Software Development

Mini Assignment 2
Due: September 21, 2018

This mini assignment explores the software techniques covered in lectures #2 through #4.

The goal is to evaluate your decision making as you attempt to construct an optimal solution (lecture 2) that is well designed from an object-oriented perspective (lectures 3 & 4).

The problem

Create a well-designed contacts application (a telephone book).

Write a **ContactsMain.java** file that will contain the main() method and all the UI for the program. Keep all the I/O within this file. You are permitted to write supporting methods.

Write a **ContactsTest.java** file to test your contacts. It does not test ConstactsMain.java. It will have a main() method that will run the testing software. You are permitted to write supporting methods.

The file structure for the **Contacts** application, other than ContactsMain.java and ContactsTest.java, is up to you but must be well-designed. Use the word "Contacts" in the class names of all additional files. Your solution must follow the instructions provided here.

Problem Description

The main() method in ContactsMain.java displays the following text menu:

1. New Contact
2. Find Contact
3. List All
4. Quit

The user stays in the menu loop until they press quit. Then the program terminates. Each menu option's I/O is handled within ContactsMain.java. You can create supporting methods. Only ContactsMain.java can have I/O, like Scanner.

New Contact:

> Prompts the user to create one of three types of contacts: Acquaintance, Business, or Friend. Each time the user presses New Contact they can create the same or different contact type. The information gathered for Acquaintance is Name and Phone number. The information gathered for Business is: Name, Phone number, Address, and Business name. The information gathered for Friend is: Name, Phone number, Address, and Birthday. These are all Strings. You do not need to validate what they input.

Find Contact:

> Prompts the user for a String. It then searches the data structure to find the contact. The search string is used to find the first contact whose personal name matches the search string. If the

person does not exist, then the program displays “Not found”. If that person does exist, then all the information for that person is displayed on the screen, for example:

For Acquaintance	For Business	For Friend
Name	Name	Name
Phone	Phone	Phone
	Address	Address
	Business Name	Birthdate

List All:

Displays all the contact information of all the users in a table-like format:

TYPE NAME PHONE ADDRESS BIRTHDATE

Leave unpopulated columns blank.

Quite:

The user exists the menu loop. The program ends.

Questions

Do not write answers to these questions in English. Your answer to these questions will be expressed in the decisions you make when writing your solution code. Think about the following questions and how they might impact your program.

The best code is optimal sharable code. I/O forces only one way to use code, which makes it hard to share since only those people who want to use the program exactly as the I/O presents the application will want to use it. How can you maximize the “share-ability” of your solution?

Does your solution need an Interface?

Does your solution need an Abstract class?

What would be the optimal data structure?

Do we need any optimal algorithms?

What OO techniques should be used for the three contact types?

Identify the parts of the program that need to be well-designed.

Since ContactsTest.java does not test ContactsMain.java, what will it test?

Make sure you protect your objects.

You are creating a well-designed application. I expect to see optimality, pretty code, and comments as documentation.

WHAT TO HAND IN

- ContactsMain.java and ContactsTest.java
- A minimal number of other .java files belonging to the application using the word “Contacts” in their name to implement the contacts (phone book).
- Zip everything into a single file

HOW IT WILL BE GRADED

For your assignment to be graded your program (a) must run, (b) did not use tools, and (c) followed the assignment instructions. You are doing this assignment on your own.

- +5 – Optimality (Big Oh and memory)
- +5 – Simplicity of solution algorithms
- +5 – Correctness of the application
- +5 – Uses encapsulation, information hiding, generalized code, code reuse, code similarity, APIs
- +5 – Well written code that is easy to read and commented well (see lecture 2 for examples)
- +5 – The implementation of **Contacts** (OO techniques)
- +5 – The implementation of **Contacts** (optimal algorithms)
- +5 – The implementation of **Contacts** (data structures)