**Detailed Design & Implementation**

**Andrew Henk, Isaac Gainey, Matt Westman**

**March 22nd, 2016**

**CEN 4072 Software Testing**

**Instructors: Dr. Ingrid Buckley**

**Software Engineering Department**

**Florida Gulf Coast University**

**Ft. Myers, FL 33965**

**i. Interfaces and Classes**

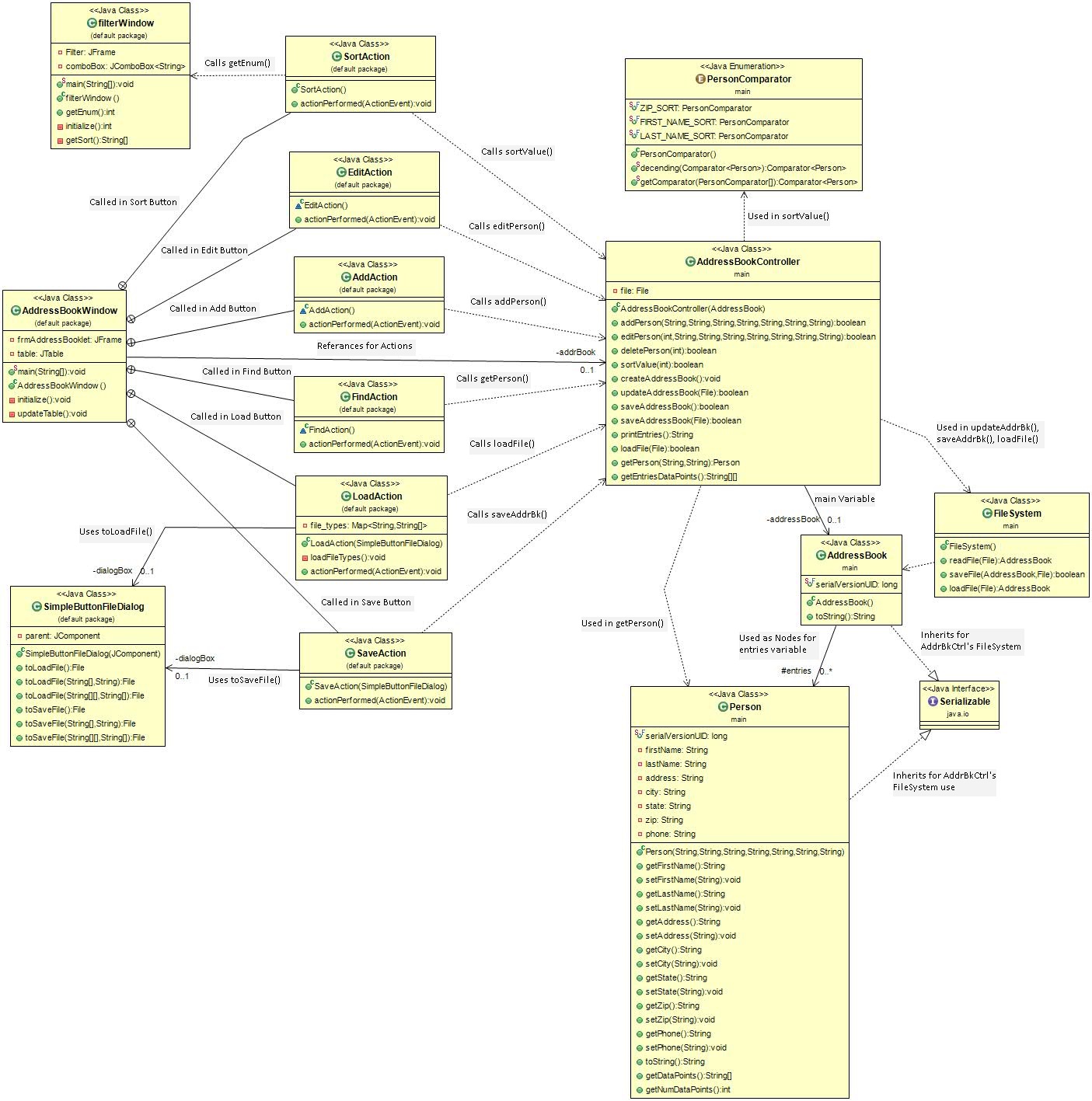
Interfaces

* FileSystem.java
  + Saves the AddressBook object and by extension Person objects to a file that is passed via arguments
  + Loads an AddressBook object and its collection from a file.
* AddressBookGUI.java
  + Uses java swing to display the AddressBook’s collection and allow easy editability.

