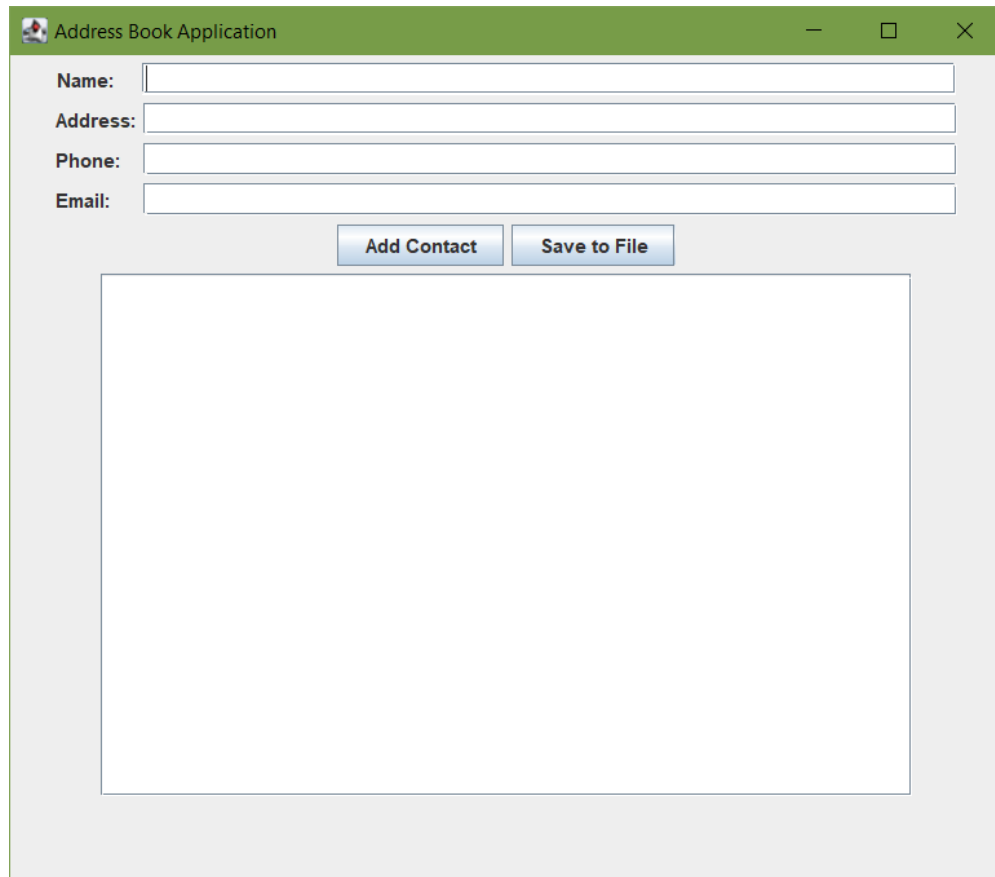


**CSE 271 Lab 8 – Graphical User Interfaces and Events**  
**Spring 2022**  
**Assigned: 3/17/2022**  
**Due: 3/20/2022**

**Introduction:**

In this lab, we will practice making a Graphical User Interface (GUI) and implementing action listeners. Create a project in your Eclipse IDE called **Lab8** and a class named **AddressBook** which has the following graphical user interface:



**Program Functionality:**

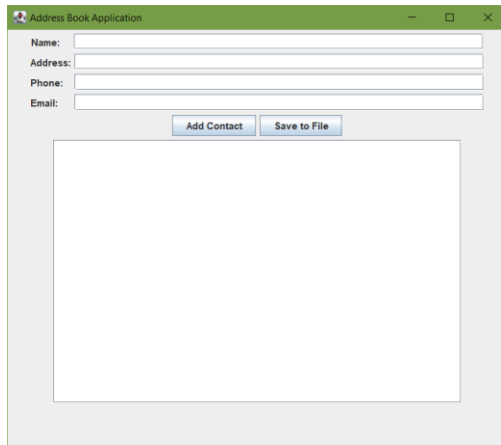
From top to bottom, there are 4 JLabel objects and 4 JTextField objects one after another. So, 4 rows of 1 JLabel and then 1 JTextField. Then, there are 2 JButton objects followed by 1 JTextArea. Also, the JTextArea will be used within a JScrollPane as shown during the lecture. Once the information has been typed into the 4 JTextFields, the user can press the "Add Contact" JButton to add the contact to the address book. That is, append the information from the JTextFields to the JTextArea. The JTextArea shows all the contacts of the address book separated by a newline character. The user may also press the "Save to File" JButton to save all the contacts listed in the JTextArea to a file named "contacts.txt". Whenever the user runs the application, it loads all the contacts from the file "contacts.txt" and prints them to the JTextArea.

### Step by Step Procedure:

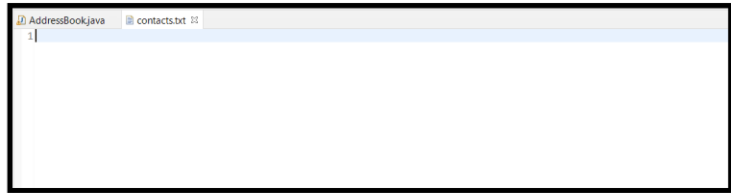
1. Your **AddressBook** class must inherit the **JFrame** class so that you can build the GUI.
2. In the **AddressBook** constructor, you have to instantiate and add all the components (JLabels, JTextFields, JButtons, and JTextArea) one after another following the GUI shown above. Remember to use a JPanel to add the GUI components first then add the JPanel object to your JFrame. Declare the components as private instance properties of the class. Note that the JTextArea should not be editable by the user, so utilize the **setEditable(boolean isEditable)** method that is available in the JTextArea class. Don't forget you must provide action listeners for your two JButtons by either using an explicit inner class or with an anonymous class.
3. Suggested sizes for the various GUI elements:
  - a. JTextArea: rows = 20, columns = 50
    - i. Note that the rows and columns stand for number of printable lines (rows) and the number of characters per line (columns).
  - b. JTextField: columns = 50
  - c. JFrame: width = 630, height = 550
  - d. JLabel: use appropriate number of spaces after the colon (this will require iterative guessing), e.g., **new JLabel("Phone: ")**
4. Set the default close operation to be **EXIT\_ON\_CLOSE**.
5. After adding all of the components to the JFrame in the constructor, you need to call a method named **readContactsFromFile()** which reads all the contacts currently saved in the "contacts.txt" file and prints them to the JTextArea.
  - a. **public void readContactsFromFile()** – This method reads the contacts from the "contacts.txt" file. It then constructs a String object of all the contacts and uses the **setText(String text)** method available in the JTextArea class.
6. For the JButton action listeners, follow the information below:
  - a. Add Contact JButton: Extract the information from the JTextFields and concatenate using a comma and a space. Then, append the concatenated String to the JTextArea using the **append(String text)** method available in the JTextArea class. Finally, it sets the text of each JTextField to be an empty String as to remove the current information that is present.
  - b. Save to File JButton: Call a method named **writeContactsToFile()** which extracts the text from the JTextArea and overwrites the information in the "contacts.txt" file. You have to define the method as **public void writeContactsToFile()**.
7. Write the main method in the **AddressBook** class. Inside the main method, create an **AddressBook** object and use it to set the size, title, default close operation, and visibility of the JFrame.

## Visual Workflow of the Program:

1. First time running the application with no information in the “contacts.txt” file:

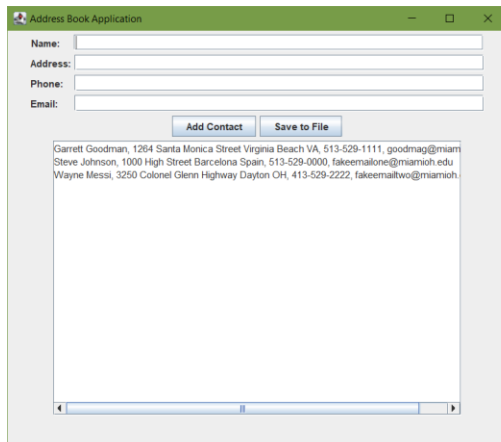


The screenshot shows the 'Address Book Application' window. It has a title bar with a green icon and standard window controls. The main area contains four input fields labeled 'Name:', 'Address:', 'Phone:', and 'Email:'. Below these fields are two buttons: 'Add Contact' and 'Save to File'. A large empty rectangular box is positioned below the buttons.

