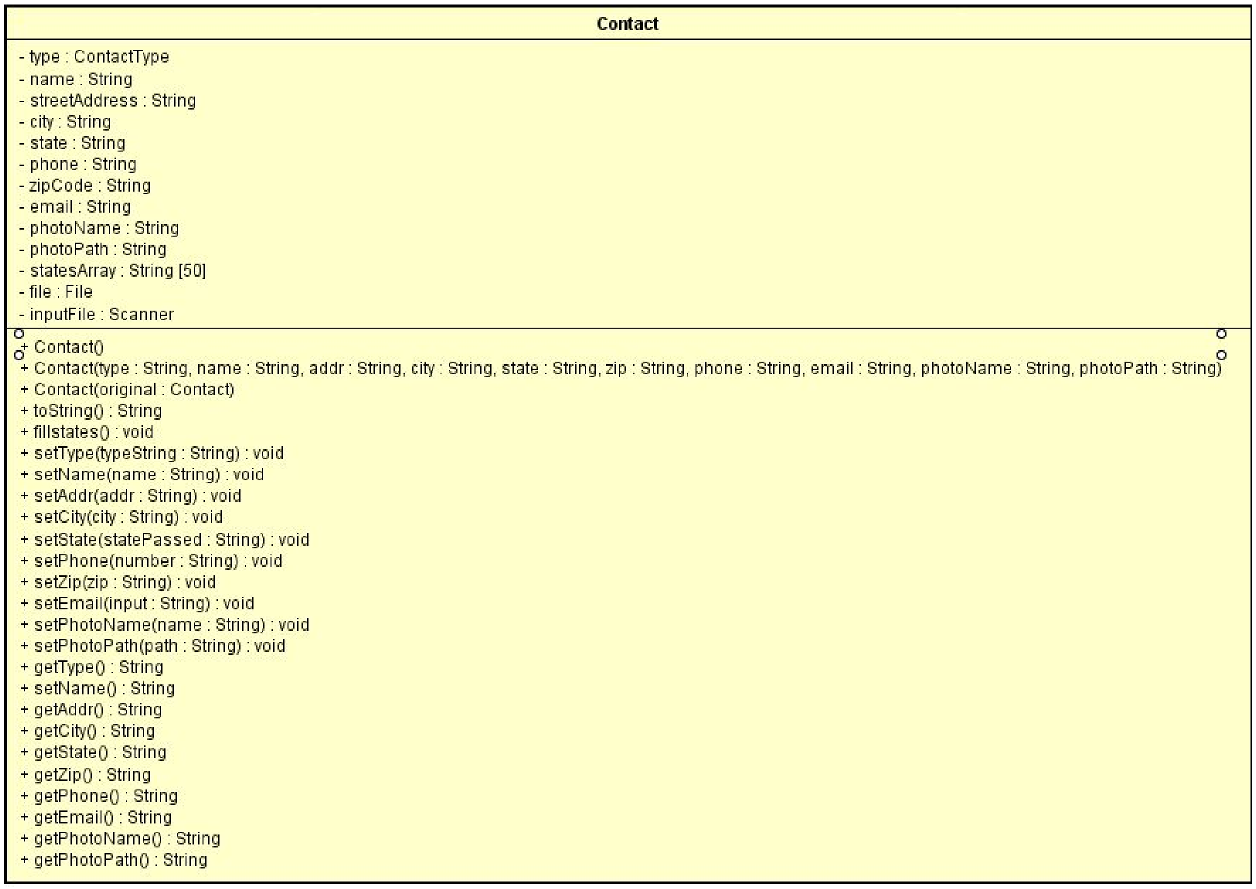
Ryan Shupe
CSCI-1260-002
08/4/18

<u>**Contact Class**</u>

Contact class is a class that is going to be used to store attributes of a contact.



```
Contact
- type : ContactType
- name : String
- streetAddress : String
- city : String
- state : String
- phone : String
- zipCode : String
- email : String
- photoName : String
- photoPath : String
- statesArray : String [50]
- file : File
- inputFile : Scanner
+ Contact()
+ Contact(type : String, name : String, addr : String, city : String, state : String, zip : String, phone : String, email : String, photoName : String, photoPath : String)
+ Contact(original : Contact)
+ toString() : String
+ fillstates() : void
+ setType(typeString : String) : void
+ setName(name : String) : void
+ setAddr(addr : String) : void
+ setCity(city : String) : void
+ setState(statePassed : String) : void
+ setPhone(number : String) : void
+ setZip(zip : String) : void
+ setEmail(input : String) : void
+ setPhotoName(name : String) : void
+ setPhotoPath(path : String) : void
+ getType() : String
+ setName() : String
+ getAddr() : String
+ getCity() : String
+ getState() : String
+ getZip() : String
+ getPhone() : String
+ getEmail() : String
+ getPhotoName() : String
+ getPhotoPath() : String
```

