new ArrayList<JButton>();
```

```
	private ArrayList<JTextField> emailAddresses = new ArrayList<JTextField>();
	private ArrayList<JComboBox> emailAddresses_type = new ArrayList<JComboBox>();
	private ArrayList<JButton> emailAddresses_mailto = new ArrayList<JButton>();
	private ArrayList<JButton> emailAddresses_delete = new ArrayList<JButton>();

	// stuff for layout of contact informations!
	private JPanel pnl_fullname;
	private JPanel pnl_addresses;
	private JPanel pnl_telephoneNumbers;
	private JPanel pnl_emailAddresses;
	private JPanel pnl_buttons;

	// buttons
	private JButton btn_add_address;
	private JButton btn_add_telephoneNumber;
	private JButton btn_add_emailAddress;
	private JButton btn_edit_name;

	// width of general thing
	private final int w = 500;

	/**
	 * Create a new ContactPanel.
	 * TODO: Make it so the information is aligned at the top of it's parent.
	 *           setAlignmentX and setAlignmentY don't seem to work.
	 * @param addressbook The AddressBook instance.
	 */
	public ContactPanel(AddressBook addressbook)
	{
		this.addressBook = addressbook;
		this.setAlignmentX(LEFT_ALIGNMENT);
		this.setAlignmentY(TOP_ALIGNMENT);
		GridBagConstraints c = new GridBagConstraints();
		this.setLayout(new GridBagLayout());

		c.gridx = 0;
		c.gridy = 0;

		pnl_fullname = new JPanel();
		pnl_fullname.setPreferredSize(new Dimension(w, 50));
		pnl_fullname.setLayout(null);
			lbl_fullname = new JLabel("<html><big>Address Book</big><br/><b>To set
started, open a file or add a contact.</b></html>");
			lbl_fullname.setBounds(0, 0, 400, 50);
			pnl_fullname.add(lbl_fullname);
		this.add(pnl_fullname, c);

		c.gridx = 0;
		c.gridy = 1;
		pnl_addresses = new JPanel();
		pnl_addresses.setPreferredSize(new Dimension(w, 20));
		pnl_addresses.setLayout(null);
		pnl_addresses.setVisible(false);
		this.add(pnl_addresses, c);

		c.gridx = 0;
		c.gridy = 2;
		pnl_telephoneNumbers = new JPanel();
		pnl_telephoneNumbers.setPreferredSize(new Dimension(w, 20));
		pnl_telephoneNumbers.setLayout(null);
		pnl_telephoneNumbers.setVisible(false);
		this.add(pnl_telephoneNumbers, c);

		c.gridx = 0;
		c.gridy = 3;
		pnl_emailAddresses = new JPanel();
		pnl_emailAddresses.setPreferredSize(new Dimension(w, 20));
```

```
		pnl_emailAddresses.setLayout(null);
		pnl_emailAddresses.setVisible(false);
		this.add(pnl_emailAddresses, c);


		c.gridx = 0;
		c.gridy = 4;
		pnl_buttons = new JPanel();
		pnl_buttons.setPreferredSize(new Dimension(w, 60));
		pnl_buttons.setLayout(null);
		JLabel lbl_settings = new JLabel("Settings: ");
		lbl_settings.setBounds(0, 5, 100, 25);
		pnl_buttons.add(lbl_settings);
		btn_edit_name = new JButton("Edit Contact's Name");
		btn_edit_name.setBounds(150, 5, 200, 25);
		btn_edit_name.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				createAndShowEditNameDialog(addressBook.getFrame());
			}
		});
		pnl_buttons.add(btn_edit_name);
		pnl_buttons.setVisible(false);
		this.add(pnl_buttons, c);



	}

	@Override
	public void setEnabled(boolean enabled) {
		super.setEnabled(enabled);
		for (int i = 0; i < addresses_type.size(); i++) {
			try { addresses_po_box.get(i).setEnabled(enabled); } catch
(IndexOutOfBoundsException e) { }
			try { addresses_extended_address.get(i).setEnabled(enabled); } catch
(IndexOutOfBoundsException e) { }
			try { addresses_set_po_box.get(i).setVisible(enabled); } catch
(IndexOutOfBoundsException e) { }
			try { addresses_set_extended_address.get(i).setVisible(enabled); }
catch (IndexOutOfBoundsException e) { }
			addresses_street_address.get(i).setEnabled(enabled);
			addresses_locality_city.get(i).setEnabled(enabled);
			addresses_region_state_province_county.get(i).setEnabled(enabled);
			addresses_postcode.get(i).setEnabled(enabled);
			addresses_country.get(i).setEnabled(enabled);
			addresses_type.get(i).setEnabled(enabled);
			addresses_delete.get(i).setVisible(enabled);
		}
		for (int i = 0; i < telephoneNumbers.size(); i++) {
			telephoneNumbers.get(i).setEnabled(enabled);
			telephoneNumbers_type.get(i).setEnabled(enabled);
			telephoneNumbers_delete.get(i).setVisible(enabled);
		}
		for (int i = 0; i < emailAddresses.size(); i++) {
			emailAddresses.get(i).setEnabled(enabled);
			emailAddresses_type.get(i).setEnabled(enabled);
			emailAddresses_mailto.get(i).setVisible(enabled);
			emailAddresses_delete.get(i).setVisible(enabled);
		}
		btn_add_address.setVisible(enabled);
		btn_add_telephoneNumber.setVisible(enabled);
		btn_add_emailAddress.setVisible(enabled);

		pnl_buttons.setVisible(enabled);
		btn_edit_name.setVisible(enabled);
	}

	/**
```

```
	 * This should be called on every change to a contact's details.
	 * Whether something is deleted, or just the selected contact changes.
	 * This includes in the undo/redo stack.
	 * @param c
	 */
	public void repopulate(Contact c)
	{
		addresses_po_box.clear();
		addresses_extended_address.clear();
		addresses_street_address.clear();
		addresses_locality_city.clear();
		addresses_region_state_province_county.clear();
		addresses_postcode.clear();
		addresses_country.clear();
		addresses_type.clear();
		addresses_delete.clear();
		telephoneNumbers.clear();
		telephoneNumbers_type.clear();
		telephoneNumbers_delete.clear();
		emailAddresses.clear();
		emailAddresses_type.clear();
		emailAddresses_mailto.clear();
		emailAddresses_delete.clear();
		pnl_addresses.removeAll();
		pnl_telephoneNumbers.removeAll();
		pnl_emailAddresses.removeAll();

		this.currentContact = c;

		if (c == null) {
			lbl_fullname.setText("<html><big>Address Book</big><br/><b>To set
started, open a file or add a contact.</b></html>");
			pnl_addresses.setVisible(false);
			pnl_telephoneNumbers.setVisible(false);
			pnl_emailAddresses.setVisible(false);
			pnl_buttons.setVisible(false);
			return;
		} else {
			lbl_fullname.setText("<html><big>" + c.getForenames() + " " +
c.getSurname() + "</big></html>");
			pnl_addresses.setVisible(true);
			pnl_telephoneNumbers.setVisible(true);
			pnl_emailAddresses.setVisible(true);
			pnl_buttons.setVisible(true);
		}
		//pnl_fullname.setPreferredSize(new Dimension(400, 35));
		//lbl_fullname.setBounds(0, 0, 400, 30);



		// fill addresses! :)
		JLabel lbl_addresses = new JLabel("Addresses: ");
		lbl_addresses.setBounds(0, 0, 120, 20);
		pnl_addresses.add(lbl_addresses);
		int current_x = 150;
		int current_y = 0;
		for (int i = 0; i < c.getAddresses().size(); i++) {
			final Address a = c.getAddresses().get(i);


			String[] types = {"Home", "Work"};
			final JComboBox box = new JComboBox(types);
			box.addFocusListener(new FocusListener() {
				int valueOnFocus;
				public Contact c;
				public void focusGained(FocusEvent e) {
					valueOnFocus = box.getSelectedIndex();
					c = currentContact;
```

```
                        }
                        public void focusLost(FocusEvent e) {
                              if (box.getSelectedIndex() != valueOnFocus) { // changes!!

                                    addressBook.getActionStack().push(new Action() {
                                          public Address ad = a;
                                          public int adtp = valueOnFocus;
                                          public int adt = box.getSelectedIndex();
                                          public boolean secondPass = false;
                                          public void doAction() {
                                                if (adt == 0) {
                                                      ad.setType(Address.Type.HOME);
                                                } else if (adt == 1) {
                                                      ad.setType(Address.Type.WORK);
                                                }
                                                if (secondPass) {
addressBook.getContactPanel().repopulate(c); }
                                                addressBook.getFrame().validate();
                                                secondPass = true;
                                          }
                                          public String getRedoText() {
                                                if (adt == 0) {
                                                      return "Set Address Type As
'Home'";
                                                } else if (adt == 1) {
                                                      return "Set Address Type As
'Work'";
                                                }
                                                return "BAD BAD BAD";
                                          }
                                          public String getUndoText() {
                                                if (adtp == 0) {
                                                      return "Set Address Type As
'Home'";
                                                } else if (adtp == 1) {
                                                      return "Set Address Type As
'Work'";
                                                }
                                                return "BAD BAD BAD!";
                                          }
                                          public void undoAction() {
                                                if (adtp == 0) {
                                                      ad.setType(Address.Type.HOME);
                                                } else if (adtp == 1) {
                                                      ad.setType(Address.Type.WORK);
                                                }

      addressBook.getContactPanel().repopulate(c);
                                                addressBook.getFrame().validate();
                                          }
                                    });
                              }
                        }
                  });
                  box.setBounds(current_x + 205, current_y, 100, 25);
                  if (a.getType().equals(Address.Type.HOME)) {
                        box.setSelectedIndex(0);
                  } else if (a.getType().equals(Address.Type.WORK)) {
                        box.setSelectedIndex(1);
                  }

                  JButton delete = new JButton(new ImageIcon("files/Delete.png"));
                  delete.setToolTipText("Remove Address");
                  delete.setBounds(current_x + 205 + 105, current_y, 25, 25);
                  delete.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                              addressBook.getActionStack().push(new Action() {
                                    public Contact c = currentContact;
```

```
                                    public Address ad = a;
                                    public void doAction() {
                                          c.removeAddress(ad);

      addressBook.getContactPanel().repopulate(currentContact);
                                          addressBook.getFrame().validate();
                                    }
                                    public String getRedoText() {
                                          return "Remove Address from " + c.getName();
                                    }
                                    public String getUndoText() {
                                          return "Add Address to " + c.getName();
                                    }
                                    public void undoAction() {
                                          c.addAddress(ad);

      addressBook.getContactPanel().repopulate(currentContact);
                                          addressBook.getFrame().validate();
                                    }
                              });
                        }
                  });


                  //addresses.add(area);
                  addresses_type.add(box);
                  addresses_delete.add(delete);

                  //pnl_addresses.add(pane);
                  pnl_addresses.add(box);
                  pnl_addresses.add(delete);

                  if (!c.getAddresses().get(i).getPOBoxNumber().trim().equals("")) {
                        JLabel lbl_number = new JLabel("PO Box #: ");
                        lbl_number.setBounds(current_x - 70, current_y, 80, 20);
                        pnl_addresses.add(lbl_number);

                        final JTextField po_box = new JTextField();
                        po_box.addFocusListener(null);
                        po_box.setBounds(current_x, current_y, 50, 25);
                        po_box.setText(c.getAddresses().get(i).getPOBoxNumber());
                        po_box.addFocusListener(new FocusListener() {
                              public String valueOnFocus;
                              public Contact c;
                              public void focusGained(FocusEvent e) {
                                    c = currentContact;
                                    valueOnFocus = po_box.getText();
                              }
                              public void focusLost(FocusEvent e) {
                                    if (!po_box.getText().equals(valueOnFocus)) {
                                          addressBook.getActionStack().push(new Action()
{
                                                public Address ad = a;
                                                public String ad_str_previous =
valueOnFocus;

                                                public String ad_str = po_box.getText();
                                                public boolean secondPass = false;
                                                public void doAction() {
                                                      ad.setPOBoxNumber(ad_str);
                                                      if (secondPass) {
addressBook.getContactPanel().repopulate(c); }
                                                      addressBook.getFrame().validate();
                                                      secondPass = true;
                                                }
                                                public String getRedoText() {
                                                      return "Change PO Box Number";
                                                }
                                                public String getUndoText() {
```

```
                                                      return "Change PO Box Number";
                                                }
                                                public void undoAction() {

      ad.setPOBoxNumber(ad_str_previous);

      addressBook.getContactPanel().repopulate(c);
                                                      addressBook.getFrame().validate();
                                                }
                                          });
                                    }
                              }
                        });

                              addresses_po_box.add(po_box);
                              pnl_addresses.add(po_box);
                              current_y += 30;
                        } else {
                              int x = addresses_type.get(i).getX();
                              int y = addresses_type.get(i).getY() + 28;
                              JButton button = new JButton("Set POBox Number");
                              button.setToolTipText("Set A PO BOX Number for this address.");
                              button.setBounds(x, y, 130, 25);
                              button.addActionListener(new ActionListener() {
                                    public Address ad = a;
                                    public void actionPerformed(ActionEvent e) {

      createAndShowSetPOBoxNumberDialog(addressBook.getFrame(), ad);
                                    }
                              });
                              addresses_set_po_box.add(button);
                              pnl_addresses.add(button);
                        }

                        if (!c.getAddresses().get(i).getExtendedAddress().trim().equals("")) {
                              JLabel lbl_extended_address = new JLabel("Business: ");
                              lbl_extended_address.setBounds(current_x - 70, current_y, 60,
20);
                              pnl_addresses.add(lbl_extended_address);

                              final JTextField extended_address = new JTextField();
                              extended_address.addFocusListener(null);
                              extended_address.setBounds(current_x, current_y, 200, 25);

      extended_address.setText(c.getAddresses().get(i).getExtendedAddress());
                              extended_address.addFocusListener(new FocusListener() {
                                    public String valueOnFocus;
                                    public Contact c;
                                    public void focusGained(FocusEvent e) {
                                          c = currentContact;
                                          valueOnFocus = extended_address.getText();
                                    }
                                    public void focusLost(FocusEvent e) {
                                          if (!
extended_address.getText().equals(valueOnFocus)) {
                                                addressBook.getActionStack().push(new Action()
{
                                                      public Address ad = a;
                                                      public String ad_str_previous =
valueOnFocus;
                                                      public String ad_str =
extended_address.getText();
                                                      public boolean secondPass = false;
                                                      public void doAction() {
                                                            ad.setExtendedAddress(ad_str);
                                                            if (secondPass) {
addressBook.getContactPanel().repopulate(c); }
                                                            addressBook.getFrame().validate();
```

