Mark Kraushar

NAIT ID: 200245877

J. Henkleman

Section 11

DMIT 104: Programming Fundamentals

Portfolio: Contact Manager

**Contact Manager: User-Guide**

This guide will cover all of the functions that my contact manager can perform. All contact data is read and written as comma separated values in “.txt” files. Note that this program utilizes 4 primary contact types: Friends, Family, Professional, and School. The program begins by presenting you with a menu with 7 numbered options. All references to phone number in this program refer to the full, 10-digit number (including area code, Example: “7809543322”). Enter one of these numbers and press “enter” to perform its corresponding task:

**1 – View all contacts:**

This will immediately present you with a second menu; there are 2 different ways to view all of your currently stored contacts:

**1 – Current list order:**

Displays your full contact list, sorted in whatever order it is currently stored. Use this whenever you want to quickly view everyone you have stored.

**2 – Grouped by contact type:**

Displays all of your contacts, they will be separated into the 4 main contact types this program uses. This option also counts how many of each contact type you have, and displays this total to separate the 4 groups. This will not alter the actual order of your list.

**2 – Sort contacts:**

This program utilizes two different ways to quickly sort your contact list. These options will change the actual order that the contacts on your list are stored, note that their list numbers change to reflect this:

**1 – First Name:**

Sort your list alphabetically, according to the contact’s first names.

**2 – Last Name:**

Sort your list alphabetically, according to the contact’s last names.

**3 – Search for a contact:**

This option presents you with a variety of search options. You can hunt down a specific contact in your list in the following ways:

**1 – By first name:**

You will be prompted to enter a first name to search for. Note that this search is not case-sensitive, and will return partial matches. For example, entering “bo” would return both “Bob Smith” and “Boris Jones” contact information.

**2 – By last name:**

The program will ask you for a last name to search for. As with the first name search, this is not case-sensitive and will return partial matches.

**3 – By contact type:**

This option allows you to isolate only one contact type to view in your list. You will be asked to choose from the friends, family, professional and school contact types, and all of the contact information for all of your contacts of this type will be displayed.

**4 – By phone number:**

Have you ever found a phone number written down, and can’t remember who it belongs to? This option allows you to enter the full, 10-digit number, and search your contacts for it. You can confirm that you already have it stored, and do not need it anymore; or that you have not yet stored it, and it may still be important. Note that this option will re-order your list according to phone number. This is so that your list can be searched more efficiently, consider visiting main menu option “2” and sorting your list by first or last name when you are finished with your phone number search.

Note: All of these search options will be followed by a menu asking if you would like to search again. Choosing yes will return you to the search type menu. This allows you to more quickly perform multiple searches until you have found what you are looking for.

**4 – Add a new contact:**

This option guides you through the process of adding another contact to your list! You will be prompted to input all of your contact’s attributes individually:

-First Name: enter the contact’s first name.\*

-Middle Name: enter the contact’s middle name.\*

-Last Name: enter the contact’s middle name.\*

-Contact Type: prompts you to enter the number that corresponds to your contact’s type.

-Phone Number: enter the contact’s 10-digit phone number.

-E-mail address: enter the contact’s e-mail address.\*

\*These fields will accept almost anything, as long as they are not empty. Don’t worry if you make a mistake, finish your contact creation and use the modify option later. You can do this as many times as you need.

**5 – Delete a contact.\***

This option displays your full contact list, in whatever order it is currently stored. From there it prompts you to “enter the contact number you would like to delete”. The contact number refers to the number in the line “Contact #1”. To delete “Contact#1” you would enter “1”.

At this point, if you have selected this option by accident, you can enter “0” to not delete a contact.

Once you have selected a contact number to delete, the single contact will be displayed, and you will be asked to confirm your deletion choice.

Regardless of what choices you have made above, your contact manager will ask you if you would like to delete another contact before returning you to the main menu.

**6 – Modify a contact.\***

This option will also display your full contact list, in whatever order it is currently stored. It also prompts you to select a “contact number” similar to the deletion process. Again, you may enter “0” if you have arrived here by accident, and do not wish to modify a contact.

Once you select a contact to modify, that contact will be displayed to you so you can easily see all of its fields. A menu will then be presented so you can select which field you wish to modify. At this point you may modify any field you choose, you may also select option “7” if you do not need to modify the selected contact.

Regardless of what choices you have made above, your contact manager will ask you if you would like to modify another contact before returning you to the main menu.

\*Note: For both the Delete and Modify tasks, you may also use the search options to hunt down your contact number. You can simply search by name or phone number, remember which contact number you wish to delete or modify, and enter it once these options prompt you. The contact manager will always confirm which contact you have selected, and give you the option to not delete or modify the contact. Just be careful of re-sorting your contact list between searches and delete / modify selections!