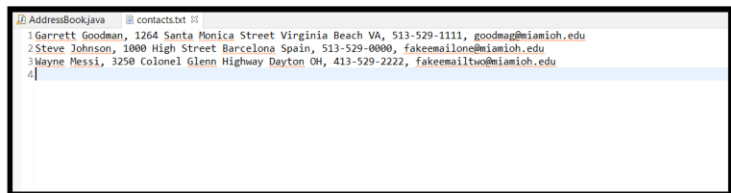


The screenshot shows a text editor window with two tabs: 'AddressBook.java' and 'contacts.txt'. The 'contacts.txt' tab is active, displaying a single line of text: '1'.

2. First time running the application with information in the “contacts.txt” file:

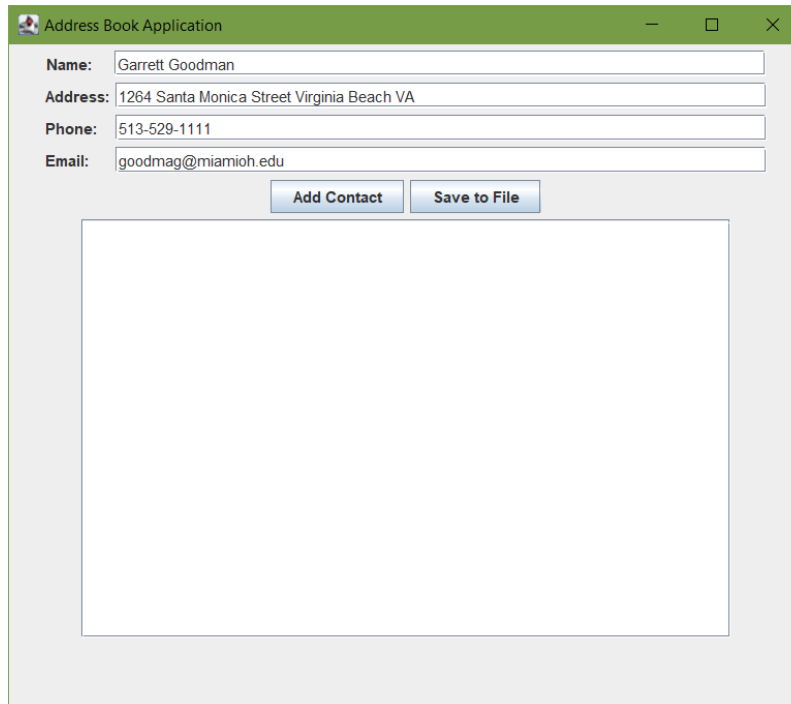


The screenshot shows the 'Address Book Application' window. The input fields are now pre-filled with contact information. Below the buttons, the text area displays the following information: 'Garrett Goodman, 1264 Santa Monica Street Virginia Beach VA, 513-529-1111, goodmag@miamioh.edu', 'Steve Johnson, 1000 High Street Barcelona Spain, 513-529-0000, fakeemailone@miamioh.edu', and 'Wayne Messi, 3250 Colonel Glenn Highway Dayton OH, 413-529-2222, fakeemailtwo@miamioh.edu'. A scrollbar is visible at the bottom of the text area.



