SOFTWARE DESIGN SPECIFICATION
PROJECT 1 : ADDRESS BOOK

Edited by

Frazer Bayley
Haley Whitman
Abdulaziz Al-Heidous
Alison Legge
Jeremy Brennan

Project 1 - Team 3

Contents

Revision History

Revision	Date	Author(s)	Description
.1	23.01.17	Haley Whitman	Created initial outline of document
.1	25.01.17	Haley Whitman	Added all code snippets

1 Introduction

1.1 Intended Audience

The following document covers all functionality of the front and back end code and how they relate with the user's experience. This document is intended for programmers, team managers, and quality assurance on the developing team to ensure that the code is running to this specification.

1.2 How to Use this Document

This document is intended to help organize all modules, classes, and functions found in the Address Book code. It is meant to help all team members design their components to this specification, as well as having a straightforward method of analyzing

2 Summary

The entirety of the program is based off of Java JDK 8, which is using a TSV file format to both load and export address book files. This document itself was created from the list of customer specifications which, with customer meetings, guided the creation of the Software Requirements Analysis, which was used to create the requirements needed to begin programming. This document will help create an Quality Assurance documents needed to test the quality of the resulting application.
This document will contain all programming interactions between modules, classes, and functions. This will be achieved by both diagrams and showing all function headers and descriptions of what they achieve and their interactions.

3 User Interface Architecture

3.1 GUI Handbook

3.2 Expected Input

The user is prompted to enter values for the following variables via text fields:

1. First name
2. Last name
3. Phone number
4. Email
5. Address: Street and 2nd Address
6. City
7. State
8. ZIP code

Other use inputs are included as buttons and from file menus, which are accessed by mouse clicks.

1. Close (Both program, and for adding a contact)
2. Edit (Accessed by mouse clicking on a contact)
3. Delete (Accessed within the edit contact menu)
4. Search (Accessed by mouse clicking the search button next to the search bar)
5. New (Accessed in file menu, creates and loads a new address book)

6. Open (Accessed in file menu, opens an existing address book from a file directory)
7. Save (Accessed in file menu,)
8. Save As... (Accessed in file menu)
9. Quit (Accessed in file menu, quits application completely.)

3.2.1 Output

The user receives a list of contacts that they are able to view through, by scrolling a scroll bar on the side of the menu. They are able to access any visible contact by mouse clicks, and able to sort through this displayed list by sorting or searching. A user is allowed to edit or add a contact, which once changed can be elected to be saved and added to the current address book for future use. Upon loading any address book the current address book will be updated visually with the new address book.

3.3 GUI Window

```
public class AddressEntryFrame extends JFrame {}
```

```
public AddressEntryFrame() {}
```

3.3.1 Buttons

```
Void deleteButton() {}
```

```
void editButton() {}
```

```
public class NewEntry {}
```

```
public void addEntry() {}
```

```
public void closeWindow() {}
```

3.4 Back-End Architecture

3.5 Contact Class

The contact class is where all data regarding each entry in the address is kept. This class utilizes strings to hold all information about each entry.

3.5.1 Attributes of Contact Class

```
public Contact(String fn, String ln, String phone, String email,
        String add, String city, String state, String zip) { }
```

This class holds the String data for first name, last name, phone number, email, address, 2nd address, city, state, and ZIP code.

3.5.2 Functions of Contact Class

```
public String getFirstName() {
        return _firstName;
}
public void setFirstName(String value) {
        _firstName = value;
}
```

Functions: Get or set first name data for a Contact.
Precondition: All values are needed for postconditions accessible to this function
Postcondition: Contact is either updated with supplied information or is returned as a String.

```
public String getLastName() {
        return _lastName;
}
public void setLastName(String value) {
        _lastName = value;
}
```

```
public String getPhone() {
    return _phone;
}
public void setPhone(String value) {
    _phone = value;
}
```

```
public String getEmail() {
    return _email;
}
public void setEmail(String value) {
    _email = value;
}
```

```
public String getAddress() {
    return _address;
}
public void setAddress(String value) {
    _address = value;
}
```

```
public String getCity() {
    return _city;
}
public void setCity(String value) {
    _city = value;
}
```

```
public String getState() {
    return _state;
}
public void setState(String value) {
    _state = value;
}
```

```
public String getZip() {
    return _zip;
}
public void setZip(String value) {
    _zip = value;
}
```

3.5.3 AddressBook Class

```
AddressBook(String filepath) { }
```

```
public void Save(String filepath) { }
```

```
public void Add(Contact data) { }
```

```
public void Delete(Contact data) { }
```

```
public Contact SearchBy(Integer key) { }
```

3.5.4 Sorting

```
public void SortByName() { }
```

```
public void SortByZip() { }
```

```
public static Comparator<Contact> COMPARE_BY_NAME =
                                  new Comparator<Contact>() {
    public int compare(Contact one, Contact other) { }
}
```

```
public static Comparator<Contact> COMPARE_BY_ZIP =
                                  new Comparator<Contact>() {
    public int compare(Contact one, Contact other) { }
}
```

4 Appendices

4.1 Definitions and Acronyms

4.1.1 Definitions

4.1.2 Acronyms and Abbreviations

GUI	Graphical User Interface
SDS	Software Design Specification
SRS	Software Requirement Specification
TSV	Tab-Separated-Values