**7 – Save and Quit.**

This option delivers what it promises. All order changes, additions, subtractions and modifications that are currently applied to your contact list will be saved for next time! It will do its work, and then prompt you that it has saved successfully. If an error message is displayed instead, contact me, and I will return the full dollar amount that you paid for this software.

# Marking Guide

|  |  |  |  |
| --- | --- | --- | --- |
| Deliverables  (Maximum Deduction: 5) | **Total** | **Mark** | **Comments** |
| * Cover Page & Presentation * Discussion is pertinent and substantive * Test plan description * Source listing |  |  |  |
| Code Quality  (Maximum Deduction: 5) |  |  |  |
| * Comment blocks * Code comments relevant * Standards * Readability * Conformance to documentation |
| Program Performance  (Maximum Deduction: 5) |  |  |  |
| * Source files * Execution * Expected results * User Interface |
| Core Portfolio Requirements  (Weight: 20) |  |  |  |
| * Methods * Arithmetic * Decisions * Loops * 1D Arrays * Exceptions |
| Advanced Portfolio Options  (Weight: 0 – 45) |  |  |  |
| * 2D or Parallel Arrays – **5** * Lists – **5** * Bubble Sort – **7.5** * Binary Search – **7.5** * File Input – **7.5** * File Output – **7.5** * Objects – **5** * Different Language / Platform – **10** |
| **TOTAL** (Weight: 20 – 65) |  |  |  |

**Testing**

This is the contents of my contact\_list.txt file at the time that these initial test windows were generated:

Mike,Henry,Haggar,1,7804626812,mhaggar@email.com

Ken,Allen,Masters,2,7804612524,kmasters@email.com

Cody,Lee,Travers,3,7809732550,ctravers@email.com

Jennifer,Susan,Walters,4,7809889518,swalters@email.com

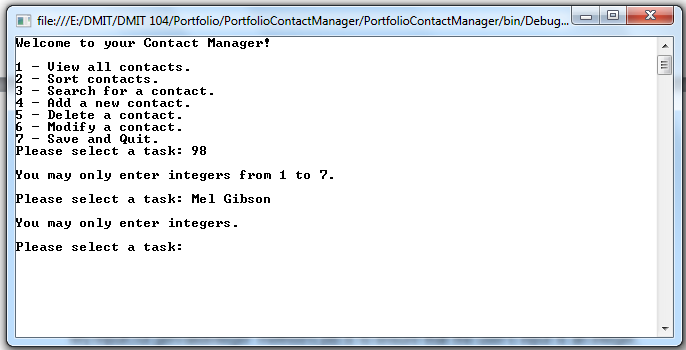
Laura,Lee,Kinney,1,7804502552,lkinney@email.com

Jill,Maria,Valentine,2,7809661812,jvalentine@email.com

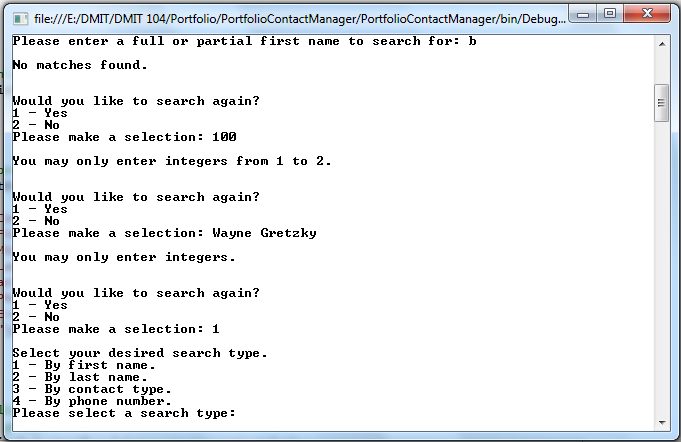
Mike,Fraud,Ross,3,7809707030,mross@email.com

Megan,Denise,Fox,4,7809945584,mfox@email.com

The testing process mostly focused on my overall logic, and then to my individual methods. I initially developed my program as a skeleton, where each menu selection would only display brief messages to confirm where the program’s logical construction was guiding its operation. The only method I needed to get up and running for this testing was my InputOutput.getValidInteger method.

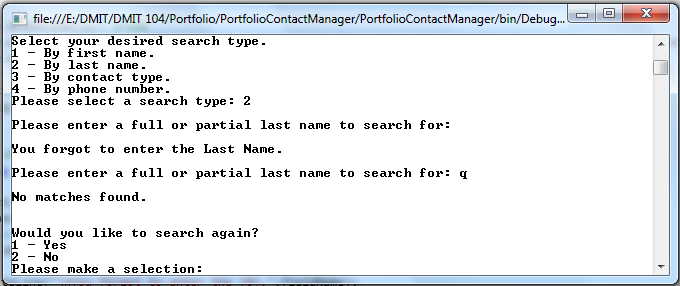
My InputOut.getValidInteger method’s job is to ensure that the user’s input is an integer, and then ensures that it falls into the acceptable integer range. It also takes in and displays the input prompt message. This method’s functionality is essential as it controls all of my menus, here is an example: 

Here you can see I have tried entering both categories of invalid input; something that is not an integer and an integer that is not in the accepted integer range. Each of these cases has its own descriptive error message, so the user knows what they have done wrong. The line, “Please select a task: “, is passed into this method, and will be looped until the user makes a valid selection. This method is utilized for every numbered menu in my program, and I have confirmed that all of them will only accept integers that correspond to valid menu choices.

