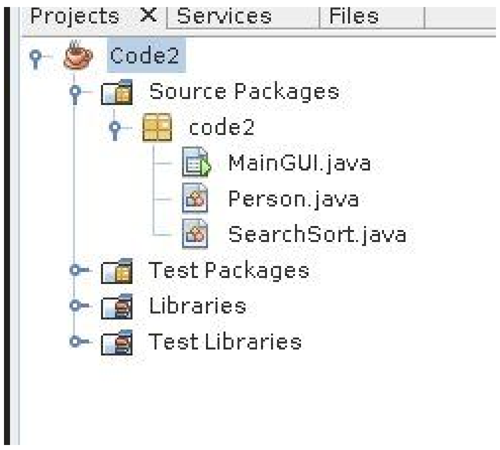
Introduction
My program is an address book with the purpose of centrally keeping track of my father's contacts. I wrote it and built the GUI in Netbeans with Java.

Summary List of Techniques

- For loop
- If statements
- Arrays
- ArrayLists
- Encapsulation
- Gui tabs
- Gui buttons
- Gui labels
- Gui combo boxes
- Gui tables
- Gui dialogue boxes
- Parameter passing
- Nested loops
- Adding to arraylists
- Deleting from arraylists
- Editing elements of arraylists
- Searching arraylists
- Printing custom log messages to the console
- Template class

Program Structure
Because my client isn't comfortable with the command line I didn't give the program a modular UI, instead I tied everything together in the MainGUI class. I created separate template classes for the object (Person) that is instantiated to form the database, and to modularize objects with larger functions like searching the database. Neither Java nor OOP are particularly well suited to my simple needs (I'd rather store my data in a .txt file and parse it with a shell/gawk script), but OOP is the focus of my IB CS class and I am more comfortable with it and Java than with shell scripting. However, the use of Java comes with the benefit of portability; a shell script written in GNU/Linux may not run well on a Mac and would certainly not work on a Windows machine.



Data Structures Used

- The main data structure I used were ArrayLists, one of which was used as the heart of the database, and others of which were used for smaller peripheral tasks of indeterminate size, like temporarily storing search results. I chose to use ArrayLists for this due to their flexibility in size when compared to Arrays. When dealing with larger datasets, lists, as fully unbalanced trees that keep each node at different places in storage or memory, are infinitely scalable and more inherently efficient in their use of storage space than Arrays.
- In a few cases where I'd be dealing with a fixed number of things, I decided to use Arrays, which store all their data in a linear block of storage or memory. Because of this, individual units of data do not need to be accompanied by memory address pointers for parent and child nodes. This allows Arrays to store more data in a given block of storage than trees, however if elements in the Array are not in use, then storage space is being wasted. Furthermore, if the Array is full (that is, making full use of the storage space allocated to it) then there is no way to add to it without making another, larger Array elsewhere in storage and copying the original data to it, resulting in a specifically sized block of unused memory addresses.

Algorithms

- The most notable algorithm I wrote was used to search and query the database. It loops through the contents of the main ArrayList and assign any instances of the Person class whose name or category attributes matches a user-defined pattern to a temporary ArrayList. The temporary ArrayList is then sent into the next algorithm in this list to be written to a table in the GUI. Pseudocode of the searching algorithm follows:

```
Method: search peopleArrayList
      Create a new ArrayList named searchResultArrayList
      Loop through peopleArrayList
            If name of current element in peopleArrayList = searchTerm then
                  Add  the current element in peopleArrayList to
                  searchResultsArrayList
      Return searchResultsArrayList
```

