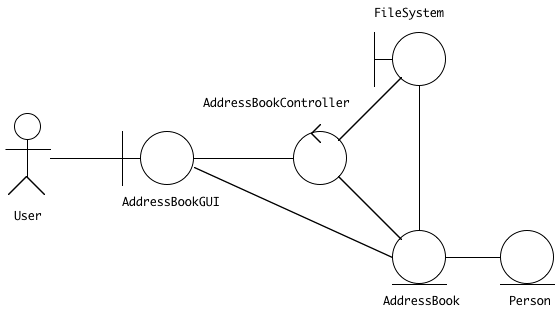
Address Book Analysis

An initial reading of the use cases suggests that the following will be part of the system.

* A single entity object representing the current address book that the program is working with (AddressBook).
* An arbitrary number of entity objects, each representing one of the people that is in the current address book (Person).
* A boundary object representing the interface between the address book system and the human user (AddressBookGUI).
* A boundary object representing the interface between the address book system and the file system on disk (FileSystem).
* A controller object that carries out the use cases in response to user gestures on the GUI (AddressBookController). (For a problem of this small size, a single controller is sufficient.)



The various use cases work with these objects, as follows:

* The Add a Person Use Case involves getting the new information from the user, and then telling the AddressBook object to add a new person with this information to its collection
* The Edit a Person Use Case involves displaying the current information about the desired person (obtained from the AddressBook), then allowing the user to enter new information for the various fields, then telling the AddressBook object to make the changes.
* The Delete a Person Use Case involves asking the user to confirm deletion, and then telling the AddressBook object to remove this person from its collection.
* The Sort Entries by Name Use Case involves telling the AddressBook object to rearrange its collection in order of name.
* The Sort Entries by ZIP Use Case involves telling the AddressBook object to rearrange its collection in order of ZIP.
* The Create New Address Book Use Case involves creating a new AddressBook object.
* The Open Existing Address Book Use Case involves getting a file specification from the user, and then telling the FileSystem object to read in an AddressBook object from this file.
* The Save Address Book Use Case involves determining whether or not the current AddressBook object has a file it was last read from / saved to; if so, telling the FileSystem object to save the current AddressBook object to this file. (If not, the Save Address Book As ... Use Case is done instead.)
* The Save Address Book As ... Use Case involves getting a file specification from the user, and then telling the FileSystem object to save the current AddressBook object to this file.
* The Print Address Book Use Case involves telling the AddressBook object to print out its collection in order.
* (The Quit Program Use Case does not involve any of the other objects)
* (The Offer to Save Changes Extension may involve performing the Save Address Book Use Case.)