```
                                                secondPass = true;
                                          }
                                          public String getRedoText() {
                                                return "Change Business Name";
                                          }
                                          public String getUndoText() {
                                                return "Change Business Name";
                                          }
                                          public void undoAction() {

      ad.setExtendedAddress(ad_str_previous);

      addressBook.getContactPanel().repopulate(c);
                                                addressBook.getFrame().validate();
                                          }
                                    });
                              }
                        }
                  });
                  addresses_extended_address.add(extended_address);
                  pnl_addresses.add(extended_address);
                  current_y += 30;
            } else {
                  int x = addresses_type.get(i).getX();
                  int y = addresses_type.get(i).getY() + 28;
                  if (c.getAddresses().get(i).getPOBoxNumber().trim().equals(""))
{ y += 28; }
                  JButton button = new JButton("Set Business Name");
                  button.setToolTipText("Set A Business Name for this address.");
                  button.setBounds(x, y, 130, 25);
                  button.addActionListener(new ActionListener() {
                        public Address ad = a;
                        public void actionPerformed(ActionEvent e) {

      createAndShowSetExtendedAddressDialog(addressBook.getFrame(), ad);
                        }
                  });
                  addresses_set_extended_address.add(button);
                  pnl_addresses.add(button);
            }

            JLabel lbl_street_address = new JLabel("Lines: ");
            lbl_street_address.setBounds(current_x - 70, current_y, 60, 20);
            pnl_addresses.add(lbl_street_address);

            final JTextArea street_address = new JTextArea();
            street_address.setFont(new Font("Arial", Font.PLAIN, 12));
            street_address.setLineWrap(true);
            street_address.setWrapStyleWord(true);
            street_address.addFocusListener(null);
            street_address.setText(c.getAddresses().get(i).getStreetAddress());
            street_address.addFocusListener(new FocusListener() {
                  public String valueOnFocus;
                  public Contact c;
                  public void focusGained(FocusEvent e) {
                        c = currentContact;
                        valueOnFocus = street_address.getText();
                  }
                  public void focusLost(FocusEvent e) {
                        if (!street_address.getText().equals(valueOnFocus)) {
                              addressBook.getActionStack().push(new Action() {
                                    public Address ad = a;
                                    public String ad_str_previous = valueOnFocus;
                                    public String ad_str =
street_address.getText();
                                    public boolean secondPass = false;
                                    public void doAction() {
                                          ad.setStreetAddress(ad_str);
```

```
                                                if (secondPass) {
addressBook.getContactPanel().repopulate(c); }
                                                addressBook.getFrame().validate();
                                                secondPass = true;
                                          }
                                          public String getRedoText() {
                                                return "Change Street Address";
                                          }
                                          public String getUndoText() {
                                                return "Change Street Address";
                                          }
                                          public void undoAction() {
                                                ad.setStreetAddress(ad_str_previous);

      addressBook.getContactPanel().repopulate(c);
                                                addressBook.getFrame().validate();
                                          }
                                    });
                              }
                        }
                  });

                  JScrollPane street_address_scroll = new JScrollPane(street_address);
                  street_address_scroll.setBounds(current_x, current_y, 200, 70);


                  addresses_street_address.add(street_address);
                  pnl_addresses.add(street_address_scroll);
                  current_y += 75;

                  JLabel lbl_city = new JLabel("City: ");
                  lbl_city.setBounds(current_x - 70, current_y, 60, 20);
                  pnl_addresses.add(lbl_city);

                  final JTextField city = new JTextField();
                  city.addFocusListener(new FocusListener() {
                        public String valueOnFocus;
                        public Contact c;
                        public void focusGained(FocusEvent e) {
                              c = currentContact;
                              valueOnFocus = city.getText();
                        }
                        public void focusLost(FocusEvent e) {
                              if (!city.getText().equals(valueOnFocus)) {
                                    addressBook.getActionStack().push(new Action() {
                                          public Address ad = a;
                                          public String ad_str_previous = valueOnFocus;
                                          public String ad_str = city.getText();
                                          public boolean secondPass = false;
                                          public void doAction() {
                                                ad.setCity(ad_str);
                                                if (secondPass) {
addressBook.getContactPanel().repopulate(c); }
                                                addressBook.getFrame().validate();
                                                secondPass = true;
                                          }
                                          public String getRedoText() {
                                                return "Change City";
                                          }
                                          public String getUndoText() {
                                                return "Change City";
                                          }
                                          public void undoAction() {
                                                ad.setCity(ad_str_previous);

      addressBook.getContactPanel().repopulate(c);
                                                addressBook.getFrame().validate();
                                          }
```

```
                                    });
                              }
                        }
                  });
                  city.setBounds(current_x, current_y, 200, 25);
                  city.setText(c.getAddresses().get(i).getCity());

                  addresses_locality_city.add(city);
                  pnl_addresses.add(city);
                  current_y += 30;

                  JLabel lbl_county = new JLabel("County: ");
                  lbl_county.setBounds(current_x - 70, current_y, 60, 20);
                  pnl_addresses.add(lbl_county);

                  final JTextField county = new JTextField();
                  county.addFocusListener(new FocusListener() {
                        public String valueOnFocus;
                        public Contact c;
                        public void focusGained(FocusEvent e) {
                              c = currentContact;
                              valueOnFocus = county.getText();
                        }
                        public void focusLost(FocusEvent e) {
                              if (!county.getText().equals(valueOnFocus)) {
                                    addressBook.getActionStack().push(new Action() {
                                          public Address ad = a;
                                          public String ad_str_previous = valueOnFocus;
                                          public String ad_str = county.getText();
                                          public boolean secondPass = false;
                                          public void doAction() {
                                                ad.setCounty(ad_str);
                                                if (secondPass) {
addressBook.getContactPanel().repopulate(c); }
                                                addressBook.getFrame().validate();
                                                secondPass = true;
                                          }
                                          public String getRedoText() {
                                                return "Change County";
                                          }
                                          public String getUndoText() {
                                                return "Change County";
                                          }
                                          public void undoAction() {
                                                ad.setCounty(ad_str_previous);

      addressBook.getContactPanel().repopulate(c);
                                                addressBook.getFrame().validate();
                                          }
                                    });
                              }
                        }
                  });
                  county.setBounds(current_x, current_y, 200, 25);
                  county.setText(c.getAddresses().get(i).getCounty());

                  addresses_region_state_province_county.add(county);
                  pnl_addresses.add(county);
                  current_y += 30;

                  JLabel lbl_postcode = new JLabel("Postcode: ");
                  lbl_postcode.setBounds(current_x - 70, current_y, 60, 20);
                  pnl_addresses.add(lbl_postcode);

                  final JTextField postcode = new JTextField();
                  postcode.addFocusListener(new FocusListener() {
                        public String valueOnFocus;
                        public Contact c;
```

```
                        public void focusGained(FocusEvent e) {
                              c = currentContact;
                              valueOnFocus = postcode.getText();
                        }
                        public void focusLost(FocusEvent e) {
                              if (!postcode.getText().equals(valueOnFocus)) {
                                    addressBook.getActionStack().push(new Action() {
                                          public Address ad = a;
                                          public String ad_str_previous = valueOnFocus;
                                          public String ad_str = postcode.getText();
                                          public boolean secondPass = false;
                                          public void doAction() {
                                                ad.setPostcode(ad_str);
                                                if (secondPass) {
addressBook.getContactPanel().repopulate(c); }
                                                addressBook.getFrame().validate();
                                                secondPass = true;
                                          }
                                          public String getRedoText() {
                                                return "Change Postcode";
                                          }
                                          public String getUndoText() {
                                                return "Change Postcode";
                                          }
                                          public void undoAction() {
                                                ad.setPostcode(ad_str_previous);

      addressBook.getContactPanel().repopulate(c);
                                                addressBook.getFrame().validate();
                                          }
                                    });
                              }
                        }
                  });
                  postcode.setBounds(current_x, current_y, 200, 25);
                  postcode.setText(c.getAddresses().get(i).getPostcode());

                  addresses_postcode.add(postcode);
                  pnl_addresses.add(postcode);
                  current_y += 30;

                  JLabel lbl_country = new JLabel("Country: ");
                  lbl_country.setBounds(current_x - 70, current_y, 60, 20);
                  pnl_addresses.add(lbl_country);

                  final JTextField country = new JTextField();
                  country.addFocusListener(new FocusListener() {
                        public String valueOnFocus;
                        public Contact c;
                        public void focusGained(FocusEvent e) {
                              c = currentContact;
                              valueOnFocus = country.getText();
                        }
                        public void focusLost(FocusEvent e) {
                              if (!country.getText().equals(valueOnFocus)) {
                                    addressBook.getActionStack().push(new Action() {
                                          public Address ad = a;
                                          public String ad_str_previous = valueOnFocus;
                                          public String ad_str = country.getText();
                                          public boolean secondPass = false;
                                          public void doAction() {
                                                ad.setCountry(ad_str);
                                                if (secondPass) {
addressBook.getContactPanel().repopulate(c); }
                                                addressBook.getFrame().validate();
                                                secondPass = true;
                                          }
                                          public String getRedoText() {
```

```
                                          return "Change Country";
                                    }
                                    public String getUndoText() {
                                          return "Change Country";
                                    }
                                    public void undoAction() {
                                          ad.setPostcode(ad_str_previous);

      addressBook.getContactPanel().repopulate(c);
                                          addressBook.getFrame().validate();
                                    }
                              });
                        }
                  }
            });
            country.setBounds(current_x, current_y, 200, 25);
            country.setText(c.getAddresses().get(i).getCountry());

            addresses_country.add(country);
            pnl_addresses.add(country);
            current_y += 30;

            if (i != c.getAddresses().size()-1) {
                  current_y += 30;
            }




      }
      btn_add_address = new JButton("Add Address");
      btn_add_address.setBounds(current_x, current_y, 150, 25);
      btn_add_address.addActionListener(new ActionListener() { public void
actionPerformed(ActionEvent e) {
            createAndShowAddAddressDialog(addressBook.getFrame());
      }});
      pnl_addresses.add(btn_add_address);
      current_y += 40;
      pnl_addresses.setPreferredSize(new Dimension(w, current_y));
      pnl_addresses.validate();




      // fill telephone numbers! :)
      JLabel lbl_telephoneNumbers = new JLabel("Telephone Numbers: ");
      lbl_telephoneNumbers.setBounds(0, 0, 120, 20);
      pnl_telephoneNumbers.add(lbl_telephoneNumbers);
      current_x = 150;
      current_y = 0;
      for (int i = 0; i < c.getTelephoneNumbers().size(); i++) {
            final TelephoneNumber tn = c.getTelephoneNumbers().get(i);

            final JTextField field = new JTextField();
            field.setBounds(current_x, current_y, 200, 25);
            field.setText(c.getTelephoneNumbers().get(i).getNumber());
            field.addFocusListener(new FocusListener() {
                  public String valueOnFocus;
                  public Contact c;
                  public void focusGained(FocusEvent e) {
                        valueOnFocus = field.getText();
                        c = currentContact;
                  }
                  public void focusLost(final FocusEvent e) {
                        if (!field.getText().equals(valueOnFocus)) {
                              addressBook.getActionStack().push(new Action() {
```

```
                                        public TelephoneNumber t = tn;
                                        public String t_str_previous = valueOnFocus;
                                        public String t_str = field.getText();
                                        public boolean secondPass = false;
                                        public void doAction() {
                                              t.setNumber(t_str);
                                              if (secondPass) {
addressBook.getContactPanel().repopulate(c); }
                                              addressBook.getFrame().validate();
                                              secondPass = true;
                                        }
                                        public String getRedoText() {
                                              return "Change " + c.getName() + "'s
Telephone Number To " + t_str;
                                        }
                                        public String getUndoText() {
                                              return "Change " + c.getName() + "'s
Telephone Number Back To " + t_str_previous;
                                        }
                                        public void undoAction() {
                                              t.setNumber(t_str_previous);

      addressBook.getContactPanel().repopulate(c);
                                              addressBook.getFrame().validate();
                                        }
                                  });
                            }
                      }
                });

                String[] types = {"Home", "Mobile / Cell", "Work"};
                final JComboBox box = new JComboBox(types);
                box.setBounds(current_x + 205, current_y, 100, 25);
                if (tn.getType().equals(TelephoneNumber.Type.HOME)) {
                      box.setSelectedIndex(0);
                } else if (tn.getType().equals(TelephoneNumber.Type.CELL)) {
                      box.setSelectedIndex(1);
                } else if (tn.getType().equals(TelephoneNumber.Type.WORK)) {
                      box.setSelectedIndex(2);
                }
                box.addFocusListener(new FocusListener() {
                      int valueOnFocus;
                      public Contact c;
                      public void focusGained(FocusEvent e) {
                            valueOnFocus = box.getSelectedIndex();
                            c = currentContact;
                      }
                      public void focusLost(FocusEvent e) {
                            if (box.getSelectedIndex() != valueOnFocus) { // changes!!

