```
/**
 *  AddressBookGUI.java.  This is part of a simplified demonstration version
 *  of the address book showing the use of a displayable collection
 *
 *  Copyright (c) 2000, 2001, 2005, 2011 - Russell C. Bjork
 *
 */

package addressbook;

import displayablecollections.DisplayableSortedSet;
import displayablecollections.DisplayableTreeSet;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.io.IOException;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.FileWriter;
import java.io.PrintWriter;
import java.util.Iterator;

/** An object of this class allows interaction between the program and the
 *  human user.
 */
public class AddressBookGUI extends JFrame
{
    /** Main method for program
     */
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new AddressBookGUI().setVisible(true);
            }
        });
    }

    /** Constructor
     */
    public AddressBookGUI()
    {
        addressBook = new DisplayableTreeSet<Person>();

        createGUI();

        // Cannot be done until after components are created

        personList.setModel(addressBook.orderedListModel());
        setTitle(UNTITLED);
        saveItem.setEnabled(false);

        pack();
    }
```

```
/** Do the Add a Person use case
 *
 *  @return whether the add operation was actually performed, rather than
 *          being cancelled
 */
private boolean doAdd()
{
    Person newPerson = readPerson(null);

    if (newPerson != null)
    {
        addressBook.add(newPerson);
        return true;
    }
    else
        return false;
}

... Edit and Delete cases omitted

/** Do the New Address Book Use Case
 *  @return whether the new address book operation was actually performed,
 *          rather than being cancelled
 */
private boolean doNew()
{
    ... Checking for need to save omitted

    addressBook = new DisplayableTreeSet<Person>();
    personList.setModel(addressBook.orderedListModel());
    currentFile = null;
    return true;
}

/** Do the Open Address Book Use Case
 *  @return whether the open address book operation was actually performed,
 *          rather than being cancelled
 *  @exception IOException if there is any exception reading the file
 *  @exception ClassNotFoundException if a class referenced in the file
 *             cannot be found
 */
private boolean doOpen() throws IOException, ClassNotFoundException
{
    ... Checking for need to save omitted and choosing file to open omitted

        addressBook = (DisplayableSortedSet<Person>) stream.readObject();
        personList.setModel(addressBook.orderedListModel());

        defaultDirectory = toOpen.getParent();
        currentFile = toOpen;
        return true;

    ...
}

... Save, Save As, Print Mailing Labels, Find, Find Again Use cases and
... Offer to Save Changes Extension omitted

... GUI construction and action listeners omitted
```

```
/**
 *  Person.java.  This is part of a simplified demonstration version
 *  of the address book showing the use of a displayable collection
 *
 *  Copyright (c) 2005, 2011 - Russell C. Bjork
 *
 */

package addressbook;

import java.io.Serializable;
import java.io.PrintWriter;
import java.util.Comparator;

/** An object of this class maintains information about a single individual
 *  in the address book
 */
public class Person implements Serializable, Comparable<Person>
{
    ...

    /** Create a string representing this person
     *
     *  @return a string containing the information for this person
     */
    public String toString()
    {
        return getFullName() + " " + address + " " + city + " " + state + " " + zip;
    }

    /** Method required by the Comparable interface
     *
     *  @param other the other person to compare to
     */
    public int compareTo(Person other)
    {
        return getFullName().compareTo(other.getFullName());
    }
```