Classes

* AddressBook.java
  + Responsible for keeping the collection of ‘Person’s as the contact list.
  + Inherits Serializable class for the file system class.
* AddressBookController.java
  + Attaches AddressBookGUI class, AddressBook class, and FileSystem interface
* Person.java
  + Holds all the contact information
    - Name
    - Address (street, state, zip etc)
    - Phone
  + Inherits Serializable class for the file system class.
* PersonComparator.java
  + Inherits Comparator class to enable default sort algorithms that collection classes use.

**ii. Class Diagram**



While the Controller class connects the GUI and AddressBook, there are methods that require the GUI to bypass the controller directly to the AddressBook. Excluding those expectations, the controller bridges the GUI and AddressBook. In which AddressBook has a collection of Person class which are the entries that the user would store their contact information in. The controller also calls the PersonComparator, which compares one attribute of two Persons and enables auto-organization for display purposes.

The FileSystem class is present as functions are directly references AddressBook and are called in the AddressBookController save and load functions. It is also the reason the AddressBook and Person class inherit Serializable, in order to save to a file as their current object state.

**iii. Dependencies**

Dependency Binding

public class AddressBookModule{

@Override

protected void configure() {

bind(Person.class).to(Serializable.class);

bind(AddressBook.class).to(Serializable.class);

}

}

Dependency Injection

public class AddressBookGUI extends JFrame{

@Inject

public AddressBookGUI (AddressBookController controller){…}

...

}

public class AddressBookController{

@Inject

public AddressBookController (AddressBook contactList){…}

...

}

**iv. Unit Test Plan (identify the mocks used in your testing)**

Each of the classes in the class diagram will be unit tested except for Serializable, Person.java, and PersonComparator.java

* AddressBookGui.java
* AddressBookController.java
* AddressBook.java
* FileSystem.java

For more, see the following test classes for integration test:

* AddressBookGuiTest.java
* AddressBookControllerTest.java
* AddressBookTest.java
* FileSystemTest.java

All the test classes end with “Test”.

We plan to achieve 90% code coverage and 80% branch coverage in unit testing. We will monitor this with EclEmma (http://eclemma.org/installation.html)

**v. Integration Test Plan**

The following classes will be tested with its dependencies in the integration test:

* AddressBookGui.java
* AddressBookController.java
* AddressBook.java

For more, see the following test classes for integration test:

* AddressBookIGuiTest.java
* AddressBookIControllerTest.java
* AddressBookITest.java

All the test classes end with “ITest”.

We plan to achieve 90% code coverage and 80% branch coverage for non-GUI functions with Integration Testing.

**vi. Splitting the Work**

The implementation of the project is split between all members. All members are responsible for unit testing their contributions. Once a class is complete, along with the dependency classes, the integration testing is performed by any of the members.

* Isaac Gainey
  + Implementation
    - AddressBookController.java
    - FileSystem.java
    - GUI package
  + Testing
    - AddressBookControllerTest.java
    - FileSystemTest.java
* Andrew Henk
  + Implementation
    - AddressBook.java
    - Person.java
    - PersonComparator.java
  + Testing
    - AddressBookTest.java
* Matt Westman
  + Testing
    - GUI.java

|  |  |
| --- | --- |
| Name | Contribution |
| Andrew Henk | Powerpoint |
| Isaac Gainey | Updated Class diagram, added link for eclEmma, a few Screen shots for powerpoint |
| Matt Westman |  |