Lab Assignment 08, Object-Oriented Programming, CSE 271, Spring 2020
Department of Computer Science and Engineering, Miami University

Graphical User Interface and Events

In this lab, you will practice how to create graphical user interfaces and implement an action listener. You must watch the following recorded lecture to complete this lab assignment:

- https://miamioh.instructure.com/courses/117194/discussion_topics/832690

You should also watch the recorded lecture posted last week:

- https://miamioh.instructure.com/courses/117194/discussion_topics/828979

Please review slides 37-52 from ***Lecture_GUI*** posted on Canvas to review JTextField and JTextArea along with the ActionListener.

Now, create a project in Eclipse named **Lab_08.** Now create a class named **AddressBook** which has the following graphical user interface:

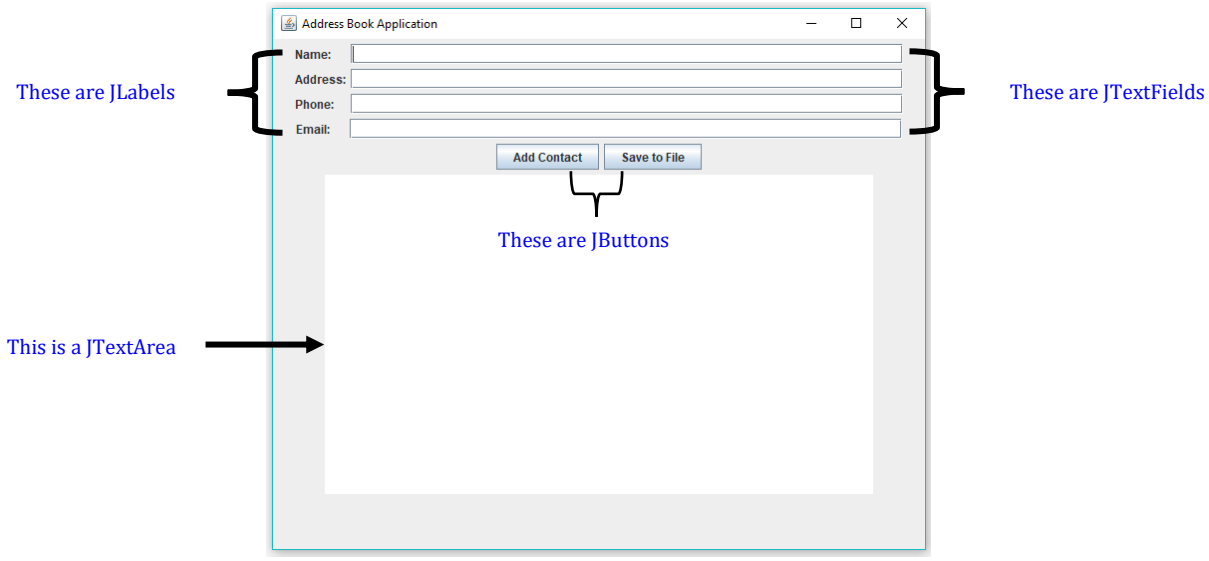


Figure 1: Graphical User Interface of the Address Book Application

Functionality of the program:

Using the JTextFields user can type name, address, phone number, and email address of a person. After typing all the information, the user can press the "Add Contact" JButton to add the contact (information from the JTextFields) to the address book, i.e. append the information from the text fields to the text area. The JTextArea shows all the contacts of the address book each separated by a newline character. At any moment, the user can save all the contacts from the JTextArea to a file named "contacts.txt". Whenever user runs the application it loads all the contacts from the file "contacts.txt" and shows on the JTextArea.

To understand the functionality of this program, look at the workflow of the program after "Grading Rubric" section (on page 3 and 4).

Step by step procedure:

1. To design the window your **AddressBook** class needs to inherit **JFrame** class. Also, it needs to implement **ActionListener** interface to add functionality to the buttons.