Once each of these menu options performed their confirmation write line, and returned to the main menu (or exited the program in the case of option 7) I moved on to the second bit of logical construction. This was the “again” decision loops. An example of these decisions can be seen below: 

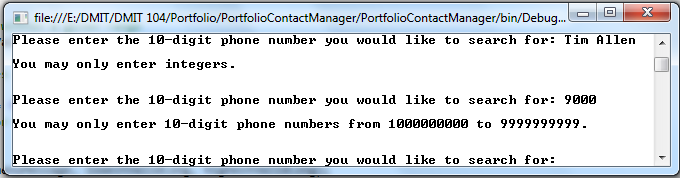
Here I have already entered “3” to search for a contact, I entered “1” to search by first name, and I entered “b” which returned no matches. You can see again, non integers and integers that are not in the accepted menu choice range, are not accepted. When “1” is entered in the search again menu, the entire search process is repeated, without returning to the main menu. I wanted these “again” options on tasks that I could see someone wanting to repeat many times at once. I wanted them to be able to do this with fewer lines in the command window. Once my main menu, all of my sub-menus, and my again decisions were structured in the way that I wanted them; I moved on to filling in the actual functionality of each choice, and building more methods.

The next method I built was InputOutput.getString. This method’s purpose is to prompt the user to enter a string (such as First name: ) and ensure that the string is not empty. It also takes in what kind of string we are trying to get from the user, so it can use this in it’s empty string error prompt. An example is given below:

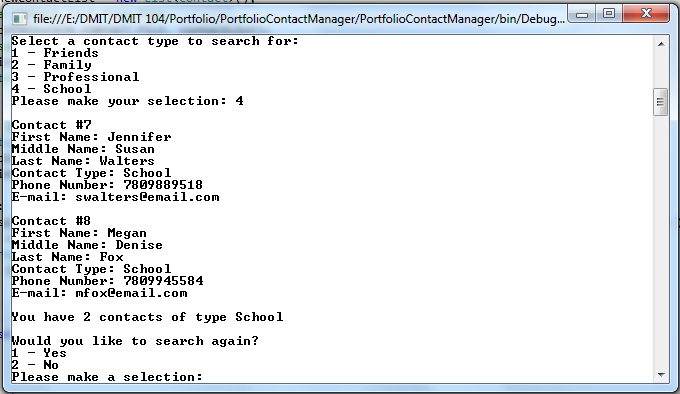


The “Please enter a full or partial last name to search for:” message string is passed into my method, and displayed until valid input is received. In the first instance of this line, I have pressed enter with no input entered. The “Last Name.” portion of the error line has also been passed into the method. Anytime the user is prompted to enter a First name, Middle name, Last name, or e-mail address, this InputOutput.getString method is utilized, and functions accordingly.

The last user-input method I needed was InputOutput.getValidLong. This method is very similar to getValidInteger, except that it uses the data-type “long” and the accepted range is always the same. This method’s purpose is to get valid, 10-digit phone numbers. It will only accept numbers between 1,000,000,000 and 9,999,999,999. An example can be seen below:



As you can see, the output is very similar to getInteger.