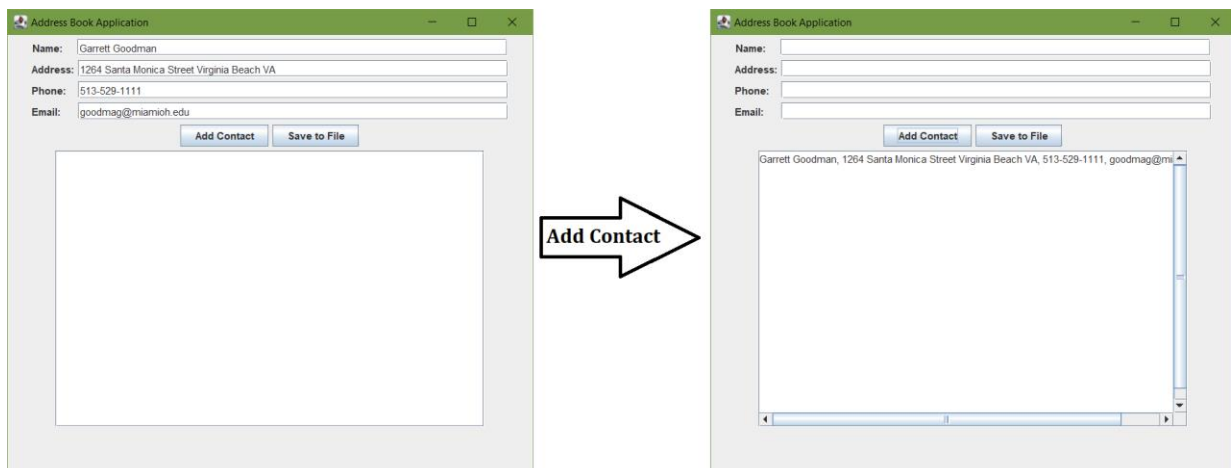
The screenshot shows the 'contacts.txt' file with three lines of text: '1 Garrett Goodman, 1264 Santa Monica Street Virginia Beach VA, 513-529-1111, goodmag@miamioh.edu', '2 Steve Johnson, 1000 High Street Barcelona Spain, 513-529-0000, fakeemailone@miamioh.edu', and '3 Wayne Messi, 3250 Colonel Glenn Highway Dayton OH, 413-529-2222, fakeemailtwo@miamioh.edu'. The first line is highlighted.

3. The user types information in the JTextFields:

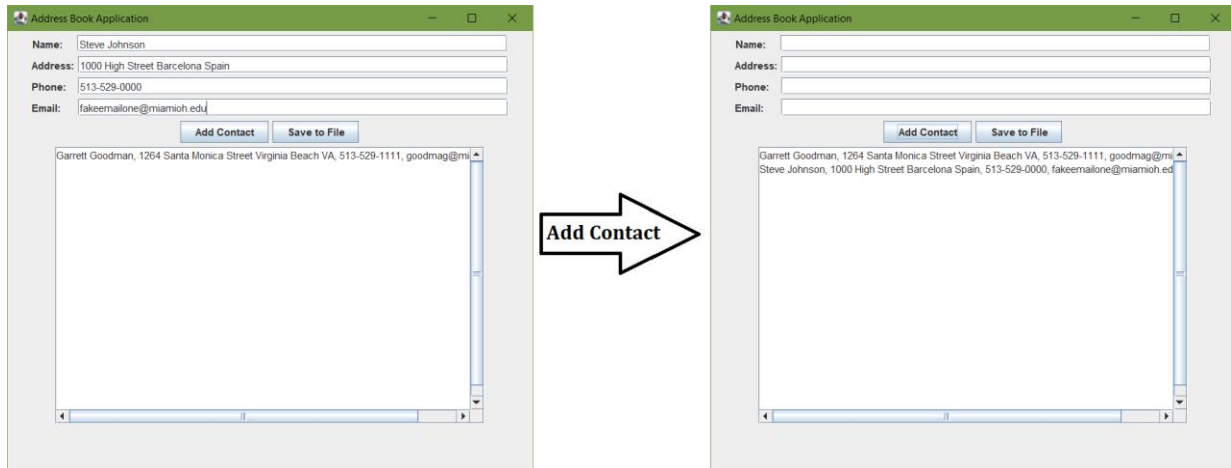


The screenshot shows a window titled "Address Book Application". It contains four text input fields labeled "Name:", "Address:", "Phone:", and "Email:". The fields are filled with the following text: "Garrett Goodman", "1264 Santa Monica Street Virginia Beach VA", "513-529-1111", and "goodmag@miamioh.edu" respectively. Below the fields are two buttons: "Add Contact" and "Save to File". A large, empty rectangular area is positioned below the buttons.