                                  addressBook.getActionStack().push(new Action() {
                                        public TelephoneNumber t = tn;
                                        public int telnumber_type_previous =
valueOnFocus;
                                        public int telnumber_type =

box.getSelectedIndex();
                                        public boolean secondPass = false;
                                        public void doAction() {
                                              if (telnumber_type == 0) {

      t.setType(TelephoneNumber.Type.HOME);
                                              } else if (telnumber_type == 1) {

      t.setType(TelephoneNumber.Type.CELL);
                                              } else if (telnumber_type == 2) {

      t.setType(TelephoneNumber.Type.WORK);
                                              }
```

```
                                                if (secondPass) {
addressBook.getContactPanel().repopulate(c); }
                                                addressBook.getFrame().validate();
                                                secondPass = true;
                                          }
                                          public String getRedoText() {
                                                if (telnumber_type == 0) {
                                                      return "Set Telephone Number Type
As 'Home'";
                                                } else if (telnumber_type == 1) {
                                                      return "Set Telephone Number Type
As 'Mobile / Cell'";
                                                } else if (telnumber_type == 2) {
                                                      return "Set Telephone Number Type
As 'Work'";
                                                }
                                                return "BAD BAD BAD";
                                          }
                                          public String getUndoText() {
                                                if (telnumber_type_previous == 0) {
                                                      return "Set Telephone Number Type
As 'Home'";
                                                } else if (telnumber_type_previous == 1)
{
                                                      return "Set Telephone Number Type
As 'Mobile / Cell'";
                                                } else if (telnumber_type == 2) {
                                                      return "Set Telephone Number Type
As 'Work'";
                                                }
                                                return "BAD BAD BAD!";
                                          }
                                          public void undoAction() {
                                                if (telnumber_type_previous == 0) {

      t.setType(TelephoneNumber.Type.HOME);
                                                } else if (telnumber_type_previous == 1)
{

      t.setType(TelephoneNumber.Type.CELL);
                                                } else if (telnumber_type_previous == 2)
{

      t.setType(TelephoneNumber.Type.WORK);
                                                }

      addressBook.getContactPanel().repopulate(c);
                                                addressBook.getFrame().validate();
                                          }
                                    });
                              }
                        }
                  });

                  JButton delete = new JButton(new ImageIcon("files/Delete.png"));
                  delete.setToolTipText("Remove Telephone Number");
                  delete.setBounds(current_x + 205 + 105, current_y, 25, 25);
                  delete.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                              addressBook.getActionStack().push(new Action() {
                                    public Contact c = currentContact;
                                    public TelephoneNumber t = tn;
                                    public void doAction() {
                                          c.removeTelephoneNumber(t);

      addressBook.getContactPanel().repopulate(currentContact);
                                          addressBook.getFrame().validate();
                                    }
```

```
                                    public String getRedoText() {
                                          return "Remove Telephone Number from " +
c.getName();
                                    }
                                    public String getUndoText() {
                                          return "Add Telephone Number to " +
c.getName();
                                    }
                                    public void undoAction() {
                                          c.addTelephoneNumber(t);

      addressBook.getContactPanel().repopulate(currentContact);
                                          addressBook.getFrame().validate();
                                    }
                              });
                        }
                  });
                  telephoneNumbers.add(field);
                  telephoneNumbers_type.add(box);
                  telephoneNumbers_delete.add(delete);

                  pnl_telephoneNumbers.add(field);
                  pnl_telephoneNumbers.add(box);
                  pnl_telephoneNumbers.add(delete);

                  current_y += 30;
            }
            btn_add_telephoneNumber = new JButton("Add Telephone Number");
            btn_add_telephoneNumber.setBounds(current_x, current_y, 150, 25);
            btn_add_telephoneNumber.addActionListener(new ActionListener() { public void
actionPerformed(ActionEvent e) {
                  createAndShowAddTelephoneNumberDialog(addressBook.getFrame());
            }});
            pnl_telephoneNumbers.add(btn_add_telephoneNumber);
            current_y += 40;
            pnl_telephoneNumbers.setPreferredSize(new Dimension(w, current_y));
            pnl_telephoneNumbers.validate();




            // email addressz? wut?
            JLabel lbl_emailAddresses = new JLabel("Email Addresses: ");
            lbl_emailAddresses.setBounds(0, 0, 120, 20);
            pnl_emailAddresses.add(lbl_emailAddresses);
            current_x = 150;
            current_y = 0;
            for (int i = 0; i < c.getEmailAddresses().size(); i++) {
                  final EmailAddress ea = c.getEmailAddresses().get(i);
                  final JTextField field = new JTextField();
                  field.setBounds(current_x, current_y, 200, 25);
                  field.setText(c.getEmailAddresses().get(i).getAddress());
                  field.addFocusListener(new FocusListener() {
                        public String valueOnFocus;
                        public Contact c;
                        public void focusGained(FocusEvent e) {
                              valueOnFocus = field.getText();
                              c = currentContact;
                        }
                        public void focusLost(final FocusEvent event) {
                              if (!field.getText().equals(valueOnFocus)) {
                                    addressBook.getActionStack().push(new Action() {
                                          public EmailAddress e = ea;
                                          public String email_str_previous =
valueOnFocus;
                                          public String email_str = field.getText();
                                          public boolean secondPass = false;
```

```
                                        public void doAction() {
                                              e.setAddress(email_str);
                                              if (secondPass) {
addressBook.getContactPanel().repopulate(c); }
                                              addressBook.getFrame().validate();
                                              secondPass = true;
                                        }
                                        public String getRedoText() {
                                              return "Change Email Address To " +
email_str;
                                        }
                                        public String getUndoText() {
                                              return "Change Email Address Back To " +
email_str_previous;
                                        }
                                        public void undoAction() {
                                              e.setAddress(email_str_previous);

      addressBook.getContactPanel().repopulate(c);
                                              addressBook.getFrame().validate();
                                        }
                                  });
                            }
                      }
                });


                String[] types = {"Personal", "Corporate"};
                final JComboBox box = new JComboBox(types);
                box.setBounds(current_x + 205, current_y, 80, 25);
                if
(c.getEmailAddresses().get(i).getType().equals(EmailAddress.Type.HOME)) {
                      box.setSelectedIndex(0);
                } else if
(c.getEmailAddresses().get(i).getType().equals(EmailAddress.Type.WORK)) {
                      box.setSelectedIndex(1);
                }
                box.addFocusListener(new FocusListener() {
                      int valueOnFocus;
                      public Contact c;
                      public void focusGained(FocusEvent e) {
                            valueOnFocus = box.getSelectedIndex();
                            c = currentContact;
                      }
                      public void focusLost(FocusEvent event) {
                            if (box.getSelectedIndex() != valueOnFocus) { // changes!!
                                  addressBook.getActionStack().push(new Action() {
                                        public EmailAddress e = ea;
                                        public int email_type_previous = valueOnFocus;
                                        public int email_type =
box.getSelectedIndex();
                                        public boolean secondPass = false;
                                        public void doAction() {
                                              if (email_type == 0) {
                                                    e.setType(EmailAddress.Type.HOME);
                                              } else if (email_type == 1) {
                                                    e.setType(EmailAddress.Type.WORK);
                                              }
                                              if (secondPass) {
addressBook.getContactPanel().repopulate(c); }
                                              addressBook.getFrame().validate();
                                              secondPass = true;
                                        }
                                        public String getRedoText() {
                                              if (email_type == 0) {
                                                    return "Set Email Type As
'Personal'";
                                              } else if (email_type == 1) {
```

```
                                                return "Set Email Type As
'Corporate'";
                                          }
                                          return "BAD BAD BAD";
                                    }
                                    public String getUndoText() {
                                          if (email_type_previous == 0) {
                                                return "Set Email Type As
'Personal'";
                                          } else if (email_type_previous == 1) {
                                                return "Set Email Type As
'Corporate'";
                                          }
                                          return "BAD BAD BAD!";
                                    }
                                    public void undoAction() {
                                          if (email_type_previous == 0) {
                                                e.setType(EmailAddress.Type.HOME);
                                          } else if (email_type_previous == 1) {
                                                e.setType(EmailAddress.Type.WORK);
                                          }

      addressBook.getContactPanel().repopulate(c);
                                          addressBook.getFrame().validate();
                                    }
                              });
                        }
                  }
            });

            final JButton mailto = new JButton(new ImageIcon("files/Mail.png"));
            mailto.setToolTipText("Send An Email To " + ea.getAddress());
            mailto.setBounds(current_x + 205 + 85, current_y, 25, 25);
            mailto.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent e) {
                        try {
                              Desktop desktop = Desktop.getDesktop();
                              desktop.mail(new URI("mailto:" + ea.getAddress()));
                        } catch (Exception ex) {

      JOptionPane.showMessageDialog(addressBook.getFrame(), "Unable to launch default
mail application.", "Error:", JOptionPane.ERROR_MESSAGE);
                        }
                  }
            });

            final JButton delete = new JButton(new ImageIcon("files/Delete.png"));
            delete.setToolTipText("Remove Email Address");
            delete.setBounds(current_x + 205 + 85 + 30, current_y, 25, 25);
            delete.addActionListener(new ActionListener() {
                  public void actionPerformed(ActionEvent e) {
                        addressBook.getActionStack().push(new Action() {
                              public Contact c = currentContact;
                              public EmailAddress e = ea;
                              public void doAction() {
                                    c.removeEmailAddress(e);

      addressBook.getContactPanel().repopulate(currentContact);
                                    addressBook.getFrame().validate();
                              }
                              public String getRedoText() {
                                    return "Remove Email Address from " +
c.getName();
                              }
                              public String getUndoText() {
                                    return "Add Email Address to " + c.getName();
                              }
                              public void undoAction() {
```

```
                                        c.addEmailAddress(e);

      addressBook.getContactPanel().repopulate(currentContact);
                                        addressBook.getFrame().validate();
                                  }
                            });
                      }
                });

                emailAddresses.add(field);
                emailAddresses_type.add(box);
                emailAddresses_mailto.add(mailto);
                emailAddresses_delete.add(delete);

                pnl_emailAddresses.add(field);
                pnl_emailAddresses.add(box);
                pnl_emailAddresses.add(mailto);
                pnl_emailAddresses.add(delete);

                current_y += 30;
          }
          btn_add_emailAddress = new JButton("Add Email Address");
          btn_add_emailAddress.setBounds(current_x, current_y, 150, 25);
          btn_add_emailAddress.addActionListener(new ActionListener() { public void
actionPerformed(ActionEvent e) {
                createAndShowAddEmailAddressDialog(addressBook.getFrame());
          }});
          pnl_emailAddresses.add(btn_add_emailAddress);
          current_y += 40;
          pnl_emailAddresses.setPreferredSize(new Dimension(w, current_y));
          pnl_emailAddresses.validate();

          pnl_buttons.setVisible(true);
    }

    /**
     * Add a POBox number to a contact's address.
     * @param parent The Parent Frame.
     * @param ad The Address to add the POBox number to.
     */
    protected void createAndShowSetPOBoxNumberDialog(JFrame parent, final Address ad)
{
          final JDialog frame = new JDialog(parent, "Add PO BOX:", true, null);
          frame.setSize(330, 100);
          frame.setLayout(null);
          frame.setResizable(false);
          frame.setIconImage(new ImageIcon("files/Home.png").getImage());
          frame.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
          frame.addWindowListener(new WindowListener() {
                public void windowActivated(WindowEvent e) {}
                public void windowClosed(WindowEvent e) {}
                public void windowClosing(WindowEvent e) {
                      addressBook.getFrame().setEnabled(true);
                      frame.dispose();
                }
                public void windowDeactivated(WindowEvent e) {}
                public void windowDeiconified(WindowEvent e) {}
                public void windowIconified(WindowEvent e) {}
                public void windowOpened(WindowEvent e) {}
          });
          frame.setLocationRelativeTo(parent);

          JPanel panel = new JPanel();
          panel.setLayout(null);
          panel.setBounds(10, 10, 300, 100);

          JLabel label = new JLabel("PO Box: ");
          label.setBounds(0, 0, 100, 20);
```

```
		panel.add(label);

		final JTextField textfield = new JTextField("");
		textfield.setBounds(100, 0, 200, 20);
		panel.add(textfield);

		JButton btn_save = new JButton("Save");
		btn_save.setBounds(0, 30, 300, 25);
		btn_save.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				addressBook.getActionStack().push(new Action() {
					Contact c = currentContact;
					Address a = ad;
					String POBOX = textfield.getText();
					public void doAction() {
						a.setPOBoxNumber(POBOX);
						addressBook.getContactPanel().repopulate(c);
						addressBook.getFrame().validate();
					}
					public String getRedoText() {
						return "Add PO Box # (" + c.getName() + ")";
					}
					public String getUndoText() {
						return "Remove PO Box # (" + c.getName() + ")";
					}
					public void undoAction() {
						a.setPOBoxNumber("");
						addressBook.getContactPanel().repopulate(c);
						addressBook.getFrame().validate();
					}
				});
				addressBook.getFrame().setEnabled(true);
				frame.dispose();
			}
		});
		panel.add(btn_save);

		frame.add(panel);
		frame.setVisible(true);
	}

	/**
	 * Add an Extended Address to a contact's address.
	 * @param parent The Parent Frame.
	 * @param ad The Address to add the Extended Address to.
	 */
	protected void createAndShowSetExtendedAddressDialog(JFrame parent, final Address
ad) {
		final JDialog frame = new JDialog(parent, "Add Business:", true, null);
		frame.setSize(330, 100);
		frame.setLayout(null);
		frame.setResizable(false);
		frame.setIconImage(new ImageIcon("files/Home.png").getImage());
		frame.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
		frame.addWindowListener(new WindowListener() {
			public void windowActivated(WindowEvent e) {}
			public void windowClosed(WindowEvent e) {}
			public void windowClosing(WindowEvent e) {
				addressBook.getFrame().setEnabled(true);
				frame.dispose();
			}
			public void windowDeactivated(WindowEvent e) {}
			public void windowDeiconified(WindowEvent e) {}
			public void windowIconified(WindowEvent e) {}
			public void windowOpened(WindowEvent e) {}
		});
		frame.setLocationRelativeTo(parent);
```

```
		JPanel panel = new JPanel();
		panel.setLayout(null);
		panel.setBounds(10, 10, 300, 100);

		JLabel label = new JLabel("Business Name: ");
		label.setBounds(0, 0, 100, 20);
		panel.add(label);

		final JTextField textfield = new JTextField("");
		textfield.setBounds(100, 0, 200, 20);
		panel.add(textfield);