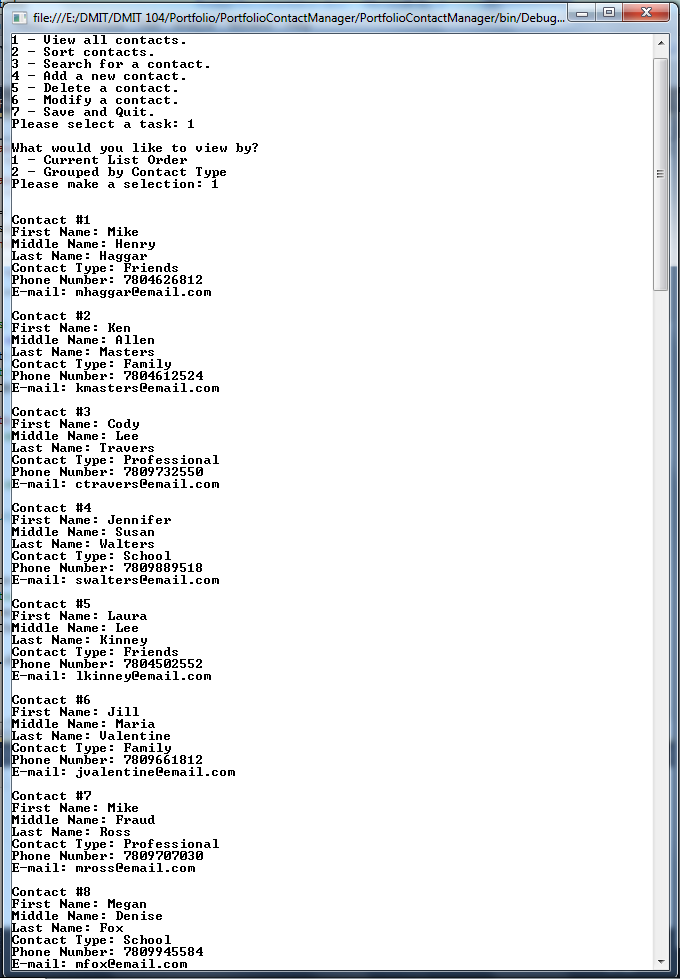
Also worth mentioning, is the way that my program uses an array to convert an integer to a contact type string. The method simply takes in the integer, from 1-4, and writes whichever contact type the number corresponds to. At this point, getInteger has already been used to ensure input to this method is an integer between 1 and 4. An example can be seen below: 

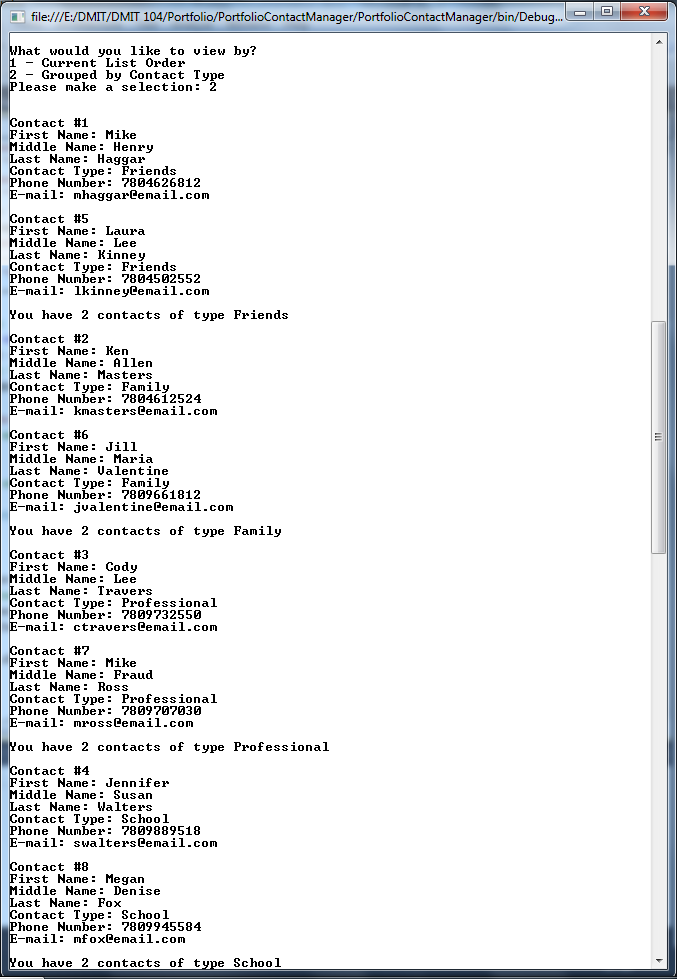
Rather than the “4” entered being used just to make a decision, it is also passed in to my InputOutput.contactTypeDecider method and used to write out the appropriate contact type from the hard-coded array of contact types. The “School”’s in the “Contact Type: School” lines, and in the “You have 2 contacts of type School” line are being written by this method here.

I will now move on to demonstrating each of the 7 main menu options. With my user input methods functioning, all that remains is to run through each modification and display option, and ensure that it is using the list correctly. Testing these options consisted mostly of picking each option, and observing the output to see how it affected my list. That the list is being read correctly can be observed in most of the screenshots in the testing section. I will leave file writing testing for last.