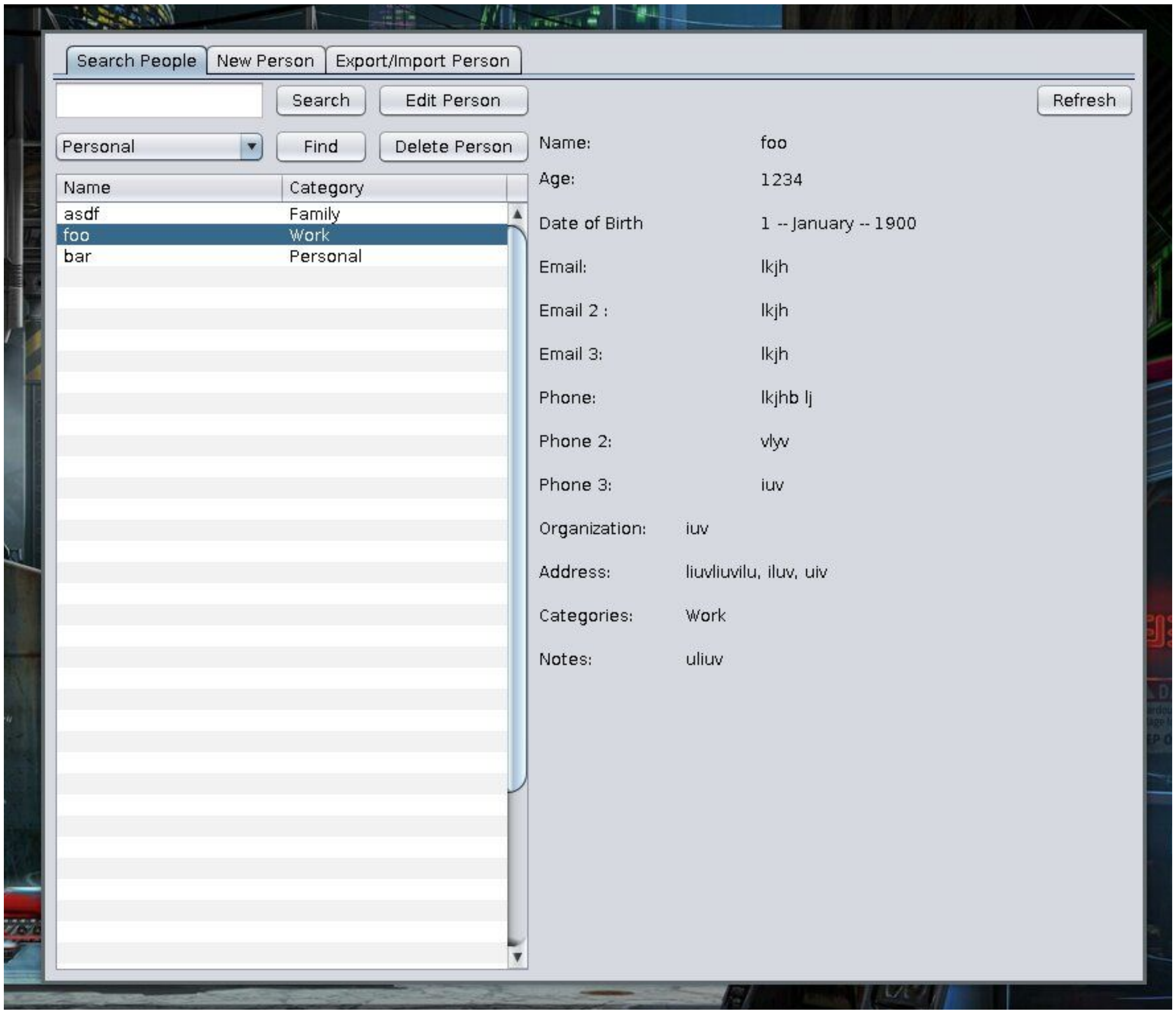
- All of the user's entries in the database are displayed in a table, the pseudocode of the algorithm used to write instances of Person to the table follows:

```
Method: writePeopleToTable
      If peopleArrayList has fewer elements that peopleTable has rows then
            Loop through peopleArrayList
```

```
                Set columns 1 and 2 in the current row in peopleTable
                equal to the name and category in the current element in
                peopleArrayList
```

User Interface

The UI for my program is a GUI made with Java Swing Components using Netbeans' drag-and-drop tool. I chose to use this tool instead of writing the Swing code myself due to its relative speed and ease of use. The final GUI:



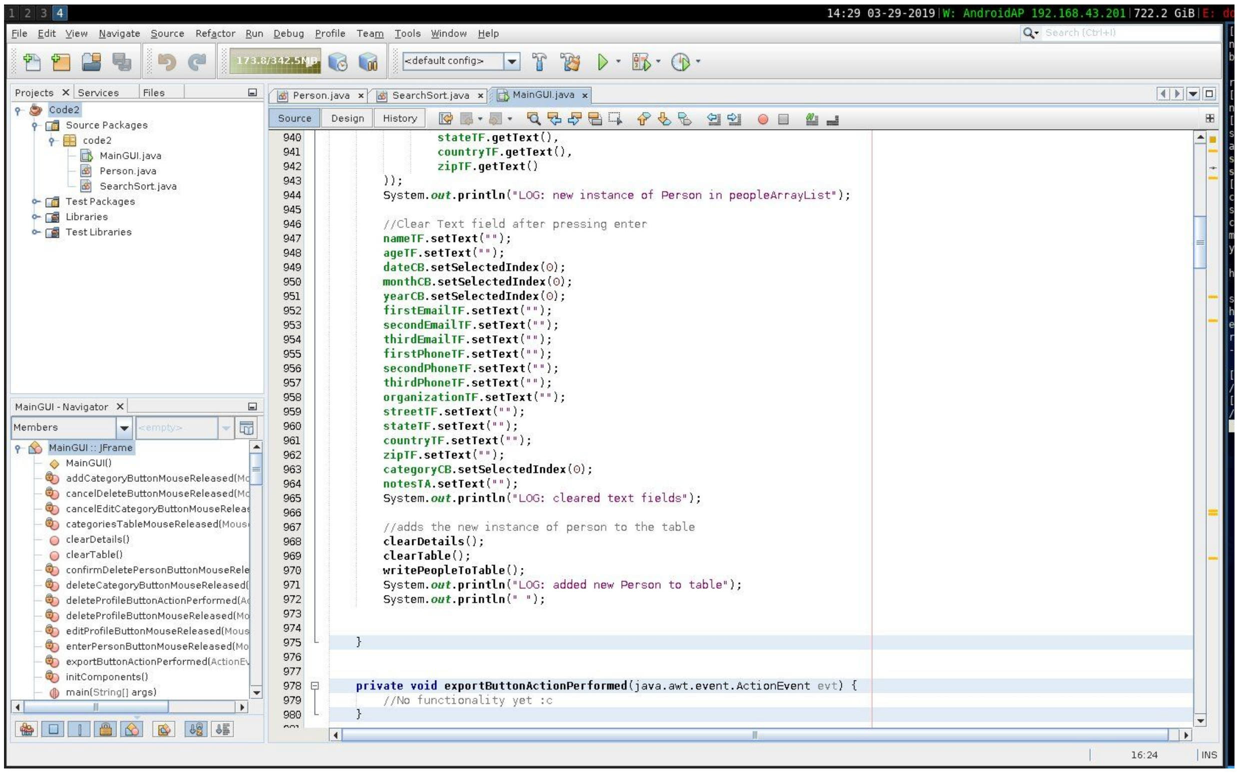
Name	Category
asdf	Family
foo	Work
bar	Personal