+Contact()

PROCESSING:

Initialize the variables in contact to default values using setters
Ex: setName(name);

+ Contact(type : String, name : String, addr : String, city : String, state : String, zip : String, phone : String, email : String, photoName : String, photoPath : String)

INPUTS:

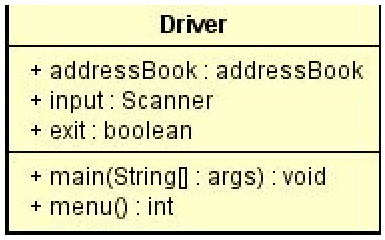
Name
Addr
City

```
        State
        Zip
        Phone
        Email
        photoName
        photoPath
(Note: the strings passed in can be any size. The setters that we call is going to format it correctly and make sure everything is a valid input)

    PROCESSING:
        Initialize the variables using the setters using the parameters of the constructor
        Ex: setType(type);

+ Card(original : Contact)
    PROCESSING:
        Initialize the variables using the Contact passed in. (Copy Constructor)
        Ex: setAddr(original.getAddr());

+ toString() : String
    PROCESSING:
        Get the values of all the variables using the getters and put into a neatly
    formatted String
        Return the String

+ fillstates() : void
    PROCESSING:
        Try to open the file ‘states.txt’
            Initialize file and Scanner variable
            While the file has a next line fill up the array called States
            Close the file
        If an Exception is thrown
            Display that the file could not be opened
```

Documentation for the setters and getters is not required according to the project pdf

Driver Class:

Driver will host the many options you can do with the AddressBook, add contacts remove, view, etc..



```
+ main(String[] : args) : void
PROCESSING:
    Display welcome message
    Ask for name and put into appropriate var
    Call menu
    Get number the user enters
        If it is 1:
            Call fill addressBook from addressbook class
        If it is 2:
            Call add a contact from addressBook class
        If it is 3:
            Call edit a contact from the addressBook class
        If it is 4:
            Call remove a contact from the addressBook class
        If it is 5:
            Call the tostring method from the addressBook class
        If it is 6:
            Try to call find a contact from the addressBook class
            If exception is thrown then display error message
        If it is 7
            Try to ask for the type
            Read in
            Call search for type from the addressBook class
            If exception is thrown then display appropriate error message
        If it is 8
            Try to ask for the zip
            Read in
            Call search for zip from the addressBook class
            If exception is thrown then display appropriate error message
        If it is 9:
            Call sort by name from the addressBook class
```

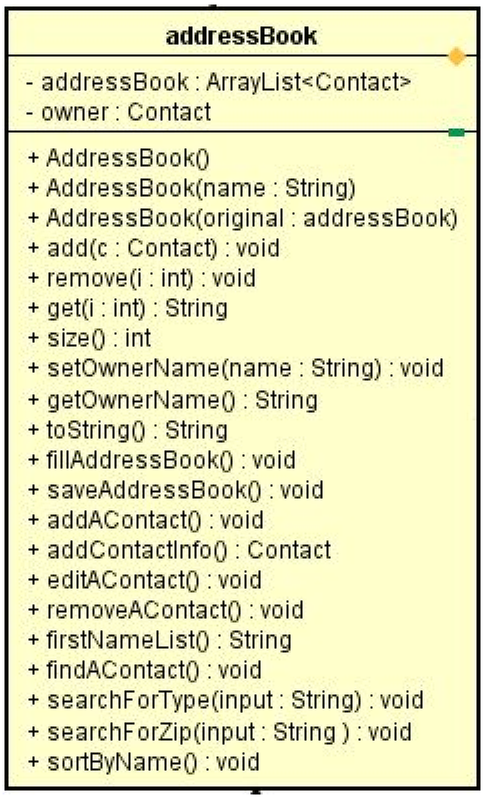
```
                If it is 10:
                        End the program
        If it is anything other than 1-10 then repeat until the number is valid.
```

+ menu() : int
PROCESSING:

```
        Display a well organized String and read in an integer input
        Return the int
```