**1 – View all contacts:**

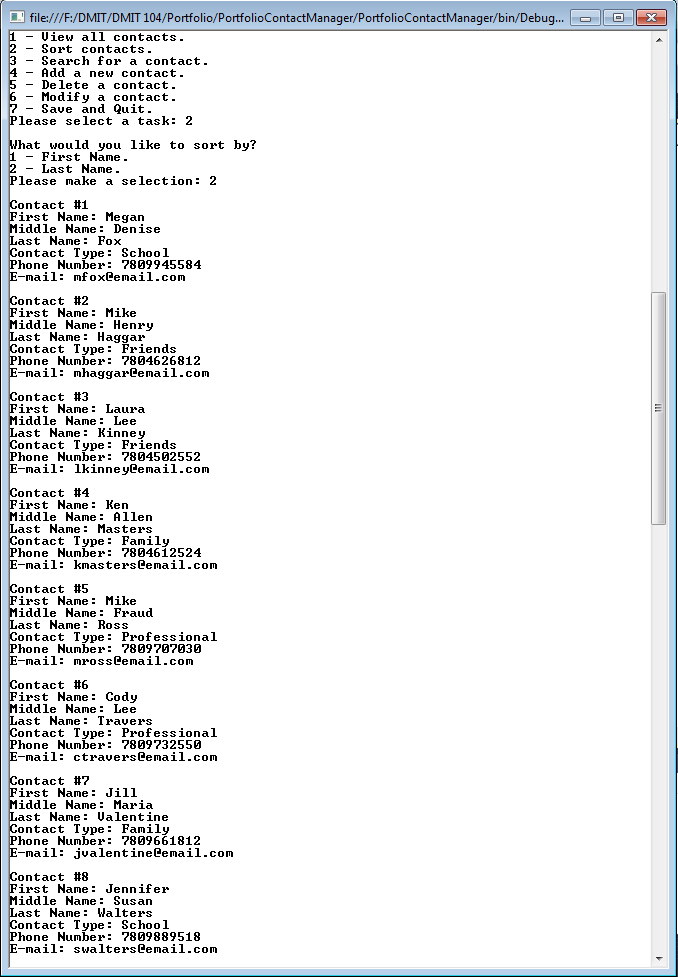
This option is fairly easy to test. It only accepts 2 integers once selected, each calling a different display method. Both of these display styles can be seen below:

****

****

Both display styles are proceeded by another display of the main menu.

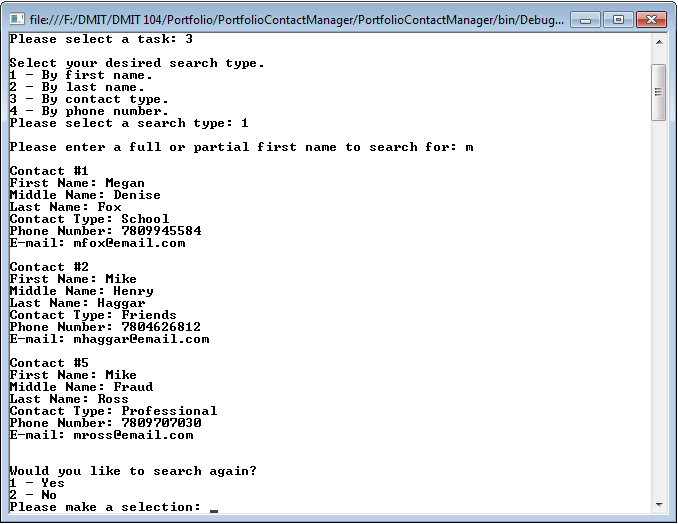
**2 – Sort Contacts:**

Both of these sorts are roughly the same, so I will only be providing an image of one. The method takes in the list, and sorts it alphabetically by either first name or last name. The first name and last name sorts are 2 separate methods. 

The list has been sorted alphabetically by last name, and is then displayed in its entirety for confirmation. The sort by first name selection operates almost identically, but sorting alphabetically by first name. Both sorts then return to the main menu and await a new selection.

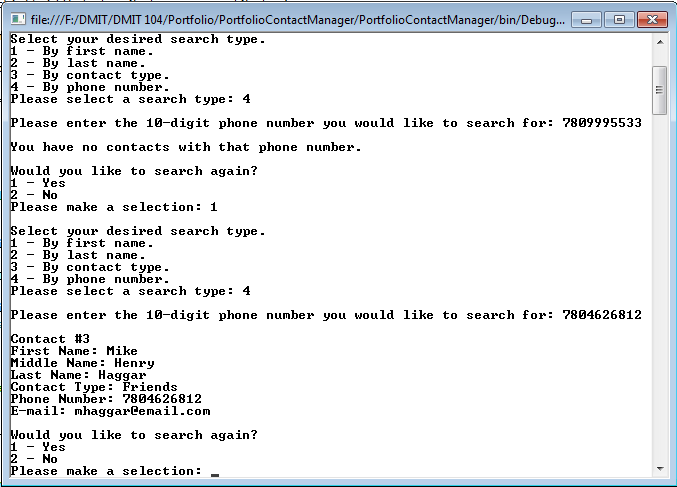
**3 – Search for a contact**

I needed to test the name searches for case sensitivity, and partial matches, and for no matches. The phone number search needed only to be tested with matches and non-matches. Testing the “By contact type.” Search consisted only of selecting the option, and ensuring it was displaying all contacts of the selected type. It functions almost the same as the “Grouped by contact type.” Display option demonstrated above. But rather than cycling through all possible contact types, it isolates and displays only the user’s desired contact type. I intended for all partial matches to be displayed in the name searches. Multiple “m” first names were included on the test list to ensure this functionality. Again the first and last name searches operate all but identically, so I will only provide an image of one:



Entering only “m” successfully displays all contacts whose first name begins with m.