		JButton btn_save = new JButton("Save");
		btn_save.setBounds(0, 30, 300, 25);
		btn_save.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				addressBook.getActionStack().push(new Action() {
					Contact c = currentContact;
					Address a = ad;
					String extendedaddress = textfield.getText();
					public void doAction() {
						a.setExtendedAddress(extendedaddress);
						addressBook.getContactPanel().repopulate(c);
						addressBook.getFrame().validate();
					}
					public String getRedoText() {
						return "Add Business Name (" + c.getName() + ")";
					}
					public String getUndoText() {
						return "Remove Business Name (" + c.getName() + ")";
					}
					public void undoAction() {
						a.setExtendedAddress("");
						addressBook.getContactPanel().repopulate(c);
						addressBook.getFrame().validate();
					}
				});
				addressBook.getFrame().setEnabled(true);
				frame.dispose();
			}
		});
		panel.add(btn_save);

		frame.add(panel);
		frame.setVisible(true);
	}

	/**
	 * Add an address to the currently selected. contact.
	 * @param parent The parent window/frame.
	 */
	protected void createAndShowAddAddressDialog(JFrame parent) {
		final JDialog add_address_dialog = new JDialog(parent, "Add Address: ",
true, null);
		add_address_dialog.setIconImage(new ImageIcon("files/Home.png").getImage());
		add_address_dialog.setSize(330, 330);
		add_address_dialog.setLayout(null);
		add_address_dialog.setResizable(false);
		add_address_dialog.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);


		JPanel panel_add_address = new JPanel();
		panel_add_address.setLayout(null);
		panel_add_address.setBounds(10, 10, 300, 295);

			JLabel lbl_add_address = new JLabel("Street: ");
			lbl_add_address.setBounds(0, 0, 100, 20);
			panel_add_address.add(lbl_add_address);
```

```
final JTextArea ta_street = new JTextArea();
ta_street.setText("");
ta_street.setFont(new Font("Arial", Font.PLAIN, 12));
ta_street.setLineWrap(true);
ta_street.setWrapStyleWord(true);
JScrollPane sp_add_address = new JScrollPane(ta_street);
sp_add_address.setBounds(100, 0, 200, 100);
panel_add_address.add(sp_add_address);


JLabel lbl_city = new JLabel("City: ");
lbl_city.setBounds(0, 105, 100, 20);
panel_add_address.add(lbl_city);
final JTextField ta_city = new JTextField();
ta_city.setBounds(100, 105, 200, 25);
panel_add_address.add(ta_city);


JLabel lbl_county = new JLabel("County: ");
lbl_county.setBounds(0, 135, 100, 20);
panel_add_address.add(lbl_county);
final JTextField ta_county = new JTextField();
ta_county.setBounds(100, 135, 200, 25);
panel_add_address.add(ta_county);

JLabel lbl_postcode = new JLabel("Postcode: ");
lbl_postcode.setBounds(0, 165, 100, 20);
panel_add_address.add(lbl_postcode);
final JTextField ta_postcode = new JTextField();
ta_postcode.setBounds(100, 165, 200, 25);
panel_add_address.add(ta_postcode);

JLabel lbl_country = new JLabel("Country: ");
lbl_country.setBounds(0, 195, 100, 20);
panel_add_address.add(lbl_country);
final JTextField ta_country = new JTextField();
ta_country.setBounds(100, 195, 200, 25);
panel_add_address.add(ta_country);

String[] types = { "Home", "Work"};
JLabel lbl_add_addresstype = new JLabel("Type: ");
lbl_add_addresstype.setBounds(0, 225, 100, 25);
panel_add_address.add(lbl_add_addresstype);
final JComboBox box_addresstype = new JComboBox(types);
box_addresstype.setSelectedIndex(0);
box_addresstype.setBounds(100, 225, 200, 25);
panel_add_address.add(box_addresstype);

JButton btn_add_address = new JButton("Save");
btn_add_address.setBounds(0, 260, 300, 25);
btn_add_address.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
            addressBook.getActionStack().push(new Action() {
                  Contact c = currentContact;
                  Address a = new Address();
                  String street = ta_street.getText();
                  String city = ta_city.getText();
                  String county = ta_county.getText();
                  String postcode = ta_postcode.getText();
                  String country = ta_country.getText();
                  int index = box_addresstype.getSelectedIndex();

                  public void doAction() {
                        if (index == 0) {
                              a.setType(Address.Type.HOME);
                        } else if (index == 1) {
                              a.setType(Address.Type.WORK);
```

```
                                        }

                                        a.setStreetAddress(street);
                                        a.setCity(city);
                                        a.setCounty(county);
                                        a.setPostcode(postcode);
                                        a.setCountry(country);
                                        c.addAddress(a);
                                        addressBook.getContactPanel().repopulate(c);
                                        addressBook.getFrame().validate();
                                        addressBook.getFrame().setEnabled(true);
                                  }
                                  public String getRedoText() {
                                        return "Add Address to " + c.getName();
                                  }
                                  public String getUndoText() {
                                        return "Remove Address from " + c.getName();
                                  }
                                  public void undoAction() {
                                        c.removeAddress(a);
                                        addressBook.getContactPanel().repopulate(c);
                                        addressBook.getFrame().validate();
                                        addressBook.getFrame().setEnabled(true);
                                  }
                            });
                            add_address_dialog.dispose();
                      }
                });
                panel_add_address.add(btn_add_address);

          add_address_dialog.add(panel_add_address);
          add_address_dialog.setLocationRelativeTo(parent);
          add_address_dialog.setVisible(true);
    }

    /**
     * Add a TelephoneNumber to the currently selected contact.
     * @param parent The parent window/frame.
     */
    protected void createAndShowAddTelephoneNumberDialog(JFrame parent) {
          final JDialog add_telephonenumber_dialog = new JDialog(parent, "Add
Telephone Number: ", true, null);
          add_telephonenumber_dialog.setIconImage(new
ImageIcon("files/Phone.png").getImage());
          add_telephonenumber_dialog.setSize(330, 130);
          add_telephonenumber_dialog.setLayout(null);
          add_telephonenumber_dialog.setResizable(false);

    add_telephonenumber_dialog.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
          add_telephonenumber_dialog.addWindowListener(new WindowListener() {
                public void windowActivated(WindowEvent e) {}
                public void windowClosed(WindowEvent e) {}
                public void windowClosing(WindowEvent e) {
                      addressBook.getFrame().setEnabled(true);
                      add_telephonenumber_dialog.dispose();
                }
                public void windowDeactivated(WindowEvent e) {}
                public void windowDeiconified(WindowEvent e) {}
                public void windowIconified(WindowEvent e) {}
                public void windowOpened(WindowEvent e) {}
          });

          JPanel panel_add_telephonenumber = new JPanel();
          panel_add_telephonenumber.setLayout(null);
          panel_add_telephonenumber.setBounds(10, 10, 300, 100);

                JLabel lbl_add_telephonenumner = new JLabel("Telephone Number: ");
                lbl_add_telephonenumner.setBounds(0, 0, 100, 20);
```

```
                panel_add_telephonenumber.add(lbl_add_telephonenumner);
                final JTextField tf_add_telephonenumber = new JTextField("");
                tf_add_telephonenumber.setBounds(100, 0, 200, 20);
                panel_add_telephonenumber.add(tf_add_telephonenumber);

                String[] types = { "Home", "Mobile / Cell", "Work"};
                JLabel lbl_add_telephonenumbertype = new JLabel("Type: ");
                lbl_add_telephonenumbertype.setBounds(0, 30, 100, 20);
                panel_add_telephonenumber.add(lbl_add_telephonenumbertype);
                final JComboBox box_telephonenumbertype = new JComboBox(types);
                box_telephonenumbertype.setSelectedIndex(0);
                box_telephonenumbertype.setBounds(100, 30, 200, 20);
                panel_add_telephonenumber.add(box_telephonenumbertype);

                JButton btn_add_telephonenumber = new JButton("Save");
                btn_add_telephonenumber.setBounds(0, 60, 300, 25);
                btn_add_telephonenumber.addActionListener(new ActionListener() {
                      public void actionPerformed(ActionEvent e) {
                            addressBook.getActionStack().push(new Action() {
                                  Contact c = currentContact;
                                  TelephoneNumber tn = new TelephoneNumber();
                                  String number = tf_add_telephonenumber.getText();
                                  int index =
box_telephonenumbertype.getSelectedIndex();
                                  public void doAction() {
                                        tn.setNumber(number);
                                        if (index == 0) {
                                              tn.setType(TelephoneNumber.Type.HOME);
                                        } else if (index == 1) {
                                              tn.setType(TelephoneNumber.Type.CELL);
                                        } else if (index == 2) {
                                              tn.setType(TelephoneNumber.Type.WORK);
                                        }
                                        c.addTelephoneNumber(tn);

      addressBook.getContactPanel().repopulate(currentContact);
                                        addressBook.getFrame().validate();
                                        addressBook.getFrame().setEnabled(true);
                                  }
                                  public String getRedoText() {
                                        return "Add Telephone Number to " +
c.getName();
                                  }
                                  public String getUndoText() {
                                        return "Remove Telephone Number from " +
c.getName();
                                  }
                                  public void undoAction() {
                                        c.removeTelephoneNumber(tn);

      addressBook.getContactPanel().repopulate(currentContact);
                                        addressBook.getFrame().validate();
                                        addressBook.getFrame().setEnabled(true);
                                  }
                            });
                            add_telephonenumber_dialog.dispose();
                      }
                });
                panel_add_telephonenumber.add(btn_add_telephonenumber);

                add_telephonenumber_dialog.add(panel_add_telephonenumber);
                add_telephonenumber_dialog.setLocationRelativeTo(parent);
                addressBook.getFrame().setEnabled(false);
                add_telephonenumber_dialog.setVisible(true);
      }

      /**
       * Add an EmailAddress to the currently selected contact.
```

```
	 * @param parent The parent window/frame.
	 */
	protected void createAndShowAddEmailAddressDialog(JFrame parent) {
		final JDialog add_email_dialog = new JDialog(parent, "Add Email Address: ",
true, null);
		add_email_dialog.setIconImage(new ImageIcon("files/Mail.png").getImage());
		add_email_dialog.setSize(330, 130);
		add_email_dialog.setLayout(null);
		add_email_dialog.setResizable(false);
		add_email_dialog.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
		add_email_dialog.addWindowListener(new WindowListener() {
			public void windowActivated(WindowEvent e) {}
			public void windowClosed(WindowEvent e) {}
			public void windowClosing(WindowEvent e) {
				addressBook.getFrame().setEnabled(true);
				add_email_dialog.dispose();
			}
			public void windowDeactivated(WindowEvent e) {}
			public void windowDeiconified(WindowEvent e) {}
			public void windowIconified(WindowEvent e) {}
			public void windowOpened(WindowEvent e) {}
		});

		JPanel panel_add_email = new JPanel();
		panel_add_email.setLayout(null);
		panel_add_email.setBounds(10, 10, 300, 100);

			JLabel lbl_add_email = new JLabel("Email Address: ");
			lbl_add_email.setBounds(0, 0, 100, 20);
			panel_add_email.add(lbl_add_email);
			final JTextField tf_add_email = new JTextField("");
			tf_add_email.setBounds(100, 0, 200, 20);
			panel_add_email.add(tf_add_email);

			String[] types = { "Personal", "Corporate"};
			JLabel lbl_add_emailtype = new JLabel("Type: ");
			lbl_add_emailtype.setBounds(0, 30, 100, 20);
			panel_add_email.add(lbl_add_emailtype);
			final JComboBox box_emailtype = new JComboBox(types);
			box_emailtype.setSelectedIndex(0);
			box_emailtype.setBounds(100, 30, 200, 20);
			panel_add_email.add(box_emailtype);

			JButton btn_add_email = new JButton("Save");
			btn_add_email.setBounds(0, 60, 300, 25);
			btn_add_email.addActionListener(new ActionListener() {
				public void actionPerformed(ActionEvent e) {
					addressBook.getActionStack().push(new Action() {
						Contact c = currentContact;
						EmailAddress ea = new EmailAddress();
						String addr = tf_add_email.getText();
						int index = box_emailtype.getSelectedIndex();
						public void doAction() {
							ea.setAddress(addr);
							if (index == 0) {
								ea.setType(EmailAddress.Type.HOME);
							} else if (box_emailtype.getSelectedIndex() ==
1) {
								ea.setType(EmailAddress.Type.WORK);
							}
							c.addEmailAddress(ea);

	addressBook.getContactPanel().repopulate(currentContact);
							addressBook.getFrame().validate();
							addressBook.getFrame().setEnabled(true);
						}
						public String getRedoText() {
							return "Add Email Address to " + c.getName();
```

```
                                    }
                                    public String getUndoText() {
                                          return "Remove Email Address from " +
c.getName();
                                    }
                                    public void undoAction() {
                                          c.removeEmailAddress(ea);

      addressBook.getContactPanel().repopulate(currentContact);
                                          addressBook.getFrame().validate();
                                          addressBook.getFrame().setEnabled(true);
                                    }
                              });
                              add_email_dialog.dispose();
                        }
                  });
                  panel_add_email.add(btn_add_email);

            add_email_dialog.add(panel_add_email);
            add_email_dialog.setLocationRelativeTo(parent);
            addressBook.getFrame().setEnabled(false);
            add_email_dialog.setVisible(true);
      }

      /**
       * Edit the name of the currently selected contact.
       * @param parent The parent window/frame.
       */
      protected void createAndShowEditNameDialog(JFrame parent) {
            JDialog e_name_d = new JDialog(parent, "Edit Contact's Name: ", true, null);
            e_name_d.setTitle("Edit Contact's Name: ");
            e_name_d.setIconImage(new ImageIcon("files/User.png").getImage());
            e_name_d.setSize(330, 130);
            e_name_d.setLayout(null);
            e_name_d.setResizable(false);

            JPanel panel_edit_name = new JPanel();
            panel_edit_name.setLayout(null);
            panel_edit_name.setBounds(10, 10, 300, 100);
                  JLabel lbl_edit_forenames = new JLabel("Forenames: ");
                  lbl_edit_forenames.setBounds(0, 0, 100, 20);
                  panel_edit_name.add(lbl_edit_forenames);
                  final JTextField tf_edit_forenames = new
JTextField(currentContact.getForenames());
                  tf_edit_forenames.setBounds(100, 0, 200, 20);
                  panel_edit_name.add(tf_edit_forenames);

                  JLabel lbl_edit_surname = new JLabel("Surname: ");
                  lbl_edit_surname.setBounds(0, 30, 100, 20);
                  panel_edit_name.add(lbl_edit_surname);
                  final JTextField tf_edit_surname = new
JTextField(currentContact.getSurname());
                  tf_edit_surname.setBounds(100, 30, 200, 20);
                  panel_edit_name.add(tf_edit_surname);

