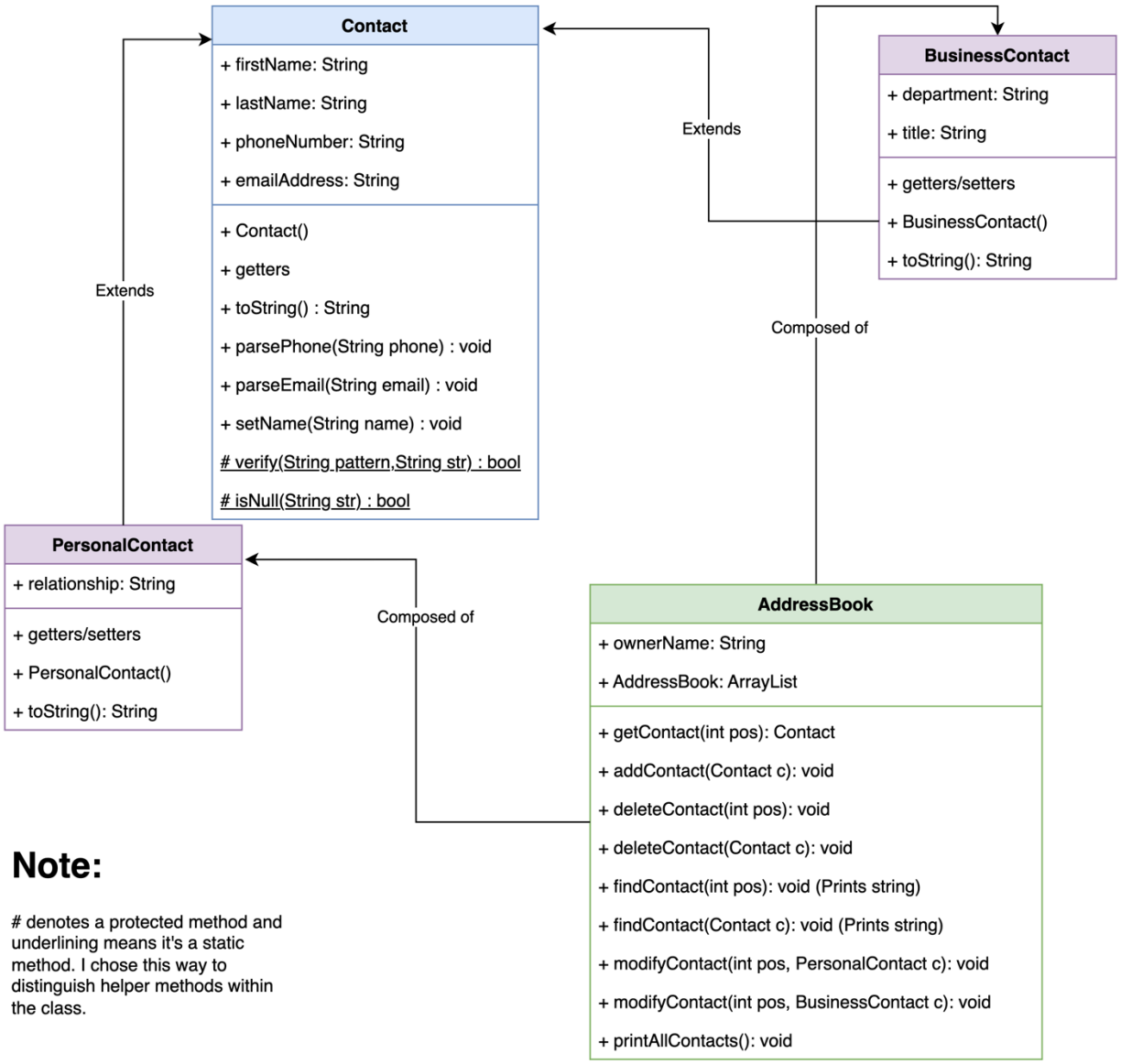
Project 2 plan

This is an implementation for a contact management system. The user will be allowed to add, delete, find, modify, and show all their contacts. Special setters with their own validation conditions are required, especially for the phone and email attributes to ensure data consistency. Each class uses the string “NA” as a default value for all attributes to help with data validation.