Phone number search testing:



Unfound numbers inform the user, and found numbers display the contact. Note that Mike Haggar is now contact #3. The bubble sort / binary search combination that the phone number search utilized actually re-orders the list according to phone number as a numeric value. However, this produces a faster search, and the list can easily be re-sorted. This is also covered in the user-manual.

From here on the list will be utilized in its “Sorted by First Name” format. Here is what contact\_list.txt looks like sorted like this:

Cody,Lee,Travers,3,7809732550,ctravers@email.com

Jennifer,Susan,Walters,4,7809889518,swalters@email.com

Jill,Maria,Valentine,2,7809661812,jvalentine@email.com

Ken,Allen,Masters,2,7804612524,kmasters@email.com

Laura,Lee,Kinney,1,7804502552,lkinney@email.com

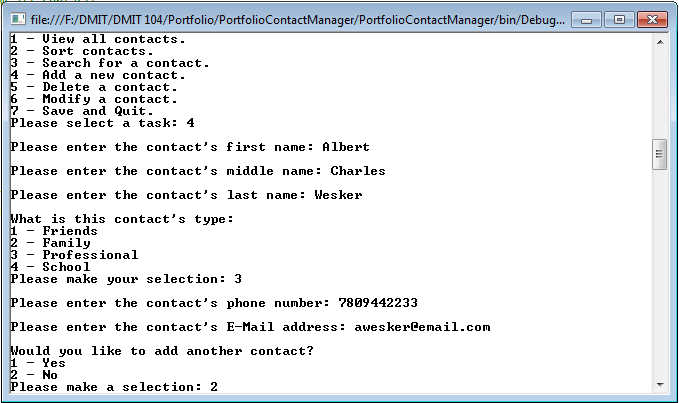
Megan,Denise,Fox,4,7809945584,mfox@email.com

Mike,Henry,Haggar,1,7804626812,mhaggar@email.com

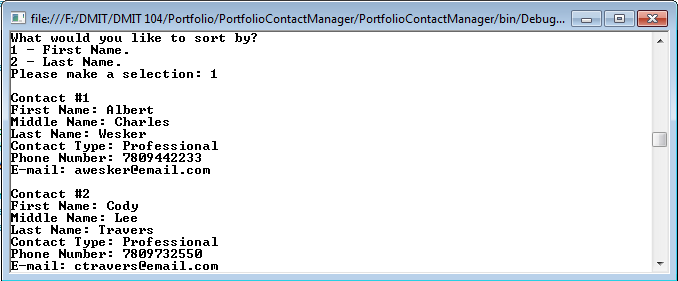
Mike,Fraud,Ross,3,7809707030,mross@email.com

**4 – Add a new contact:**

I first confirmed that all of my user input methods were being implemented correctly here, then proceeded to actually create a new contact and see it my list would store it and display it:



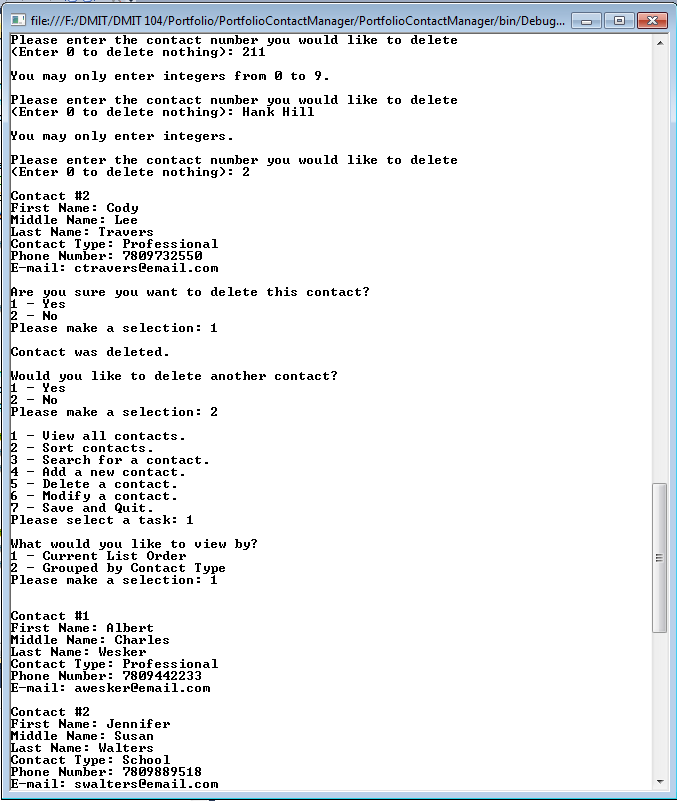
Well that doesn’t help much, let’s sort by first name again and see if the contact was actually created:



Theres a new #1 contact in town!

