COMP1161 PROJECT 2 & 3 (SUMMER 2023)

Due: Saturday July 15, 2023 @ 11:55pm

This project can be done in group of 3-4 persons.

THE PROBLEM

In Part 1 of the project, you implemented some classes that store data about a contact in an address book. Part 2 of the project will require that you use those classes along with some controller classes to implement the address application in conformance with the specification of functionality and the design requirements stated herein in a Text User Interface. Part 3 will require you to use those classes along with some controller classes to implement the address application in conformance with the specification of functionality and the design requirements stated herein in a Graphical User Interface.

A. Specification of Functionality

I. The user the application shall read contact data from a file (if one exists) and store the data in a in internal memory structure such as a list, array, or any other appropriate structure.

II. The application should allow the user to perform the following operations:

- Create a new contact record by inputting name, gender, alias, address and date of birth of the contact.
- Search for a contact by entry # and, if found, display the full details of the contact.
- Search for contact by email address and, if found, display full details of the contact.
- Modify any details of the contact:
  - Last name
  - Alias (new alias should be unique)
  - Address
  - Add a telephone number (no duplicate allowed)
  - Delete a specified telephone for a contact.
  - Add an email address for a contact (no duplicate allowed)
  - Delete a specified email address for a contact.
- Display the list of contacts in the address book (entry#, name, gender, email address only) sorted by entry #.
- Display the list of contacts in the address book (entry#, name, gender, email address only) sorted by last name, then by first name.
- Delete a contact from the address book given by the entry # of the contact.
- Delete a contact from the address book given an email address.

On closing, the application shall prompt the user on whether to save the address book data to a file. If the user responds affirmatively then the application shall write address data to a text file.

B. Design Requirements

You can design in any way you wish subject to the following:

1. Your design shall include a class (or set of classes) that provide(s) a textual user interface (whether menu driven, or command line) to the application.
2. Your design shall include a class (or set of classes) providing a graphical user interface of the application.
3. Your design shall include a class, or set of classes, that control(s) the application by providing methods that a user interface object can invoke in order to perform operations such as creating a new contact, retrieving the data for a specified contact, updating the contact data and deleting contact (think of CRUD-Create Read Update Delete).
4. Your design shall include a class that knows how to write data to a file in a format of your choosing, and how to read the data from the file.

C. OOP Principles

Your code will be marked for observation of proper OOP principles.

1. Separation of concerns – each class in the application should have specific responsibilities. For example, all displaying of information to the screen, or inputting of values should be done by the user interface classes. Similarly, the user interface objects should not be responsible for application logic such as direct communication with the memory structures that contain the contact objects, or direct interaction with the data manager object.
2. Support methods are used sensibly to reduce the complexity of large methods.
3. Appropriate use of static variables and methods.
4. Use of the Comparable interface.
5. Use of a Comparator interface.
6. Appropriate use of exception handling.
7. Use of the sort method of the Collections class, or the Arrays class.
8. Use of the binarySearch method of the Collections class, or the Arrays class.
9. Student can highlight where polymorphism is used in the application.
10. Observation of rules for identifiers (camel case for variables, etc.)
11. Proper use of visibility modifiers.
12. Consistent indentation of code.
13. Documentation of code using javadoc tags.

Graphical User Interface Appearance

You can make your GUI in any appearance that you want as long as you are aware of the layout manager used and it carries out the instructions like that asked from the Textual User Interface. Here is an example of one below which I have used a BorderLayout:

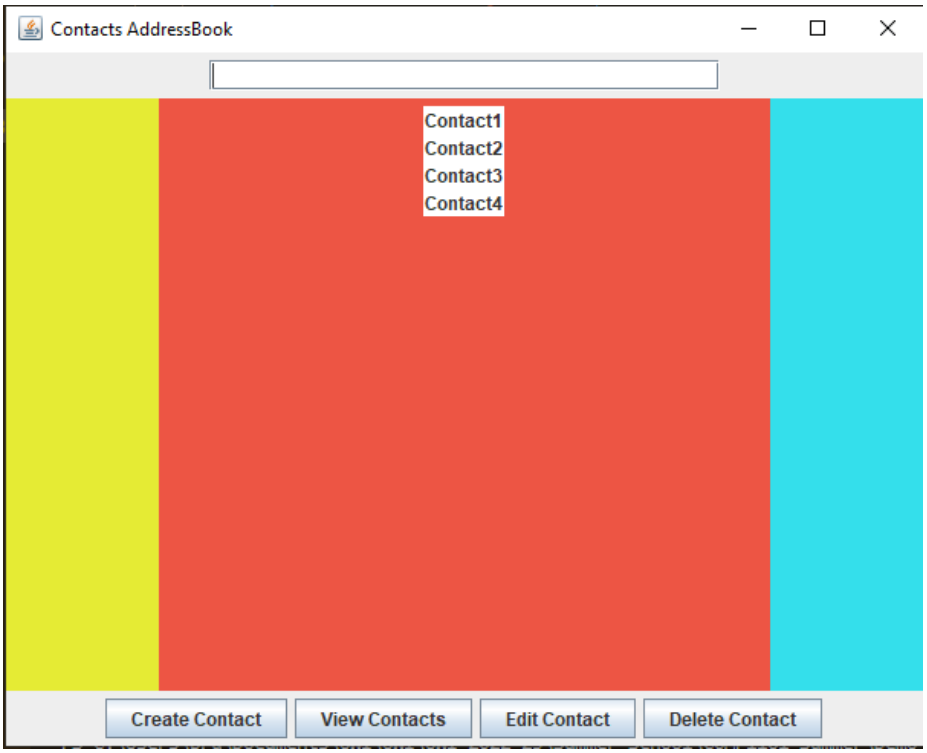

Figure 1 here shows an example of a GUI page.

You don't necessarily have to use the exact appearance like this but rather make a neat GUI that will do the functionalities like that of the Text User Interface, i.e., utilizing your actionListeners, itemListeners (if any), and your components that make a Graphical User Interface via (awt and swing java libraries). No code has been given for the panel class(es) so you can write your own.
However, note that the ContactsGUI class has a main entry in the JFrame as well as adding the ContactsPanel constructor to the contentsPane so **do not modify that code at all since that panel is the launch appearance for the GUI.**
You are allowed to create multiple JPanel classes to trigger some of the functional events listed in the ContactsPanel class and those classes can have their own main method to bring up another window of the task and closes it after its completion. These classes can be stored in the ui package in the source folder. Therefore, all your panel classes should be stored in the ui package.

As from the last lecture regarding creating a GUI, ie, adding panel classes, you can use a gui builder app and generate one rather than doing it from scratch. Just be mindful that what code you generate from the builder, the constructor code is what should be in the ContactsPanel panel as well as any other panels that you create a GUI for to generate code.

Also, if you want to add additional features such as images or anything else that is fine but really focus on the functional requirements.

Both parts are worth out of 100 and coursework grade of 20%. Will provide a mark scheme later on how each part is graded.

Compiling the Program

1. Look in the project folder when using the terminal/command prompt. When you run dir or ls you should see the list of folders:

   ```
   src, bin, docs and/or any other directories that you have
   ```

2. Call the javac program on all the files in the src folder passing the option to put the compiled binaries into the bin folder. If you have packages (*.java files in sub-folders under src) to compile as well, specify those before.

```
javac -d bin src/app/*.java src/contact/*.java src/data/*.java src/ui/*.java  src/*.java
```

   Note that there are multiple packages in the source folder: app, contact, data and ui

This will compile all the source codes in the sub-folders under src before compiling the java files in the top-level src folder. If you wish to compile certain packages you can do so without compiling everything but be mindful that some packages import from other packages.

Generating Java Documentation (Required)

We will using the Javadoc utility to generate our java documentation as shown in the lecture on Monday June 26, 2023 or you can look at the Creation of Jar File Manual. In creating the documentation, you will have to add them to the docs directory. Ensure that all methods and constructors for each class have documentation in them.

```
javadoc -d docs src/app/*.java src/contact/*.java src/data/*.java src/ui/*.java  src/*.java
```

Executing The Java Program

After compilation is complete, you can execute the program. In doing this, use the Main class since that class contains the main method. A text-based UI should appear.

```
java -cp bin ContactProgram
```

This will bring up a menu prompting you to select between a text user interface or a graphical user interface.

SUBMISSION INSTRUCTIONS

You are required to submit a Java Archive (jar file) **NOT A ZIP ONE** containing the src folder, the package folders and other files and folders that were included when creating the Jar file. The jar is required to be executable since this is where your application will be launched.

By now you guys should be veterans in creating the jar file but if any questions, let me know asap. You can work in a group with no more than 4 persons and submission should have the format below:

id1_id2_id3_id4_project2_3.jar

The due date will be on Saturday July 15, 2023 at 11:55pm. There is an option for extension up to July 19, 2023. This will be demonstrated a day or two after submission with me.

All the best!