Name:	foo
Age:	1234
Date of Birth	1 -- January -- 1900
Email:	lkjh
Email 2 :	lkjh
Email 3:	lkjh
Phone:	lkjhb lj
Phone 2:	vlyv
Phone 3:	iuv
Organization:	iuv
Address:	liuvliuvilu, iluv, uiv
Categories:	Work
Notes:	uliuv

While text-based user interfaces are usually faster, more powerful, and more elegant than most graphical interfaces, they usually have a relatively high learning curve. My client, though moderately tech-literate, prefers GUIs over CLIs due to their lower learning curve. Furthermore, his choice of OS (MacOS) is centered around GUIs, such that using a CLI program in an elegant fashion would require extensive customization when compared to other operating systems like GNU/Linux.

Software Tools Used

For the programming stage I would have prefered to use a simple, lightweight, and comfortable text editor like Vim or GNU/Emacs, however the relative difficulty of manually writing Java Swing elements forced me to turn to the Netbeans IDE. Beyond it's GUI building tools, I only used the very basic text editing features of Netbeans, prefering to manage backups on GitHub with the command-line git client built into my OS. I also made extensive use of GNU/Emacs' Org-Mode to work on criterions A, B, and C. My coding environment:


14:29 03-29-2019 | W: AndroidAP 192.168.43.201 | 722.2 GiB
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
Projects Services Files
Code2
Source Packages
code2
MainGUI.java
Person.java
SearchSort.java
Test Packages
Libraries
Test Libraries
Person.java SearchSort.java MainGUI.java
Source Design History
MainGUI - Navigator
Members
MainGUI :: JFrame
MainGUI()
addCategoryButtonMouseReleased(
cancelDeleteButtonMouseReleased(
cancelEditCategoryButtonMouseReleas
categoriesTableMouseReleased(
clearDetails()
clearTable()
confirmDeletePersonButtonMouseRele
deleteCategoryButtonMouseReleased(
deleteProfileButtonActionPerformed(
deleteProfileButtonMouseReleased(
editProfileButtonMouseReleased(
enterPersonButtonMouseReleased(
exportButtonActionPerformed(
initComponents()
main(String[] args)
16:24 INS

```java
                stateTF.getText(),
                countryTF.getText(),
                zipTF.getText()
        ));
        System.out.println("LOG: new instance of Person in peopleArrayList");

        //Clear Text field after pressing enter
        nameTF.setText("");
        ageTF.setText("");
        dateCB.setSelectedIndex(0);
        monthCB.setSelectedIndex(0);
        yearCB.setSelectedIndex(0);
        firstEmailTF.setText("");
        secondEmailTF.setText("");
        thirdEmailTF.setText("");
        firstPhoneTF.setText("");
        secondPhoneTF.setText("");
        thirdPhoneTF.setText("");
        organizationTF.setText("");
        streetTF.setText("");
        stateTF.setText("");
        countryTF.setText("");
        zipTF.setText("");
        categoryCB.setSelectedIndex(0);
        notesTA.setText("");
        System.out.println("LOG: cleared text fields");

        //adds the new instance of person to the table
        clearDetails();
        clearTable();
        writePeopleToTable();
        System.out.println("LOG: added new Person to table");
        System.out.println(" ");


    }


    private void exportButtonActionPerformed(java.awt.event.ActionEvent evt) {
        //No functionality yet :c
    }
```

Word Count: 746