**5 – Delete a Contact:**

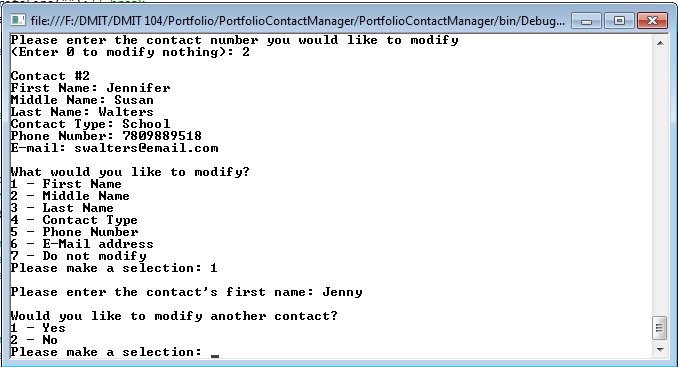
Same general testing as adding, but deleting. This was also the first method to utilize my “Contact #” line as a way of selecting a contact to operate on. Let’s just get rid of Cody altogether, Albert is so great, we don’t need him anymore:



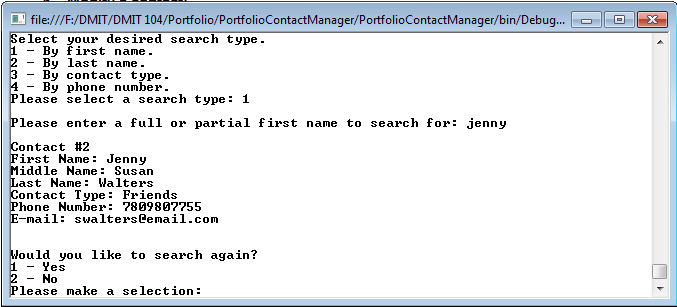
Just confirming that my InputOutput methods are still going strong. The contact is displayed along with a confirmation menu as intended. Program also confirms to the user after that the contact has been deleted. And we can see that he has in fact been removed.

**6 – Modify a contact:**

I tried modifying every property of multiple contacts, and ensuring that this was reflected in my sorts and searches afterwards. For this demonstration screenshot I will alter only one string property (First name), the phone number and contact type for contact #2.