Pseudocode for the UI:

```
def showUserOptions():

    String menu = ("\nMENU\n"
        + "1 – Add contact to address book\n"
```

```
        + "2 – Remove contact\n"
        + "3 – Find contact\n"
        + "4 – Modify Contact\n"
        + "5 – Output all contact information\n"
        + "6 – Quit")

    Println("\n"+menu)
def main():

    boolean done = false

    String ownerName  = input("Enter your first name: ")
    AddressBook addressBook = new AddressBook(ownerName)

    showUserOptions()
    while not done:
      try:
          choice  = input("\nChoose an option: ")

           switch (choice) :
           case ('1'):

                       contactType = input(“Enter contact type: p for personal, b for business”)
                       name = input(“Enter the name of the contact. n/na to pass”)
                       phone = input(“Enter 10 digit phone number. n/na to pass”)
                       email = input(“Enter the email address. n/na to pass”)

                       switch contactType:
                            case “p”:
                                 Personalcontact p =  new Contact()
                                 p.setName(name)
                                 p.parsePhoneNumber(phone)
                                 p.parseEmail(email)

                                 relationship = input(“Who is this contact to you? (friend, best
      friend, family, etc.)”)
                                 p.setRelationship(relationship)

                                 addressBook.addContact(p)


                                 break

                            case “b”:
                                 Businesscontact b =  new Contact()
                                 b.setName(name)
                                 p.parsePhoneNumber(phone)
                                 p.parseEmail(email)
```

```
                                        department = input(“What department do they work at?”)
                                        title = input(“What is this contact’s job title?”)

                                        addressBook.addContact(b)
                                        break
                                default:
                                        Println(“Invalid entry. Please start again.”)
                                        break


                        showUserOptions()
                        break

        case ('2'):
                        // I may add a way to delete by information lookup but for now, this is how I would do it
by index

                Println("\nHere is your current list of contacts:\n")
                addressBook.printAllContacts()
                position = input("\n\nEnter the location number of the contact you wish to delete as it
appears above! “)


            addressBook.removeContact(pos)
            Println("\nNew list: \n")
            addressBook.printAllContacts()

            showUserOptions()
            break

        case ('3'):

        // This part would be similar to the delete cae just without the action of removal. Again, I may
        implement a way to search by info lookup but I haven’t decided on the criteria for it yet.

            position = input("\n\nEnter the location number of the contact you wish to find“)
            addressBook.findContact(pos)
            showUserOptions()
            break

        case(‘4’):

        /* Modification is tricky to implement because first, you would need to find the contact, return
        the contact back to determine its type, prompt the user for the necessary attributes, create a
        similar contact of that type, and then modify the found contact based on the modifier contact’s
        attributes. However, because the list contains the parent objects, the found contact also would
        need to be cast as a child object within the class modifying methods so that the setter methods
```

could be extended. I plan to implement this only by index to save time. But here is what it could look like: */

```
boolean done = false
      Println("\nHere is the list of your address book contacts:\n")
      addressBook.printAllContacts()

      int pos = input("\n\nEnter the location number of the contact you wish to modify as it appears above! ") AS int


      while  not done:

         try:

            /*
            The getClass method returns the runtime object, and the getSimpleName method returns the name of that object's type as a string.
            These are built-in methods for every class just like the toString() method.
            https://www.scaler.com/topics/getclass-in-java/
            https://www.geeksforgeeks.org/class-getsimplename-method-in-java-with-examples/
            */

            switch (addressBook.getContactInfo(pos).getClass().getSimpleName()) //
               case "BusinessContact":

                  String name = input("\nEnter the modified contact's name. Enter n or na to pass. ")
                  String phoneNumber = input("Enter the modified contact's phone number in XXX XXX XXXX format. Enter n or na to pass ")

                  String email = input("Enter the modified contact's email. Enter n or na to pass ")
                  String dept = input("Enter the contact's department. Enter n or na to pass ")
                  String title = input("Enter the contact's job title. Enter n or na to pass ")


                  BusinessContact bMod = new BusinessContact()
                  bMod.setName(name)
                  bMod.parsePhoneNumber(phoneNumber)
                  bMod.parseEmail(email)
                  bMod.setDepartment(dept)
                  bMod.setTitle(title)

                  addressBook.modifyContact(pos, bMod)
                  done = true
                  break

               case "PersonalContact":
```

```
                String name = input("\nEnter the modified contact's name. Enter n or na to pass. ")
                String phoneNumber = input("Enter the modified contact's phone number in XXX
XXX XXXX format. Enter n or na to pass ")

                String email = input("Enter the modified contact's email. Enter n or na to pass ")
                String dept = input("Enter the contact's department. Enter n or na to pass ")
                String title = input("Enter the contact's job title. Enter n or na to pass ")


                String relationship = input("What is the contact's relationship to you? Enter n or na
to pass ")


                PersonalContact p = new PersonalContact()
                p.setName(name)
                p.parsePhoneNumber(phoneNumber)
                p.parseEmail(email)
                p.setRelationship(relationship)


                addressBook.modifyContact(pos, p)
                done = true
                break

            Println("\nNew Info:")
            Println(addressBook.getContactInfo(pos))


        catch (NullPointerException noneType)

            done = true

showUserOptions()
break



 case ('5'):
    addressBook.printAllContacts() // Part of the AddressBook class
    showUserOptions()
    break


 case ('6'):
    Println()
    done = true
    break
```

```
 default:
    Println("Invalid entry. Please try again")
    break

catch (StringOutofBounds) :
        Print(“Blank entry. Please start again”)
        showUserOptions()
```