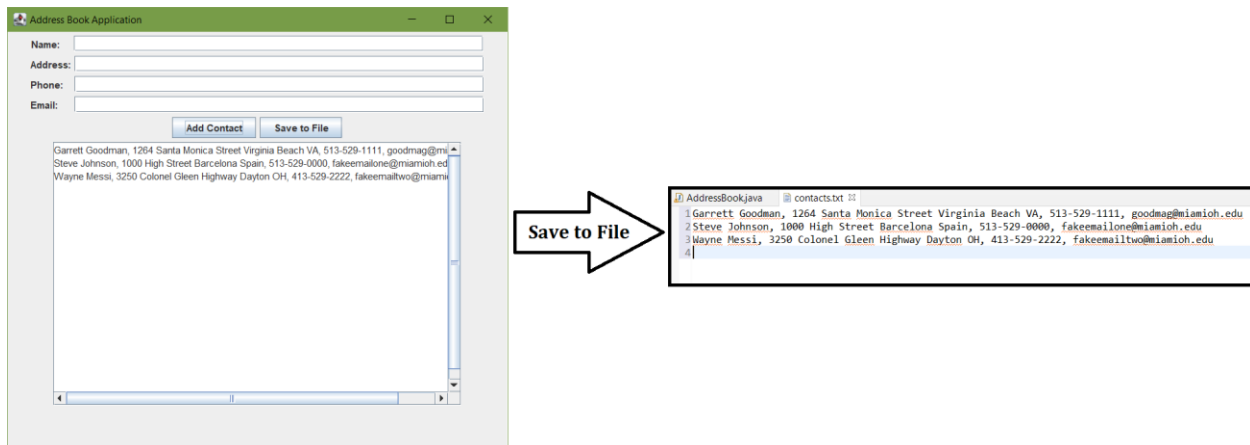
4. The user pressed the **Add Contact** button to add a single persons contact information to the JTextArea via the **append(String text)** method available in the JTextArea class:



5. The user added one more contact to the JTextArea.



6. Saving to the “contacts.txt” file by pressing the **Save to File** button:



### Additional Notes:

- Make sure you include JavaDoc comments for all methods and the class which includes parameter and return descriptions.
- Make sure that all classes are named correctly.
- There is no explicit validation checking needed for this Lab assignment.

### Submission Instructions:

After you have completed the lab assignment, locate your source code (**AddressBook.java**) in your workspace and submit it to the corresponding Lab 8 Canvas folder.

**Rubric:**

| <b>Task</b>   | <b>Grade</b> |
|---|--------------|
| <b>AddressBook</b> class extends JFrame   | 3            |
| Components correctly added (11 componenets: JLabels, JTextFields, JButtons, JTextArea)  | 35           |
| Used JPanel to add components   | 3            |
| Correctly applied an action listener to both buttons  | 8            |
| Reads data from the “contacts.txt” file when program runs and sets the JTextArea  | 12           |
| Appends to the JTextArea when the <b>Add Contact</b> button is pressed  | 5            |
| Writes contacts from the JTextArea to the “contacts.txt” file when the <b>Save to File</b> button is pressed  | 12           |
| Sets JFrame properties (size, title, and default close operation)   | 6            |
| Correctly sets the visibility   | 2            |
| Followed the step-by-step guideline (GUI design, appropriate method headers, constructor, main method, readContactsFromFile, and writeContactsToFile) | 10           |
| Adequate JavaDoc provided for the class and all methods and followed the Miami University coding guidelines   | 4            |
| <b>Total</b>  | <b>100</b>   |