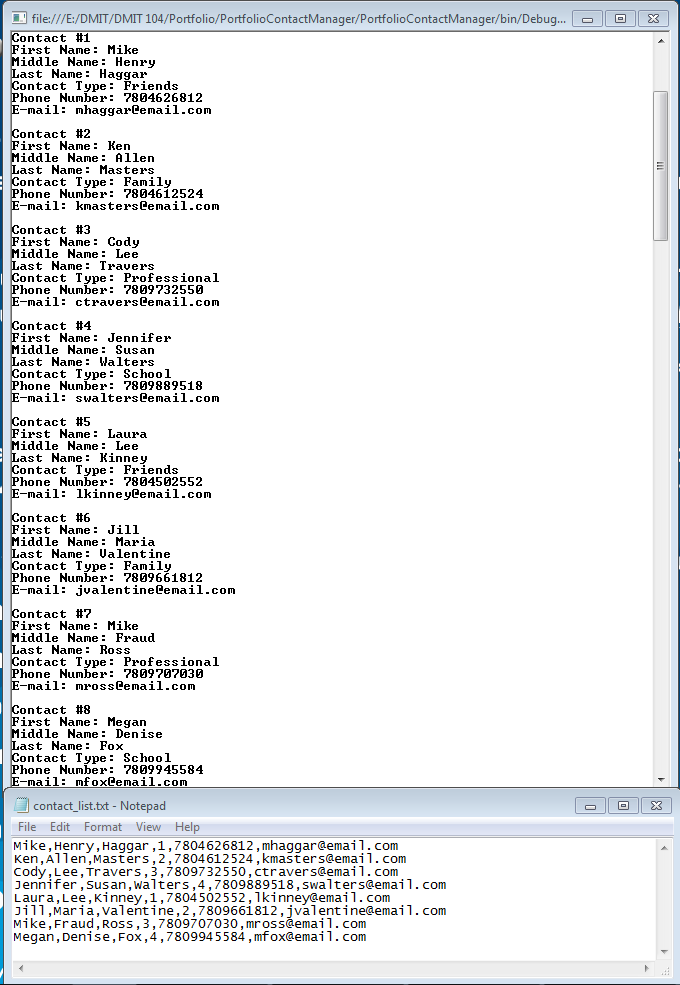


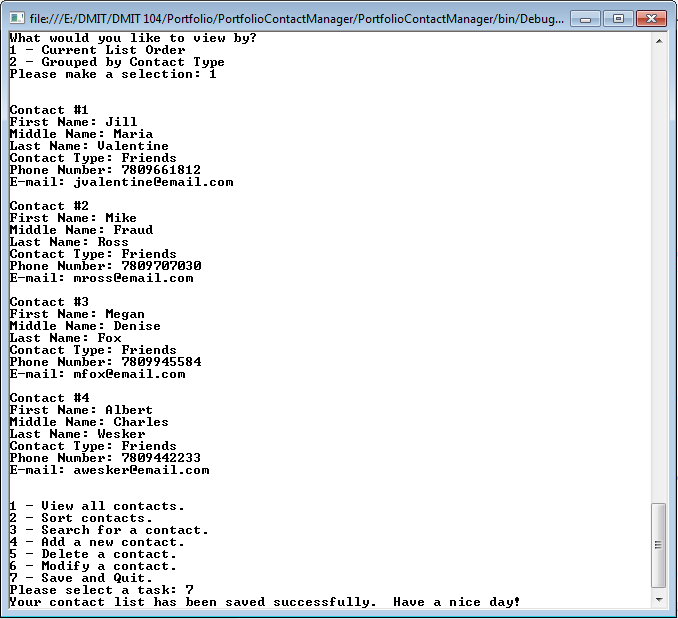
Repeat this process for Contact Type: “1” and Phone Number “7809807755”.

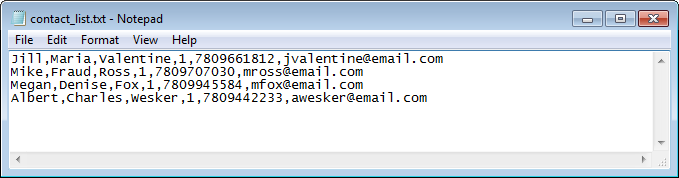


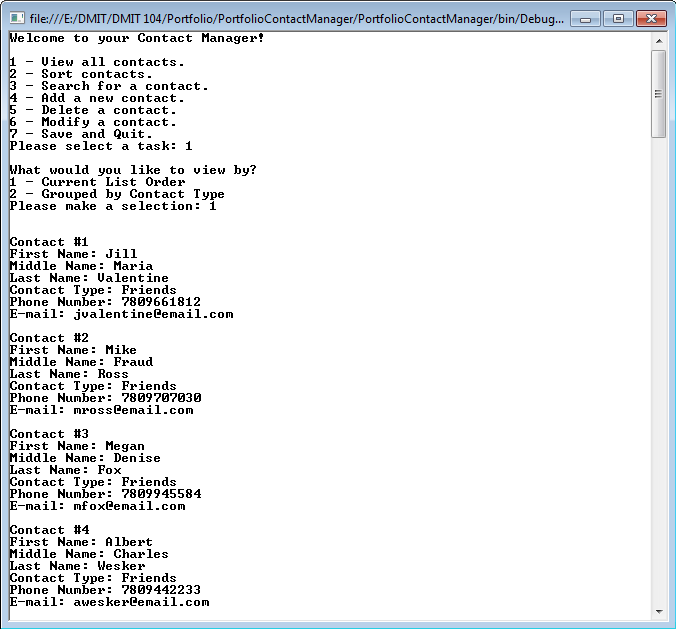
Now that we’re friends she told me her real phone number, and it turns out she prefers to go by Jenny!

**7 – Save and Quit.**

****This part was giving me the most trouble. Mostly because I was testing it incorrectly. I have now confirmed that the file writing works both by locating the actual altered “contact\_list.txt” and by re-reading that file and displaying it in my program to confirm that it is retaining it’s changes. At this point I have returned to my original contact\_list.txt from the beginning of the testing section. Here is a screenshot of the list in my program and in my .txt side by side for reference:

Now I will try entering “1” into my deletion menu 5 times. This should delete everyone but Jill, Mike (Ross) and Megan. I will add Albert Wesker back to the list, and make everyone remaining into a friend:

This screen shows the state of my list upon selecting “Save and Quit”. The program then exits when you press any key. Time for the moment of truth, locate contact\_list.txt and restart the program to see if the changes persist:



Note that the “1” in the .txt file after the contacs last name indicates the contact type. “Friends” being contact type “1” in my contact type array. The program appears to be successfully writnig to contact\_list.txt, and is able to refer back to the format it writes in for re-populating the program’s list upon starting the program again. This concludes my testing. My input methods appear to work as intended, and my logic appears to guide the user through my program on the path that I intended. The touchiest part of this program was getting the file to write correctly. This also appears to be working now.