                  JButton btn_edit_name_save = new JButton("Save");
                  btn_edit_name_save.setBounds(0, 60, 300, 25);
                  btn_edit_name_save.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent e) {
                              String forenames = tf_edit_forenames.getText();
                              String surname = tf_edit_surname.getText();

                              currentContact.setForenames(forenames);
                              currentContact.setSurname(surname);

                              lbl_fullname.setText("<html><big>" + forenames + " " +
surname + "</big></html>");
```

```
                                addressBook.refreshList();
                            }
                        });
                        panel_edit_name.add(btn_edit_name_save);

                e_name_d.add(panel_edit_name);
                e_name_d.setLocationRelativeTo(parent);
                e_name_d.setVisible(true);
        }
}

package org.ag.addressbook;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Scanner;

import javax.swing.JOptionPane;

import org.ag.addressbook.property.Address;
import org.ag.addressbook.property.EmailAddress;
import org.ag.addressbook.property.TelephoneNumber;
import org.ag.util.StringUtil;

/**
 * Importer
 * This is used to import contacts from a file into the existing address book instance.
 * Note the difference between Open and Import. They both use this class to add
contacts from a file.
 * @author Ashley Gwinnell
 */
public class Importer
{
        private ArrayList<File> fs = new ArrayList<File>();

        /**
         * Create a new Importer from a string reference.
         * @param f The Absolute Filename in a string.
         */
        public Importer(String f) {
                this(new File(f));
        }

        /**
         * Create a new Importer from a file.
         * @param f The file to import.
         */
        public Importer(File f)
        {
                this.fs.add(f);
        }

        /**
         * Create a new Importer with multiple files.
         * Used in Import and not in Open.
         * @param f An array of files to open/import/get contacts from.
         */
        public Importer(File[] f)
        {
                for (int i = 0; i < f.length; i++) {
                        this.fs.add(f[i]);
                }
        }

        /**
```

```
	 * Reads all of the files asked and returns the contacts from all combined.
	 * @return the contacts from all files combined.
	 */
	public ArrayList<Contact> load()
	{
		ArrayList<Contact> list = new ArrayList<Contact>();
		try
		{
			for (int i = 0; i < this.fs.size(); i++) {
				String ext = Importer.getFileExtension(this.fs.get(i));
				if (ext.equals(Importer.BUAB)) {
					list.addAll(this.importBUAB(i));
				} else if (ext.equals(Importer.VCARD_1) ||
ext.equals(Importer.VCARD_2)) {
					Structure s = Importer.getVCardVersion(this.fs.get(i));
					if (s == null) {
						return list;
					} else if (s.equals(Structure.VCARD30)) {
						list.addAll(this.importVCard30(i));
					}
				}
			}
		}
		catch (Exception e)
		{
			e.printStackTrace();
		}
		return list;
	}


	/**
	 * Cleans a string on importing.
	 * @param s the file structure.
	 * @param str the "dirty" string to clean for that file structure.
	 * @return a "clean" string.
	 */
	private String clean(Structure s, String str) {
		if (s.equals(Structure.VCARD30)) {
			return str.replace("\\,", ",").replace("\\n", "\n");
		}
		return "";
	}

	/**
	 * Import contacts from a VCard 3.0 file.
	 * @param x The array index from a list of files in the importer.
	 * @return contacts from a VCard 3.0 file.
	 */
	private ArrayList<Contact> importVCard30(int x) {
		ArrayList<Contact> contacts = new ArrayList<Contact>();

		try {
			BufferedReader reader = new BufferedReader(new
FileReader(this.fs.get(x)));
			String file = "";
			String line = "";
			while ((line = reader.readLine()) != null) {
				file += line + "\n";
			}

			Contact c = null;
			Scanner s = new Scanner(file);
			while (s.hasNextLine()) {
				line = s.nextLine();
				try {
					if (line.equals("BEGIN:VCARD"))
					{
```

```
                                    c = new Contact();
                              }
                              else if (line.substring(0, 3).equals("FN:"))
                              {
                                    c.setName(line.substring(3));
                              }
                              else if (line.substring(0, 6).equals("EMAIL;"))
                              {
                                    EmailAddress ea = new EmailAddress();
                                    if (line.toUpperCase().contains("TYPE=HOME")) {
                                          ea.setType(EmailAddress.Type.HOME);
                                    } else if (line.toUpperCase().contains("TYPE=WORK"))
{
                                          ea.setType(EmailAddress.Type.WORK);
                                    }
                                    if (line.toUpperCase().contains("TYPE=pref")) {
                                          ea.setPreferred(true);
                                    } else {
                                          ea.setPreferred(false);
                                    }
                                    ea.setAddress(line.substring(line.lastIndexOf(":")
+1));
                                    c.addEmailAddress(ea);
                              }
                              else if (line.substring(0,
4).toUpperCase().equals("TEL;"))
                              {
                                    TelephoneNumber tel = new TelephoneNumber();
                                    if (line.toUpperCase().contains("TYPE=HOME")) {
                                          tel.setType(TelephoneNumber.Type.HOME);
                                    } else if (line.toUpperCase().contains("TYPE=CELL"))
{
                                          tel.setType(TelephoneNumber.Type.CELL);
                                    } else if (line.toUpperCase().contains("TYPE=WORK"))
{
                                          tel.setType(TelephoneNumber.Type.WORK);
                                    }
                                    if (line.toUpperCase().contains("TYPE=PREF")) {
                                          tel.setPreferred(true);
                                    } else {
                                          tel.setPreferred(false);
                                    }
                                    tel.setNumber(line.substring(line.lastIndexOf(":")
+1));
                                    c.addTelephoneNumber(tel);
                              }
                              else if (line.substring(0,
4).toUpperCase().equals("ADR;"))
                              {
                                    Address a = new Address();
                                    if (line.toUpperCase().contains("TYPE=HOME")) {
                                          a.setType(Address.Type.HOME);
                                    } else if (line.toUpperCase().contains("TYPE=WORK"))
{
                                          a.setType(Address.Type.WORK);
                                    }
                                    if (line.toUpperCase().contains("TYPE=PREF")) {
                                          a.setPreferred(true);
                                    } else {
                                          a.setPreferred(false);
                                    }
                                    String addressLine =
line.substring(line.lastIndexOf(":")+1);
                                    String[] parts =
StringUtil.splitWithoutTrimming(addressLine, ';');
                                    a.setPOBoxNumber(this.clean(Structure.VCARD30,
parts[0]));
                                    a.setExtendedAddress(this.clean(Structure.VCARD30,
```

```
parts[1]));
                                    a.setStreetAddress(this.clean(Structure.VCARD30,
parts[2]));
                                    a.setCity(this.clean(Structure.VCARD30, parts[3]));
                                    a.setCounty(this.clean(Structure.VCARD30,
parts[4]));
                                    a.setPostcode(this.clean(Structure.VCARD30,
parts[5]));
                                    a.setCountry(this.clean(Structure.VCARD30,
parts[6]));
                                    c.addAddress(a);
                              }
                              else if (line.equals("END:VCARD"))
                              {
                                    contacts.add(c);
                                    c = null;
                              }
                        } catch (Exception e) {
                              //e.printStackTrace();
                              JOptionPane.showMessageDialog(null, "Could not load part
of the file.", "Error:", JOptionPane.ERROR_MESSAGE);
                              return contacts;
                        }
                  }

            } catch (IOException e) {
                  return contacts;
            }
            return contacts;
      }

      /**
       * Import contacts from a BUAB file.
       * @param x the array index of the file in the list in the importer.
       * @return a list of contacts from the BUAB file.
       * @throws FileNotFoundException
       */
      private ArrayList<Contact> importBUAB(int x) throws FileNotFoundException
      {
            ArrayList<Contact> contacts = new ArrayList<Contact>();
            Scanner scan = new Scanner(fs.get(x));
            String line;
            int i = 0;
            int lines_per_contact = 4;

            String forenames = "";
            String surname = "";
            TelephoneNumber homephone = new TelephoneNumber();
            TelephoneNumber mobilephone = new TelephoneNumber();

            while (scan.hasNextLine()) {
                  line = scan.nextLine();
                  if (i == lines_per_contact) {
                        i = 0;
                  }

                  switch (i) {
                        case 0:
                              String[] names = line.split(" ");
                              for (int j = 0; j < names.length-1; j++) {
                                    forenames += names[j];
                              }
                              surname = names[names.length-1];
                              break;
                        case 1:
                              homephone.setNumber(line);
                              homephone.setType(TelephoneNumber.Type.HOME);
                              homephone.setPreferred(true);
```

```
                        break;
                    case 2:
                        mobilephone.setNumber(line);
                        mobilephone.setType(TelephoneNumber.Type.CELL);
                        mobilephone.setPreferred(false);
                        break;
                    case 3:
                        Contact c = new Contact();
                        c.setForenames(forenames);
                        c.setSurname(surname);
                        c.addTelephoneNumber(homephone);
                        c.addTelephoneNumber(mobilephone);
                            Address a = new Address();
                            a.setStreetAddress(line.replace(", ",
"\n").replace(",", "\n"));
                            a.setType(Address.Type.HOME);
                            a.setPreferred(true);
                            c.addAddress(a);
                        contacts.add(c);
                        forenames = "";
                        surname = "";
                        homephone = new TelephoneNumber();
                        mobilephone = new TelephoneNumber();
                        break;
                }


                i++;
            }

            return contacts;
        }


        /** File extension for buab **/
        public static final String BUAB = new String("buab");
        /** File extension for vcard **/
        public static final String VCARD_1 = new String("vcf");
        /** Another file extension for vcard **/
        public static final String VCARD_2 = new String("vcard");

        /**
         * Get the file extension from a file object.
         * @param f the file object to get the extension for.
         * @return the file extension.
         */
        public static String getFileExtension(File f) {
          String ext = "";
          String s = f.getName();
          int i = s.lastIndexOf('.');

          if (i > 0 &&  i < s.length() - 1) {
              ext = s.substring(i+1).toLowerCase();
          }
          return ext;
      }

        /**
         * Checks whether the file has a valid extension.
         * @param f the file to check the extension on.
         * @return true on valid extension.
         */
        public static boolean isValidExtension(File f) {
            String s = Importer.getFileExtension(f);
            if (s.equals(Importer.BUAB) || s.equals(Importer.VCARD_1) ||
s.equals(Importer.VCARD_2)) {
                return true;
            } else {
```

```
                return false;
            }
        }

        /**
         * Reads the VCard file and returns the version in a Structure enumeration.
         * @param f the VCard file to check the version of.
         * @return the version in a Structure enumeration
         */
        public static Structure getVCardVersion(File f) {
               // read part of the file and determine it's vcard version?
            try {
                String file = "";
                String line = "";
                BufferedReader reader = new BufferedReader(new FileReader(f));
                while ((line = reader.readLine()) != null) {
                    file += line;
                }
                if (file.length() == 0) {
                    return Structure.VCARD30;
                } else if (file.contains("VERSION:3.0")) {
                    return Structure.VCARD30;
                } else if (file.contains("VERSION:2.1")) {
                    return Structure.VCARD21;
                }
            } catch (IOException e) {
                JOptionPane.showMessageDialog(null, "Could not get VCard version from
file:\r\n " + f.getAbsolutePath(), "Error: ", JOptionPane.ERROR_MESSAGE);
            }
            return null;
        }
}

package org.ag.addressbook;

import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;

import org.ag.addressbook.property.Address;
import org.ag.addressbook.property.EmailAddress;
import org.ag.addressbook.property.TelephoneNumber;

/**
 * Exporter
 * This is used to save address book isntances to a file.
 * You can override or edit this class to implement extra data/file structures.
 * @author Ashley Gwinnell
 */
public class Exporter
{
        private File f;
        private ArrayList<Contact> contacts;

        /**
         * Create a new Exporter.
         */
        public Exporter(File f, ArrayList<Contact> contacts)
        {
            this.f = f;
            this.contacts = contacts;
        }

        /**
         * Write the list of contacts to the file, overwriting the file.
         * This method should determine the file type and how to write that file.
```

```
	 */
	public void write() {
		String ext = Importer.getFileExtension(this.f);
		if (ext.equals(Importer.BUAB)) {
			this.write(Structure.BUAB);
		} else if (ext.equals(Importer.VCARD_1) || ext.equals(Importer.VCARD_2)) {
			Structure s = Importer.getVCardVersion(this.f);
			if (s == null) {
				return;
			}
			this.write(s);
		}
	}

	/**
	 * Writes to the file from a structure enum.
	 * @param s
	 */
	private void write(Structure s) {
		if (s.equals(Structure.BUAB)) {
			this.writeBUAB();
		} else if (s.equals(Structure.VCARD21)) {
			 this.writeVCARD21();
		} else if (s.equals(Structure.VCARD30)) {
			this.writeVCARD30();
		}
	}

	/**
	 * Write a BUAB file.
	 */
	private void writeBUAB() {
		//System.out.println("Writing BUAB File!");
		try {
			String file = "";
			for (int i = 0; i < this.contacts.size(); i++) {
				Contact c = this.contacts.get(i);
				file += c.getForenames() + " " + c.getSurname() + "\r\n" +
						this.BUAB_getTelephoneNumber(c,
TelephoneNumber.Type.HOME, true) + "\r\n" +
						this.BUAB_getTelephoneNumber(c,
TelephoneNumber.Type.CELL, false) + "\r\n" +
						this.BUAB_getPreferredAddress(c) + "\r\n";
			}
			BufferedWriter writer = new BufferedWriter(new FileWriter(this.f));
			writer.write(file);
			writer.close();
		} catch (IOException e) {
			e.printStackTrace();
		}
	}

	/**
	 * Write a VCard2.1 file.
	 */
	private void writeVCARD21() {
		//System.out.println("Writing VCard File V2.1!");
	}

	/**
	 * Write a VCard3.0 file.
	 */
	private void writeVCARD30() {
		try {
			String file = "";
			for (int i = 0; i < this.contacts.size(); i++) {
				Contact c = this.contacts.get(i);
				String emails = "";
```

```
                        for (int j = 0; j < c.getEmailAddresses().size(); j++) {
                              EmailAddress ea = c.getEmailAddresses().get(j);
                              emails += "EMAIL;TYPE=INTERNET";
                              if (ea.getType().equals(EmailAddress.Type.HOME)) {
                                    emails += ";TYPE=HOME";
                              } else if (ea.getType().equals(EmailAddress.Type.WORK)) {
                                    emails += ";TYPE=WORK";
                              }
                              if (ea.isPreferred()) {
                                    emails += ";TYPE=PREF";
                              }
                              emails += ":" + ea.getAddress() + "\r\n";
                        }