2. In the **AddressBook** constructor, you have to initialize and add all the components (JLabels, JTextFields, JButtons, and JTextField) one after another following the graphical user interface shown in Figure 1. Remember to use a JPanel to add components to JPanel object first and then add JPanel object to the JFrame object. Declare the components as instance variables of the class. You also need to register the buttons using "**addActionListener**" method and provide parameter "this". Here we are registering both buttons to a single action listener i.e. "**actionPerformed**" method, which you need to override in the AddressBook class.

Hint:
To initialize a JTextArea object use the following code snippet:
JTextArea contactsTextArea = new JTextArea(20, 50);
It creates a JTextArea object which can hold 20 lines (rows) each 50 characters long (columns).

Here are the recommended sizes for JTextFields, JTextArea and JFrame to have a view similar to the one presented in Figure 1:

- JTextField: columns: 50
- JTextArea: rows: 20, columns: 50
- JFrame: width: 670, height: 550

Also, make the default close operation ***EXIT_ON_CLOSE***.

3. After adding all the components to the JFrame object in the constructor, you need to call a method named "**readContactsFromFile**()" which reads all the contacts from the file named "contacts.txt". You have to define the method with the following header:
public void readContactsFromFile()
It reads the contacts from the "contacts.txt" file and constructs a string object. Then it sets the text of the JTextArea using the **setText(String text)** method.
Hint:
To set the string contacts to the JTextArea object "contactsTextArea" use the following code snippet:
contactsTextArea.setText(contacts);

4. Implementing button actions:
Both buttons register to the same action listener i.e. pressing any of the buttons will result calling the same actionPerformed(ActionEvent event) method. You can use the following if…else block to decide what to do based on the source of the event i.e. which button has been pressed.

```
@Override
public void actionPerformed(ActionEvent event) { // start of the method body

	if(event.getActionCommand().equals("Add Contact")) {
		// Append information from the JTextFields to JTextArea object
	}
	else if(event.getActionCommand().equals("Save to File")) {
		// Write contacts from the JTextArea to the file "contacts.txt"
	}
}// end of the method body
```

   a. Add Contact:
   Get the information from the text fields and concatenate using a comma and a space. Then append the concatenated string to the JTextArea. You can use the following code snippet:
   contactsTextArea.append(concatenatedString);

   b. Save to File:
   Call a method named "**writeContactsToFile**()" which gets the text from the JTextArea and writes (previous content will be replaced) to the file named "contacts.txt". You have to define the method with the following header:
   public void writeContactsToFile ()

Lab Assignment 08, Object-Oriented Programming, CSE 271, Spring 2020
Department of Computer Science and Engineering, Miami University

Graphical User Interface and Events

5. Write the main method in the **AddressBook** class. Inside the main method, create a **AddressBook** object. Use that object to set the size, title, default close operation of the window. Then make it visible.

Important Note:
Make sure the file name is correct and code is well commented. No late submission. You will get zero for late submission.

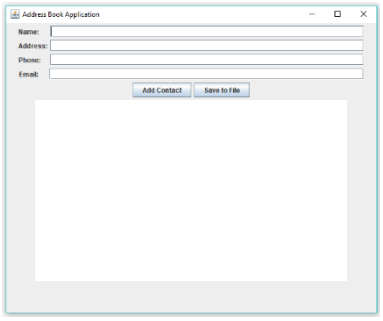
Submission:
Submit java file (no Javadoc) to the appropriate submission folder on the Canvas by the due time.

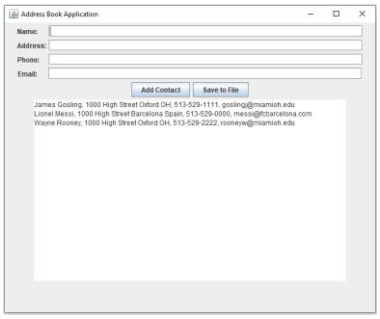
Grading Rubric:

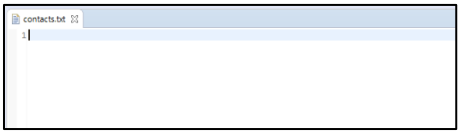
Class extends JFrame and implements ActionListener	2
Adding components (11 components: JLabels, JTextFields, JButtons, JTextAreas)	33
Use JPanel to add components	3
Register buttons to the action listener (each 2 points)	4
Overrides actionPerformed method	3
Compares event source inside actionPerformed method	4
Loads information from the "contacts.txt" file at beginning and sets to JTextArea	12
Appends to JTextArea when **"Add Contact"** button pressed	5
Writes contacts from the JTextArea to the file "contacts.txt" when "**Save to File**" pressed	12
Sets window properties (size, title, default close operation)	6
Makes the window visible	2
Well-commented code (Javadoc and non-Javadoc comments)	4
Followed step by step guideline (GUI design, appropriate method header, constructor, main, readContactsFromFile, writeContactsToFile)	10
Total	**100**

Workflow of the program:

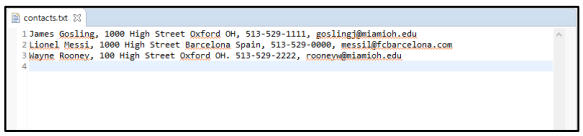
1. First-time running the application:







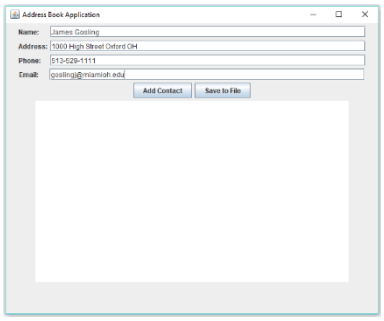
The "contacts.txt" file is empty



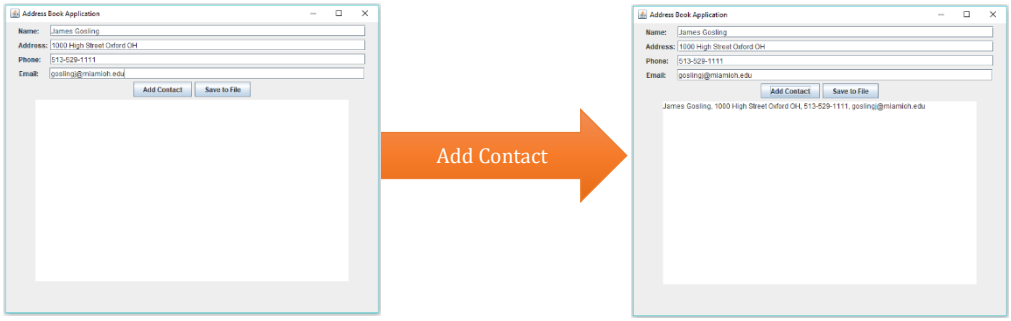
The "contacts.txt" file has contact information

Graphical User Interface and Events

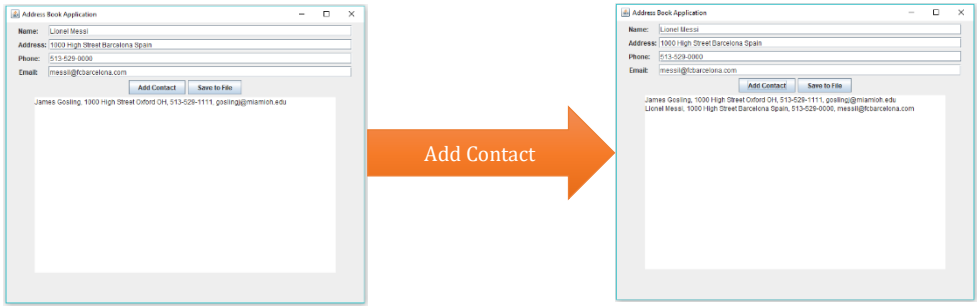
2. User types the information in the JTextFields



3. User then press “Add Contact” button to add contact and the information is appended to the JTextArea



4. Adding one more contact appends the new information to the JTextArea



5. Saving to the file by pressing “Save to File” button.