<u>AddressBook class</u>

Store many different contact objects into an arrayList, also contain methods that will help add, delete, edit, sort, etc.



```
addressBook
- addressBook : ArrayList<Contact>
- owner : Contact
+ AddressBook()
+ AddressBook(name : String)
+ AddressBook(original : addressBook)
+ add(c : Contact) : void
+ remove(i : int) : void
+ get(i : int) : String
+ size() : int
+ setOwnerName(name : String) : void
+ getOwnerName() : String
+ toString() : String
+ fillAddressBook() : void
+ saveAddressBook() : void
+ addAContact() : void
+ addContactInfo() : Contact
+ editAContact() : void
+ removeAContact() : void
+ firstNameList() : String
+ findAContact() : void
+ searchForType(input : String) : void
+ searchForZip(input : String ) : void
+ sortByName() : void
```

+ AddressBook()
PROCESSING:

No arg constructor will set the name of the owner to a default value by calling set name

+ AddressBook(name : String)
PROCESSING:

Take the string passed in and then set the name of the owner to that string by calling set name

+ AddressBook(original : addressBook)
PROCESSING:
```
    Set the owner name to the owner of the original addressBook
    Create a for loop to fill up the current arraylist using the contacts from the original
```

+ add(c : Contact) : void
PROCESSING:
```
    Get the contact passed in and call the add method in arraylist to add to the arraylist
```

+ remove(i : int) : void
PROCESSING:
```
    Get the number passed in and call remove from arraylist to remove the contact in that
position
```

+ get(i : int) : String
PROCESSING:
```
    Get the number passed in and call the get method from arraylist to get the contact object
    Then call the objects tostring
    Return the string
```

+ size() : int
PROCESSING:
```
    Return the size of the arraylist using the size method from arraylist class
```

+ setOwnerName(name : String) : void
PROCESSING:
```
    Get the string passed in and then set the owner name to that string.
```

+ getOwnerName() : String
PROCESSING:
```
    Return the name of the owner
```

+ toString() : String
PROCESSING:
```
    Use a for loop that calls the get method
    Use the tostring returned from the get method to create a string that has all the contacts
in it
    Format it neatly
    Return the string
```

+ fillAddressBook() : void
PROCESSING:
```
    Use JFileChooser to get a file.
```

Open the file
Read from the file and split into a string array
Use string array to add a contact to fill the addressBook using the add method (using the contact's param constructor)
Close the file.

+ saveAddressBook() : void
PROCESSING:
```
    Use JFileChooser to get the file
    Write to the file in the proper format
    Close the file.
```

+ addAContact() : void
PROCESSING
```
    Call the addcontactinfo method
    Display message
```

+ addContactInfo() : Contact
PROCESSING:
```
    Go through each of the steps to add a contact calling the setters from contact
    Add the contact using the add method
```

+ editAContact() : void
PROCESSING:
```
    Get the name of the contact they wish to edit
    Find the contact
    Call editContactInfo
```

+ removeAContact() : void
PROCESSING
```
    Get the name of the contact they with to remove
    Find and remove the contact
```

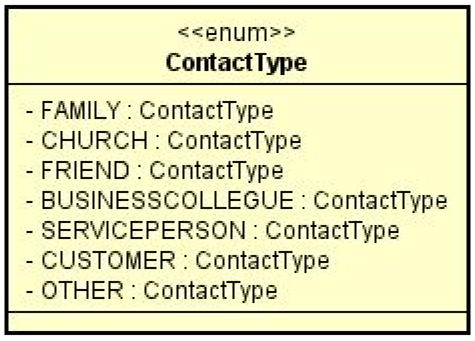
+ firstNameList() : String
PROCESSING:
```
    Gets the first name of all the contacts in the addressBook
    Put the names in a formatted String
    Return the string
```

+ findAContact() : void
PROCESSING:
```
    Get the contact they wish to find
    Display the toString from that contact
```

```
+ searchForType(input : String) : void
PROCESSING:
    Get the string sent in and see if it is a valid type
    Search for the type
    If the type is found anywhere
        Display the tostring for the contact

+ searchForZip(input : String ) : void
PROCESSING:
    Get the string sent in and see if it is a valid zip
    Search for the zip
    If the zip is found anywhere
        Display the tostring for the contact

+ sortByName() : void
PROCESSING:
    Call collections.sort
    Pass it two contacts
    Compare names to see if it needs to be swapped
```

<u>ContactType enum:</u>

Create the type Contact type and store possible values for it



- FAMILY : ContactType
- CHURCH : ContactType
- FRIEND : ContactType
- BUSINESSCOLLEGUE : ContactType
- SERVICEPERSON : ContactType
- CUSTOMER : ContactType
- OTHER : ContactType