                        String telephones = "";
                        for (int j = 0; j < c.getTelephoneNumbers().size(); j++) {
                              TelephoneNumber t = c.getTelephoneNumbers().get(j);
                              telephones += "TEL";
                              if (t.getType().equals(TelephoneNumber.Type.HOME)) {
                                    telephones += ";TYPE=HOME";
                              } else if (t.getType().equals(TelephoneNumber.Type.WORK))
{
                                    telephones += ";TYPE=WORK";
                              } else if (t.getType().equals(TelephoneNumber.Type.CELL))
{
                                    telephones += ";TYPE=CELL";
                              }
                              if (t.isPreferred()) {
                                    telephones += ";TYPE=PREF";
                              }
                              telephones += ":" + t.getNumber() + "\r\n";
                        }

                        String addresses = "";
                        for (int j = 0; j < c.getAddresses().size(); j++) {
                              Address a = c.getAddresses().get(j);
                              addresses += a.toString(c, Structure.VCARD30);
                        }

                        file += "BEGIN:VCARD\r\n" +
                                    "VERSION:3.0\r\n" +
                                    "N:" + c.getFamilyName() + ";" + c.getGivenName() +
";" + c.getAdditionalNames() + ";" + c.getPrefixes() + ";" + c.getSuffixes() + "\r\n" +
                                    "FN:" + c.getName() + "\r\n" +
                                    emails +
                                    telephones +
                                    addresses +
                                    "END:VCARD\r\n";
                  }
                  BufferedWriter writer = new BufferedWriter(new FileWriter(this.f));
                  writer.write(file);
                  writer.close();
            } catch (IOException e) {
                  e.printStackTrace();
            }
      }

      /**
       * Attempt to get the preferred telephone number
       * for BUAB backwards compatibility.
       * @param c The contact
       * @param t The Type
       * @param preferred Whether it must be preferred or not
       * @return The preferred best effort telephone number.
       */
      private String BUAB_getTelephoneNumber(Contact c, TelephoneNumber.Type t, boolean
preferred) {
            if (c.getTelephoneNumbers().size() == 0) {
```

```
                return "";
            } else {
                // look for preferred home number.
                for (int i = 0; i < c.getTelephoneNumbers().size(); i++) {
                    if (c.getTelephoneNumbers().get(i).getType().equals(t)) {
                        if (preferred) {
                            if (c.getTelephoneNumbers().get(i).isPreferred()) {
                                return
c.getTelephoneNumbers().get(i).getNumber();
                            }
                        } else {
                            return c.getTelephoneNumbers().get(i).getNumber();
                        }
                    }
                }
                return "";
            }
        }

        /**
         * Similar principle to BUAB_getTelephoneNumber.
         * @param c the contact
         * @return the best effort address for the contact.
         */
        private String BUAB_getPreferredAddress(Contact c) {
            if (c.getAddresses().size() == 0) {
                return "";
            } else {
                // look for preferred home number.
                for (int i = 0; i < c.getAddresses().size(); i++) {
                    if (c.getAddresses().get(i).isPreferred()) {
                        String buabline = "";
                        buabline +=
c.getAddresses().get(i).getStreetAddress().replace("\n", ", ");
                        return buabline;
                    }
                }
                return "";
            }
        }
}

package org.ag.addressbook;

/**
 * A Enumeration of the valid file structures.
 * @author Ashley Gwinnell
 */
public enum Structure {
        BUAB, VCARD21, VCARD30
}


package org.ag.addressbook;

import org.ag.util.undoredo.ActionStack;

/**
 * ABActionStack is an extension to the ActionStack class that I have
 * created and abstracted from this application's package.
 *
 * It extends the functionality of refreshUI to change the address book's
 * frame title.
 *
 * @author Ashley Gwinnell
 */
public class ABActionStack extends ActionStack
```

```
{
	public AddressBook addressBook;

	/**
	 * Create a new ABActionStack
	 * @param ab The AddressBook instance.
	 */
	public ABActionStack(AddressBook ab) {
		this.addressBook = ab;
	}

	@Override
	/**
	 * Extended functionality to change the window/frame's title.
	 */
	public void refreshUI() {
		super.refreshUI();
		if (this.addressBook.getFrame() == null) {
			return;
		}
		if (this.addressBook.getSavedAtStackLocation() != -1) {
			if (this.addressBook.getSavedAtStackLocation() != this.getTop()) {
				this.addressBook.getFrame().setTitle("Address Book - " +
this.addressBook.getCurrentlyOpenedFile().getAbsolutePath() + "(*)");
			} else {
				this.addressBook.getFrame().setTitle("Address Book - " +
this.addressBook.getCurrentlyOpenedFile().getAbsolutePath());
			}
		} else {
			if (this.getTop() != 0) {
				this.addressBook.getFrame().setTitle("Address Book - Untitled
Document(*)");
			} else {
				this.addressBook.getFrame().setTitle("Address Book - Untitled
Document");
			}
		}
	}
}


package org.ag.addressbook;

import java.awt.Dimension;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;
import java.io.File;

import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.WindowConstants;

import org.ag.addressbook.filter.BUABFileFilter;
import org.ag.addressbook.filter.VCardFileFilter;
import org.ag.util.undoredo.Action;

/**
 * ABDialogs contains a bunch of static methods to create various
```

```
 * frames in the aplication.
 * @author Ashley Gwinnell
 */
public class ABDialogs
{
	/**
	 * The dialog to show when a file has an invalid file extension.
	 * @param parent The parent window/frame.
	 * @param f The File that has the invalid extension.
	 */
	public static void createAndShowInvalidExtensionDialog(JFrame parent, File f) {
		JOptionPane.showMessageDialog(	parent,
				"You file you specified was an invalid type.\r\n" +
				"Only .buab .vcf and .vcard are supported!",
				"Error",
				JOptionPane.ERROR_MESSAGE);
	}

	/**
	 * The dialog to show when the user has unsaved changes.
	 * @param frame The parent window/frame.
	 * @return The JOptionPane response from YES_NO_CANCEL.
	 */
	public static int createAndShowUnsavedChangesDialog(JFrame frame) {
		return JOptionPane.showConfirmDialog(frame, "You have unsaved changes, would
you like to save?", "Warning: ", JOptionPane.YES_NO_CANCEL_OPTION,
JOptionPane.WARNING_MESSAGE);
	}

	/**
	 * The dialog to show when a user wants to open a file.
	 * @param frame The parent window/frame.
	 * @param canSelectMultiple Whether the user is allowed to select multiple files.
	 * @param title The Dialog's title.
	 * @return An array of the selected files.
	 */
	public static File[] createAndShowOpenFileSelector(JFrame frame, boolean
canSelectMultiple, String title) {

		JFileChooser chooser = new JFileChooser();
		chooser.setName(title);
		chooser.setDialogTitle(title);
		chooser.setMultiSelectionEnabled(canSelectMultiple);
		chooser.setAcceptAllFileFilterUsed(true);
		chooser.addChoosableFileFilter(new BUABFileFilter());
		chooser.addChoosableFileFilter(new VCardFileFilter());
		chooser.setFileFilter(chooser.getAcceptAllFileFilter());
		int returnVal = chooser.showOpenDialog(frame);
		if (returnVal == JFileChooser.APPROVE_OPTION) {
			if (canSelectMultiple) {
				File[] files = chooser.getSelectedFiles();
				return files;
			} else {
				File[] files = {chooser.getSelectedFile()};
				return files;
			}
		}
		return null;
	}

	/**
	 * The dialog to show when the user wants to save a new document or wants to save
as.
	 * @param frame The parent window/frame.
	 * @return The file that the user has chosen to save to.
	 */
	public static File createAndShowSaveFileSelector(JFrame frame) {
		File f = null;
```

```
            JFileChooser chooser = new JFileChooser();
            chooser.setMultiSelectionEnabled(false);
            chooser.setAcceptAllFileFilterUsed(false);
            chooser.addChoosableFileFilter(new BUABFileFilter(true));
            chooser.addChoosableFileFilter(new VCardFileFilter(true));
            while (true) {
                  int returnVal = chooser.showDialog(frame, "Save File");
                  if (returnVal == JFileChooser.APPROVE_OPTION) {
                        File file = chooser.getSelectedFile();
                        if (( chooser.getFileFilter() instanceof BUABFileFilter
                                          && !
Importer.getFileExtension(file).equals(Importer.BUAB))
                                    ||
                                    (chooser.getFileFilter() instanceof VCardFileFilter
                                          && !
Importer.getFileExtension(file).equals(Importer.VCARD_1)
                                          && !
Importer.getFileExtension(file).equals(Importer.VCARD_2))
                                    ) {
                                    if (Importer.getFileExtension(file).equals("")) {
                                          if (chooser.getFileFilter() instanceof
BUABFileFilter) {
                                                file = new File(file.getAbsolutePath() +
".buab");
                                          } else if (chooser.getFileFilter() instanceof
VCardFileFilter) {
                                                file = new File(file.getAbsolutePath() +
".vcard");
                                          }
                                    } else {
                                          JOptionPane.showMessageDialog(     chooser,
                                                                                "File
extension used does not match filter.\r\n" +
                                                                                "
VCard File must use .vcard or .vcf.\r\n" +
                                                                                "
BUAB File must use .buab.\r\n" +
                                                                                "You
can leave the extension blank and we'll do the work!",
                                                                                "Erro
r:",
      JOptionPane.ERROR_MESSAGE);
                                          continue;
                                    }
                              }

                              if (!file.exists()) {
                                    f = file;
                                    break;
                              } else {
                                int confirm = JOptionPane.showConfirmDialog(chooser, "Overwrite
file? " + file.getAbsolutePath());
                              if (confirm == JOptionPane.OK_OPTION) {
                                  f = file;
                              } else if (confirm == JOptionPane.NO_OPTION) {
                                  continue;
                              }
                              break;
                        }
                        } else { // wasn't approved?!
                              break;
                        }
            }
            return f;
      }

      /**
```

```
	 * The dialog to show to add a new contact to the Address Book.
	 * @param ab The AddressBook instance.
	 */
	public static void createAndShowAddDialog(final AddressBook ab) {
		final JDialog frame_add = new JDialog(ab.getFrame(), "Add Contact: ", true,
null);
		frame_add.setIconImage(new ImageIcon("files/User.png").getImage());
		frame_add.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
		frame_add.setResizable(false);
		frame_add.setLayout(null);

		int current_y = 10;

		JLabel lbl_addcontact = new JLabel("<html><b>Add Contact</b></html>");
		lbl_addcontact.setBounds(10, current_y, 150, 20);
		frame_add.add(lbl_addcontact);

		current_y += 30;

		JLabel lbl_forenames = new JLabel("Forenames: ");
		lbl_forenames.setBounds(10, current_y, 100, 20);
		frame_add.add(lbl_forenames);

		final JTextField tf_forenames = new JTextField();
		tf_forenames.setBounds(110, current_y, 175, 20);
		frame_add.add(tf_forenames);

		current_y += 30;

		JLabel lbl_surname = new JLabel("Surname: ");
		lbl_surname.setBounds(10, current_y, 100, 20);
		frame_add.add(lbl_surname);

		final JTextField tf_surname = new JTextField();
		tf_surname.setBounds(110, current_y, 175, 20);
		frame_add.add(tf_surname);

		current_y += 30;

		JButton btn_add = new JButton("Add Contact");
		btn_add.setBounds(10, current_y, 275, 30);
		btn_add.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				// validation?
				if (tf_forenames.getText().trim().equals("")) {
					JOptionPane.showMessageDialog(ab.getFrame(), "Forename(s)
cannot be left empty!", "Error:", JOptionPane.ERROR_MESSAGE);
					return;
				} //else if (tf_surname.getText().trim().equals("")) {
					//JOptionPane.showMessageDialog(frame, "Surname cannot be
left empty!", "Error:", JOptionPane.ERROR_MESSAGE);
					//return;
				//}
				ab.getActionStack().push(new Action() {
					public String forenames = tf_forenames.getText();
					public String surname = tf_surname.getText();
					public Contact c = new Contact();
					public String getUndoText() {
						return "Remove Contact";
					}
					public String getRedoText() {
						return "Add Contact";
					}
					public void doAction() {
						c.setForenames(forenames);
						c.setSurname(surname);
						ab.getContacts().add(c);
						ab.refreshList();
```

```
                        }
                        public void undoAction() {
                              ab.getContacts().remove(this.c);
                              ab.refreshList();
                        }
                  });


                  frame_add.dispose();
            }
      });
      frame_add.add(btn_add);

      frame_add.setSize(300, current_y + 65);
      frame_add.setLocationRelativeTo(ab.getFrame());

      frame_add.setVisible(true);
}

/**
 * The application's About dialog shown when F1 is pressed or when the
 * user selects Help > About.
 * @param frame The parent window/frame.
 */
public static void createAndShowAboutDialog(JFrame frame) {
      JDialog dialog = new JDialog(frame, "About: ", true, null);
      dialog.setIconImage(new ImageIcon("files/Info.png").getImage());
      dialog.setSize(400,250);
      dialog.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
      dialog.setLocationRelativeTo(frame);
      dialog.setResizable(false);
      JTextArea a = new JTextArea("Address Book\r\nVersion 0.6\r\n\r\n" +
                  "Free Icons by Axialis Team.\r\n" +
                  "http://www.axialis.com/free/icons/\r\n\r\n" +
                  "Created by Ashley Gwinnell.\r\n" +
                  "Website: http://www.ashleygwinnell.co.uk/\r\n" +
                  "Feedback: info@ashleygwinnell.co.uk\r\n\r\n" +
                  "Copyright 2009.");
      a.setEditable(false);
      dialog.add(a);
      dialog.setVisible(true);
}


/** The response from createAndShowImportDialog() to combine all the contacts */
public static final int IMPORT_DIALOG_COMBINE_ALL = 0;
/** The response from createAndShowImportDialog() to combine one of the contacts
*/
public static final int IMPORT_DIALOG_COMBINE_ONE = 1;
/** The response from createAndShowImportDialog() to replace all the contacts */
public static final int IMPORT_DIALOG_REPLACE_ALL = 2;
/** The response from createAndShowImportDialog() to replace one of the contacts
*/
public static final int IMPORT_DIALOG_REPLACE_ONE = 3;
/** The response from createAndShowImportDialog() to keep all the existing
contacts */
public static final int IMPORT_DIALOG_KEEP_ALL = 4;
/** The response from createAndShowImportDialog() to keep one of the existing
contacts */
public static final int IMPORT_DIALOG_KEEP_ONE = 5;
/** used internally do not change. */
private static int IMPORT_DIALOG_response = -1;

/**
 * The dialog to show when a duplicate contact is found on import.
 * @param addressBook The AddressBook instance.
 * @param parent The dialog's parent window/frame.
```

```
	 * @param title The title of the dialog.
	 * @param existing The existing contact.
	 * @param replacement The replacement contact
	 * @param fromFile Which file the replacement can be found in.
	 * @return either IMPORT_DIALOG_COMBINE_ALL, IMPORT_DIALOG_COMBINE_ONE ,
IMPORT_DIALOG_REPLACE_ALL, IMPORT_DIALOG_REPLACE_ONE , IMPORT_DIALOG_KEEP_ALL or
IMPORT_DIALOG_KEEP_ONE
	 */
	public static int createAndShowImportDialog(AddressBook addressBook, JFrame
parent, String title, Contact existing, Contact replacement, File fromFile)
	{
		final JDialog dialog = new JDialog(parent, title, true, null);
		dialog.setSize(new Dimension(670, 480));
		dialog.setLocationRelativeTo(parent);
		dialog.setIconImage(new ImageIcon("files/User.png").getImage());
		dialog.setDefaultCloseOperation(WindowConstants.DO_NOTHING_ON_CLOSE);
		dialog.addWindowListener(new WindowListener() {
			public void windowActivated(WindowEvent e) { }
			public void windowClosed(WindowEvent e) { }
			public void windowDeactivated(WindowEvent e) { }
			public void windowDeiconified(WindowEvent e) { }
			public void windowIconified(WindowEvent e) { }
			public void windowOpened(WindowEvent e) { }
			public void windowClosing(WindowEvent e) {
				if (IMPORT_DIALOG_response == -1) {
					JOptionPane.showMessageDialog(dialog, "Please select a
displayed option.", "Error:", JOptionPane.ERROR_MESSAGE);
				}
			}
		});
		dialog.setResizable(false);
		dialog.setLayout(null);

		JLabel lbl_title = new JLabel("<html><big>Duplicate Contact
Found:</big></html> ");
		lbl_title.setBounds(10, 10, 400, 35);
		dialog.add(lbl_title);

		JLabel lbl_existing = new JLabel("<html><b>Existing:</b></html>");
		lbl_existing.setBounds(46, 50, 100, 25);
		dialog.add(lbl_existing);

		ContactPanel pnl_existing = new ContactPanel(addressBook);
		pnl_existing.repopulate(existing);
		pnl_existing.setEnabled(false);
		pnl_existing.validate();
		JScrollPane scr_existing = new JScrollPane(pnl_existing);
		scr_existing.setBounds(110, 50, 540, 140);
		dialog.add(scr_existing);

		JLabel lbl_replacement = new JLabel("<html><b>Replacement:</b></html>");
		lbl_replacement.setBounds(15, 200, 100, 25);
		dialog.add(lbl_replacement);

		ContactPanel pnl_replacement = new ContactPanel(addressBook);
		pnl_replacement.repopulate(replacement);
		pnl_replacement.setEnabled(false);
		pnl_replacement.validate();
		JScrollPane scr_replacement = new JScrollPane(pnl_replacement);
		scr_replacement.setBounds(110, 200, 540, 140);
		dialog.add(scr_replacement);


		JButton combine_all = new JButton("Combine All Duplicates");
		JButton combine_one = new JButton("Combine One");
		combine_one.setBounds(110, 360, 150, 30);
		combine_all.setBounds(110, 395, 150, 30);
		combine_one.addActionListener(new ActionListener() { public void
```

```
actionPerformed(ActionEvent e) {
                  IMPORT_DIALOG_response = IMPORT_DIALOG_COMBINE_ONE;
                  dialog.dispose();
            }});
            combine_all.addActionListener(new ActionListener() { public void
actionPerformed(ActionEvent e) {
                  IMPORT_DIALOG_response = IMPORT_DIALOG_COMBINE_ALL;
                  dialog.dispose();
            }});
            dialog.add(combine_all);
            dialog.add(combine_one);

            JButton replace_all = new JButton("Replace All Duplicates");
            JButton replace_one = new JButton("Replace One");
            replace_one.setBounds(265, 360, 150, 30);
            replace_all.setBounds(265, 395, 150, 30);
            replace_one.addActionListener(new ActionListener() { public void
actionPerformed(ActionEvent e) {
                  IMPORT_DIALOG_response = IMPORT_DIALOG_REPLACE_ONE;
                  dialog.dispose();
            }});
            replace_all.addActionListener(new ActionListener() { public void
actionPerformed(ActionEvent e) {
                  IMPORT_DIALOG_response = IMPORT_DIALOG_REPLACE_ALL;
                  dialog.dispose();
            }});
            dialog.add(replace_all);
            dialog.add(replace_one);

            JButton keep_all = new JButton("Keep Existing Contacts");
            JButton keep_one = new JButton("Keep One");
            keep_one.setBounds(420, 360, 150, 30);
            keep_all.setBounds(420, 395, 150, 30);
            keep_one.addActionListener(new ActionListener() { public void
actionPerformed(ActionEvent e) {
                  IMPORT_DIALOG_response = IMPORT_DIALOG_KEEP_ONE;
                  dialog.dispose();
            }});
            keep_all.addActionListener(new ActionListener() { public void
actionPerformed(ActionEvent e) {
                  IMPORT_DIALOG_response = IMPORT_DIALOG_KEEP_ALL;
                  dialog.dispose();
            }});
            dialog.add(keep_all);
            dialog.add(keep_one);

            dialog.setVisible(true);

            int r = IMPORT_DIALOG_response;
            IMPORT_DIALOG_response = -1;
            return r;
      }


}
```

```
package org.ag.addressbook;
import java.awt.Color;
import java.awt.Dimension;

import javax.swing.JPanel;

/**
 * AB Separator
 * This is just a JPanel with a specified background
 * and size to cut down on lines.
 * @author Ashley Gwinnell
```

```
 */
public class ABSeparator extends JPanel
{
	/**
	 * Create a new separator.
	 */
	public ABSeparator()
	{
		this.setBackground(Color.LIGHT_GRAY);
		this.setPreferredSize(new Dimension(10,30));
	}
}
```

```
package org.ag.addressbook.filter;
import java.io.File;

import javax.swing.filechooser.FileFilter;

import org.ag.addressbook.Importer;

/**
 * A FileFilter to filter out files that aren't .BUAB
 * @author Ashley Gwinnell
 */
public class BUABFileFilter extends FileFilter {

	private boolean singular = false;
	public BUABFileFilter() {
		this(false);
	}
	public BUABFileFilter(boolean singular) {
		this.singular = singular;
	}
	@Override
	public boolean accept(File f) {
		if (f.isDirectory()) { return true; }
		String extension = Importer.getFileExtension(f);
		if (extension != null) {
			if (extension.equals(Importer.BUAB)
					|| extension.equals("lnk")) {
				return true;
			}
			return false;
		}
		return false;
	}

	@Override
	public String getDescription() {
		if (this.singular) {
			return "BUAB File";
		} else {
			return "BUAB Files";
		}
	}

}
```

```
package org.ag.addressbook.filter;
import java.io.File;

import javax.swing.filechooser.FileFilter;

import org.ag.addressbook.Importer;

/**
```

```
 * A FileFilter to filter out files that aren't .vcf or .vcard
 * @author Ashley Gwinnell
 */
public class VCardFileFilter extends FileFilter {

	public boolean singular = false;
	public VCardFileFilter() {
		this(false);
	}
	public VCardFileFilter(boolean singular) {
		this.singular = singular;
	}
	@Override
	public boolean accept(File f) {
		if (f.isDirectory()) { return true; }
		String extension = Importer.getFileExtension(f);
		if (extension != null) {
			if (extension.equals(Importer.VCARD_1)
					|| extension.equals(Importer.VCARD_2)
					|| extension.equals("lnk")) {
				return true;
			}
			return false;
		}
		return false;
	}

	@Override
	public String getDescription() {
		if (this.singular) {
			return "VCard File";
		} else {
			return "VCard Files";
		}
	}

}


package org.ag.addressbook.property;

import org.ag.addressbook.Contact;
import org.ag.addressbook.Structure;

/**
 * An Address that can be added to a Contact.
 * @author Ashley Gwinnell
 */
public class Address
{
	public enum Type {HOME, WORK};
	private Type type = Type.HOME;
	private boolean preferred = false;

	private String poBoxNumber = "";
	private String extendedAddress = "";
	private String streetAddress = "";
	private String city = ""; // locality / city.
	private String county = ""; // region / state / province / county.
	private String postcode = "";
	private String country = "";

	public Address() {

	}

	public Address(Type type, boolean preferred) {
		this.setType(type);
```

```
        this.setPreferred(preferred);
}

public void setPOBoxNumber(String poBoxNumber) {
        this.poBoxNumber = poBoxNumber;
}

public String getPOBoxNumber() {
        return poBoxNumber;
}

public void setExtendedAddress(String extendedAddress) {
        this.extendedAddress = extendedAddress;
}
public String getExtendedAddress() {
        return extendedAddress;
}

public void setStreetAddress(String streetAddress) {
        this.streetAddress = streetAddress;
}
public String getStreetAddress() {
        return streetAddress;
}

public void setCity(String city) {
        this.city = city;
}
public String getCity() {
        return city;
}

public void setCounty(String county) {
        this.county = county;
}
public String getCounty() {
        return county;
}

public void setPostcode(String postcode) {
        this.postcode = postcode;
}
public String getPostcode() {
        return postcode;
}

public void setCountry(String country) {
        this.country = country;
}
public String getCountry() {
        return country;
}




public void setPreferred(boolean preferred) {
        this.preferred = preferred;
}
public boolean isPreferred() {
        return preferred;
}
public void setType(Type type) {
        this.type = type;
}
public Type getType() {
        return type;
}
```

```
	/*
	public void addLine(String s) {
		this.lines.add(s);
	}
	public void addLines(String[] s) {
		for (int i = 0; i < s.length; i++) {
			if (s[i].trim().length() != 0) {
				this.addLine(s[i].trim());
			}
		}
	}
	public void setLines(String[] s) {
		this.lines.clear();
		this.addLines(s);
	}
	public ArrayList<String> getLines() {
		return this.lines;
	}
	*/

	private String clean(Structure s, String str) {
		if (s.equals(Structure.VCARD30)) {
			return str.replace(",", "\\,").replace("\n", "\\n");
		}
		return "";
	}

	public String toString(Contact c, Structure f) {
		if (f.equals(Structure.BUAB)) {

		} else if (f.equals(Structure.VCARD30)) {
			// addr
			String line = "ADR";
			if (this.getType().equals(Address.Type.HOME)) {
				line += ";TYPE=HOME";
			} else if (this.getType().equals(Address.Type.WORK)) {
				line += ";TYPE=WORK";
			}
			if (this.isPreferred()) {
				line += ";TYPE=PREF";
			}
			line += ":";
			line += this.clean(Structure.VCARD30, this.getPOBoxNumber()) + ";";
			line += this.clean(Structure.VCARD30, this.getExtendedAddress()) +
";";
			line += this.clean(Structure.VCARD30, this.getStreetAddress()) + ";";
			line += this.clean(Structure.VCARD30, this.getCity()) + ";";
			line += this.clean(Structure.VCARD30, this.getCounty()) + ";";
			line += this.clean(Structure.VCARD30, this.getPostcode()) + ";";
			line += this.clean(Structure.VCARD30, this.getCountry()) + "\r\n";

			// label
			line += "LABEL";
			if (this.getType().equals(Address.Type.HOME)) {
				line += ";TYPE=HOME";
			} else if (this.getType().equals(Address.Type.WORK)) {
				line += ";TYPE=WORK";
			}
			if (this.isPreferred()) {
				line += ";TYPE=PREF";
			}

			String label_line = "";
			label_line += ":";
			label_line += c.getName() + "\\n";
			if (this.getPOBoxNumber().trim().length() != 0) {
				label_line += this.getPOBoxNumber() + "\\n";
			} if (this.getExtendedAddress().trim().length() != 0) {
```

```
                        label_line += this.getExtendedAddress() + "\\n";
                  } if (this.getStreetAddress().trim().length() != 0) {
                        label_line += this.clean(Structure.VCARD30,
this.getStreetAddress()) + "\\n";
                  } if (this.getCity().trim().length() != 0) {
                        label_line += this.getCity() + "\\n";
                  } if (this.getCounty().trim().length() != 0) {
                        label_line += this.getCounty() + "\\n";
                  } if (this.getPostcode().trim().length() != 0) {
                        label_line += this.getPostcode() + "\\n";
                  } if (this.getCountry().trim().length() != 0) {
                        label_line += this.getCountry() + "\\n";
                  }
                  if (label_line.substring(label_line.length()-2,
label_line.length()).equals("\\n")) {
                        label_line = label_line.substring(0, label_line.length()-2);
                  }
                  label_line += "\r\n";
                  return line + label_line;
            }
            return "";
      }

      public String toSearchString() {
            String searchString = new String("");
            searchString += poBoxNumber + " " + extendedAddress + " " +
                              streetAddress + " " + city + " " +
                              county + " " + postcode + " " + country;
            return searchString;
      }
}

package org.ag.addressbook.property;

/**
 * An EmailAddress that can be added to a Contact.
 * @author Ashley Gwinnell
 */
public class EmailAddress
{
      private boolean preferred = false;
      private Type type = Type.HOME;
      public enum Type {HOME, WORK};
      private String address;

      public EmailAddress() {

      }
      public EmailAddress(String address, boolean preferred, Type t) {
            this.address = address;
            this.preferred = preferred;
            this.type = t;
      }

      public void setAddress(String address) {
            this.address = address;
      }
      public String getAddress() {
            return address;
      }

      public void setType(Type type) {
            this.type = type;
      }
      public Type getType() {
            return type;
      }
```

```
	public void setPreferred(boolean preferred) {
		this.preferred = preferred;
	}
	public boolean isPreferred() {
		return preferred;
	}

	public String toString() {
		return new String(this.address + " " + preferred + " " + type);
	}

	public String toSearchString() {
		return new String(this.address);
	}

}


package org.ag.addressbook.property;

/**
 * An Telephone Number that can be added to a Contact.
 * @author Ashley Gwinnell
 */
public class TelephoneNumber
{
	private boolean preferred = false;
	private String number;
	private Type type = Type.HOME;
	public enum Type {HOME, CELL, WORK};

	public TelephoneNumber() {

	}
	public TelephoneNumber(String number, boolean preferred, Type t) {
		this.number = number;
		this.preferred = preferred;
		this.type = t;
	}

	public String getNumber() {
		return number;
	}
	public Type getType() {
		return type;
	}
	public void setNumber(String number) {
		this.number = number;
	}
	public void setPreferred(boolean preferred) {
		this.preferred = preferred;
	}
	public void setType(Type type) {
		this.type = type;
	}
	public boolean isPreferred() {
		return preferred;
	}
	public String toString() {
		return new String(this.number + " " + preferred + " " + type);
	}
	public String toSearchString() {
		return new String(this.number);
	}
}

package org.ag.util;
```

```java
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.ArrayList;

/**
 * RecentFileSet
 * This is used to keep a record of recently opened files outside of runtime.
 * @author Ashley Gwinnell
 */
public class RecentFileSet
{
	private static int GENERATED_ID = 0;

	private int id;
	private int size;
	private ArrayList<String> items;
	private String filename;

	/**
	 * Create a RecentFileSet with a maximum of 8 recent files.
	 */
	public RecentFileSet()
	{
		this(8);
	}

	/**
	 * Create a RecentFileSet with the specified number of recent files.
	 * @param size The size of the RecentFileSet.
	 */
	public RecentFileSet(int size) {
		this.id = RecentFileSet.generateId();
		this.size = size;
		this.items = new ArrayList<String>();
		this.filename = new String("recentfiles_" + this.id + ".adr");
		this.items = this.get();
	}

	/**
	 * Used internally to keep a record of the recent file set used.
	 * WARNING: A program ideally should only have one RecentFileSet object per
runtime.
	 * @return
	 */
	private static int generateId() {
		int id = GENERATED_ID;
		GENERATED_ID++;
		return id;
	}

	/**
	 * Add a String to the recent file set.
	 * @param file The absolute path of the file to add to the recent fileset.
	 */
	public void add(String file)
	{
		if (this.items.contains(file)) {
			return;
		}
		this.items.add(file);
		if (this.items.size() > this.size) {
			this.items = (ArrayList<String>) this.items.subList(0, this.size);
```

```
        }

        FileOutputStream fos = null;
        ObjectOutputStream out = null;
        try
        {
                fos = new FileOutputStream(this.filename);
                out = new ObjectOutputStream(fos);
                out.writeObject(this.items);
                out.close();
        }
        catch(IOException ex)
        {
                ex.printStackTrace();
        }
}

/**
 * Get a list of the recent files.
 * @return a list of the recent files.
 */
public ArrayList<String> get()
{
        ArrayList<String> localfiles = new ArrayList<String>();
        FileInputStream fis = null;
        ObjectInputStream in = null;
        try
        {
                fis = new FileInputStream(this.filename);
                in = new ObjectInputStream(fis);
                localfiles = (ArrayList<String>) in.readObject();
                in.close();
        }
        catch(FileNotFoundException ex)
        {
                // no recent files!
        }
        catch(ClassNotFoundException ex)
        {
                ex.printStackTrace();
        }
        catch(IOException ex)
        {
                ex.printStackTrace();
        }
        return localfiles;
}

/**
 * Determines whether a String is in the recent file set.
 * @param filename the String (filename) to check.
 * @return true if it is a recent file.
 */
public boolean isRecentFile(String filename)
{
        return this.isRecentFile(new File(filename));
}

/**
 * Determines whether a File is in the recent file set.
 * @param file the File to check.
 * @return true if it is a recent file.
 */
public boolean isRecentFile(File file) {
        ArrayList<String> files = this.get();
        for (int i = 0; i < files.size(); i++) {
                if (file.getAbsoluteFile().equals(files.get(i))) {
                        return true;
```

```
                }
            }
            return false;
      }

}

package org.ag.util;

import java.io.File;
import java.util.ArrayList;
import java.util.prefs.Preferences;

/**
 *
 * @author Ashley Gwinnell
 * @deprecated
 * @see {@link RecentFileSet}
 */
public class ~~RecentFileSetPreferences~~
{
      private Preferences preferences;

      public RecentFileSetPreferences()
      {
            this(~~RecentFileSetPreferences~~.class);
      }

      public RecentFileSetPreferences(Class<?> c) {
            this.preferences = Preferences.userNodeForPackage(c);
      }

      public void add(String file)
      {
            ArrayList<String> files = this.get();
            switch (files.size()) {
                  case 0:
                        this.preferences.put("recentfile_0", file);
                        break;
                  case 1:
                        this.preferences.put("recentfile_1",
this.preferences.get("recentfile_0", "null"));
                        this.preferences.put("recentfile_0", file);
                        break;
                  case 2:
                        this.preferences.put("recentfile_2",
this.preferences.get("recentfile_1", "null"));
                        this.preferences.put("recentfile_1",
this.preferences.get("recentfile_0", "null"));
                        this.preferences.put("recentfile_0", file);
                        break;
                  case 3:
                        this.preferences.put("recentfile_2",
this.preferences.get("recentfile_1", "null"));
                        this.preferences.put("recentfile_1",
this.preferences.get("recentfile_0", "null"));
                        this.preferences.put("recentfile_0", file);
                        break;
            }
      }

      public ArrayList<String> get()
      {
            ArrayList<String> files = new ArrayList<String>();
            String zero = this.preferences.get("recentfile_0", "null");
            if (!zero.equals("null")) {
                  files.add(zero);
            }
```

```
            String one = this.preferences.get("recentfile_1", "null");
            if (!one.equals("null")) {
                  files.add(one);
            }
            String two = this.preferences.get("recentfile_2", "null");
            if (!two.equals("null")) {
                  files.add(two);
            }
            return files;
      }

      public boolean isRecentFile(String filename)
      {
            return this.isRecentFile(new File(filename));
      }

      public boolean isRecentFile(File file) {
            ArrayList<String> files = this.get();
            for (int i = 0; i < files.size(); i++) {
                  if (file.getAbsoluteFile().equals(files.get(i))) {
                        return true;
                  }
            }
            return false;
      }

}


package org.ag.util;

import java.util.ArrayList;

/**
 * A bunch of String Utility methods.
 * @author Ashley Gwinnell
 */
public class StringUtil
{
      /**
       * String.split(String regex) doesn't do the job properly.
       * We want it to keep items that are empty!
       * @param subject The string to be splitting.
       * @param splitAt The character to split at!
       * @return The array of Strings after splitting!
       */
      public static String[] splitWithoutTrimming(String subject, char splitAt) {
            ArrayList<String> strings = new ArrayList<String>();
            int beginIndex = 0;
            for (int i = 0; i < subject.length(); i++) {
                  if (subject.charAt(i) == splitAt) {
                        strings.add(subject.substring(beginIndex, i));
                        beginIndex = i + 1;
                  }
            }
            strings.add(subject.substring(subject.lastIndexOf(';')+1));
            String[] strs = new String[strings.size()];
            for (int i = 0; i < strings.size(); i++) {
                  strs[i] = strings.get(i);
            }
            return strs;
      }
}


package org.ag.util;
```

```java
import java.util.ArrayList;

/**
 * A bunch of String Utility methods.
 * @author Ashley Gwinnell
 */
public class StringUtil
{
	/**
	 * String.split(String regex) doesn't do the job properly.
	 * We want it to keep items that are empty!
	 * @param subject The string to be splitting.
	 * @param splitAt The character to split at!
	 * @return The array of Strings after splitting!
	 */
	public static String[] splitWithoutTrimming(String subject, char splitAt) {
		ArrayList<String> strings = new ArrayList<String>();
		int beginIndex = 0;
		for (int i = 0; i < subject.length(); i++) {
			if (subject.charAt(i) == splitAt) {
				strings.add(subject.substring(beginIndex, i));
				beginIndex = i + 1;
			}
		}
		strings.add(subject.substring(subject.lastIndexOf(';')+1));
		String[] strs = new String[strings.size()];
		for (int i = 0; i < strings.size(); i++) {
			strs[i] = strings.get(i);
		}
		return strs;
	}
}
```

```java
package org.ag.util.undoredo;

import java.util.ArrayList;

import javax.swing.JButton;
import javax.swing.JMenuItem;

/**
 * ActionStack
 * This is used for the undo/redo stack.
 * It should be portable to any Java swing/awt application.
 * @author Ashley Gwinnell
 */
public class ActionStack
{
	private ArrayList<Action> stack = new ArrayList<Action>();
	private int top = 0;

	private JMenuItem m_undo, m_redo;
	private JButton tb_undo, tb_redo;
	private boolean autoRefreshingUI = true;

	/**
	 * Create a new ActionStack unlimited in size.
	 */
	public ActionStack() {

	}

	/**
	 * Set's the UI so the ActionStack can enable/disable appropriate elements
accordingly.
	 * @param m_undo The JMenuItem for "Undo".
	 * @param m_redo The JMenuItem for "Redo".
```

```
	 * @param tb_undo The JButton on the JToolBar for "Undo".
	 * @param tb_redo The JButton on the JToolBar for "Redo".
	 */
	public final void setUI(JMenuItem m_undo, JMenuItem m_redo, JButton tb_undo,
JButton tb_redo)
	{
		this.m_undo = m_undo;
		this.m_redo = m_redo;
		this.tb_undo = tb_undo;
		this.tb_redo = tb_redo;
	}

	/**
	 * If true, the UI will refresh on every action push/pop.
	 * @param autoRefreshingUI
	 */
	public final void setAutoRefreshingUI(boolean autoRefreshingUI) {
		this.autoRefreshingUI = autoRefreshingUI;
	}

	/**
	 * Checks whether the ActionStack is auto refreshing the UI on every action
push/pop.
	 * @return whether the ActionStack is auto refreshing the UI on every action
push/pop.
	 */
	public final boolean isAutoRefreshingUI() {
		return autoRefreshingUI;
	}

	/**
	 * Push the most recent action.
	 * This should be called by your "redo" items.
	 */
	public final void push() {
		this.push(this.stack.get(this.top));
	}

	/**
	 * Push an action.
	 * @param a The Action to push onto the ActionStack.
	 */
	public final void push(Action a) {
		try {
			stack.set(top, a);
		} catch (IndexOutOfBoundsException e) {
			stack.add(top, a);
		}
		a.doAction();
		top++;
		if (autoRefreshingUI) { this.refreshUI(); }
	}

	/**
	 * Pops and returns the most recent Action.
	 * This should be called by your "undo" items.
	 * @return the most recent Action.
	 */
	public final Action pop() {
		Action a = stack.get(top-1);
		a.undoAction();
		top--;
		if (top < 0) {
			top = 0;
		}
		if (autoRefreshingUI) { this.refreshUI(); }
		return a;
	}
```

```
	/**
	 * Gets the size of the ActionStack. i.e. the total number of Actions in the
stack.
	 * @return
	 */
	public final int getSize() {
		return stack.size();
	}

	/**
	 * Gets the pointer to the top of the ActionStack.
	 * @return
	 */
	public final int getTop() {
		return this.top;
	}

	/**
	 * Gets the action at the particular index of the stack.
	 * @param i The index to look at for an Action.
	 * @return the action at the particular cell of the stack.
	 */
	public final Action getAction(int i) {
		return this.stack.get(i);
	}

	/**
	 * Checks whether the ActionStack is at the most recent item.
	 * i.e. there is nothing to redo.
	 * @return true of the Actionstack is at the most recent item.
	 */
	public final boolean isAtTop() {
		return (this.top == this.getSize());
	}

	/**
	 * Checks whether the ActionStack is at the first added item.
	 * i.e. there is nothing to undo.
	 * @return
	 */
	public final boolean isAtBottom() {
		return (this.top == 0);
	}

	/**
	 * Clears the action stack of all Action items.
	 */
	public final void clear() {
		this.stack.clear();
		this.top = 0;
	}

	/**
	 * Refreshes the UI.
	 * This is called automatically on every push/pop if it is set
	 * to automatic refreshing.
	 */
	public void refreshUI()
	{
		if (m_redo == null || m_undo == null || tb_undo == null || tb_redo == null)
{
			System.out.println("SHITSHIT!");
			return;
		}
		if (this.getSize() == 0) {
			m_redo.setText("Redo");
			m_redo.setEnabled(false);
```

```
                    m_undo.setText("Undo");
                    m_undo.setEnabled(false);
                    tb_undo.setEnabled(false);
                    tb_redo.setEnabled(false);
               } else if (this.isAtTop()) {
                    m_redo.setText("Redo");
                    m_redo.setEnabled(false);
                    tb_redo.setEnabled(false);
                    if (this.getTop() >= 1) {
                         m_undo.setText("Undo (" + this.getAction(this.getTop()-
1).getUndoText() + ")");
                         m_undo.setEnabled(true);
                         tb_undo.setEnabled(true);
                    }
               } else if (!this.isAtTop() && !this.isAtBottom()) {
                    m_undo.setText("Undo (" + this.getAction(this.getTop()-
1).getUndoText() + ")");
                    m_undo.setEnabled(true);
                    m_redo.setText("Redo (" + this.getAction(this.getTop()).getRedoText()
+ ")");
                    m_redo.setEnabled(true);
                    tb_undo.setEnabled(true);
                    tb_redo.setEnabled(true);
               } else if (this.isAtBottom()) {
                    m_undo.setText("Undo");
                    m_undo.setEnabled(false);
                    tb_undo.setEnabled(false);
                    if (this.getSize() > 0) {
                         m_redo.setText("Redo (" +
this.getAction(this.getTop()).getRedoText() + ")");
                         m_redo.setEnabled(true);
                         tb_redo.setEnabled(true);
                    }
               }
          }
}
```