Hands-on Activity
Application of Event-Driven Programming

Objectives:

At the end of the exercise, the students should be able to:

- Describe the event handling process; and
- Incorporate events in a program.

Materials:

- One (1) personal computer with NetBeans IDE and Java Development Kit (JDK) 8
- Internet connection

Instructions:

1. Launch NetBeans. Click **File** > **New Project** > **Java Application** > **Next**. The project name should be **EventDriven_(LastName)** (ex. EventDriven_Reyes). Then, click **Finish**.

2. Create a simple program that includes *JFrame*, *JPanel*, *JLabel*, *JFields*, *JTextArea*, and *JButton*. The program should ask the user for the following input:
    - First name
    - Last name
    - Middle name
    - Mobile number
    - E-mail address.

3. Use the following methods and classes in creating the program:

public class EventDriven extends JFrame	This should be the main class of the program and should contain the list of components and its corresponding declaration.
public EventDriven	This method should include the formatting of components in the **INPUT** frame.
class btnSubmit implements ActionListener	This class should be within the *public class EventDriven extends JFrame* and should contain the actions that would be performed by the **Submit** button in the **INPUT** frame.
class btnClearAll implements ActionListener	This class should be within the *public class EventDriven extends JFrame* and should contain the actions that would be performed by the **Clear All** button in the **INPUT** frame.
class btnOkay implements ActionListener	This class should be within the *public class EventDriven extends JFrame* and should contain the actions that would be performed by the **Okay** button in the **OUTPUT** frame.
public static void main (String[] args)	This will serve as the main method of the program.

5. Two (2) frames will be used in this program: OUTPUT and INPUT. The following conditions must be satisfied by the program.

   Input Frame:
   - Set the window name to **INPUT**.
   - The input frame should collect all the details listed on Step 2.
   - It should contain **Submit** and **Clear All** buttons.
   - When the **Submit** button is clicked, it should generate the output frame and disable the **Submit** button.
   - When the **Clear All** button is clicked, it should clear all the content that were entered by the user in the input frame, and close the output frame if it is open.

   Output Frame:
   - Set the window name to **OUTPUT**.
   - The output frame should display all the details that were entered by the user with the corresponding label.
   - The output frame should contain an **Okay** button.
   - When the **Okay** button is clicked, it should automatically close the output frame and clear the content of the input frame.

6. See *Figures 1* and *2* below for the expected results.

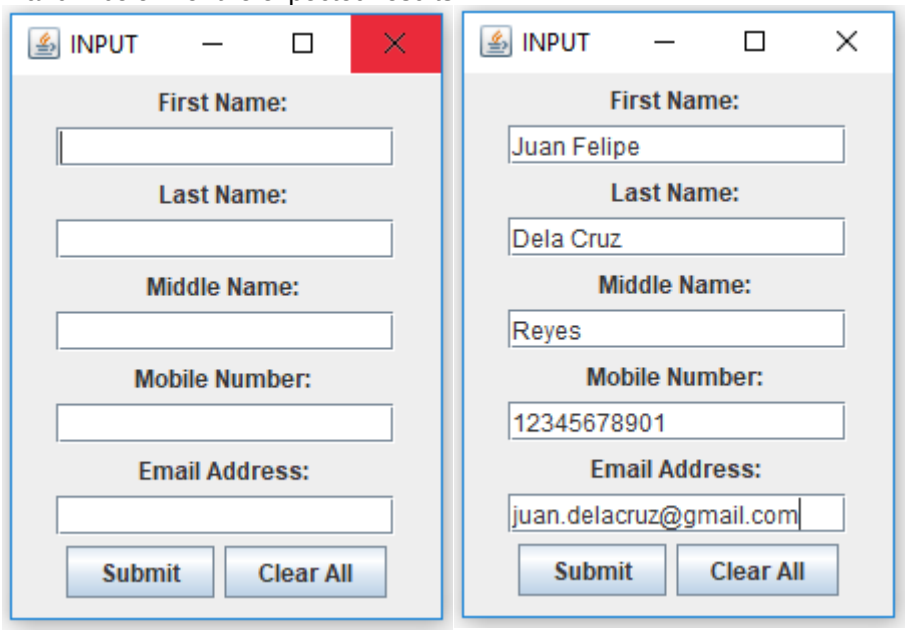

Figure 1. L–R: Initial output of the program with the input window; text fields with sample input

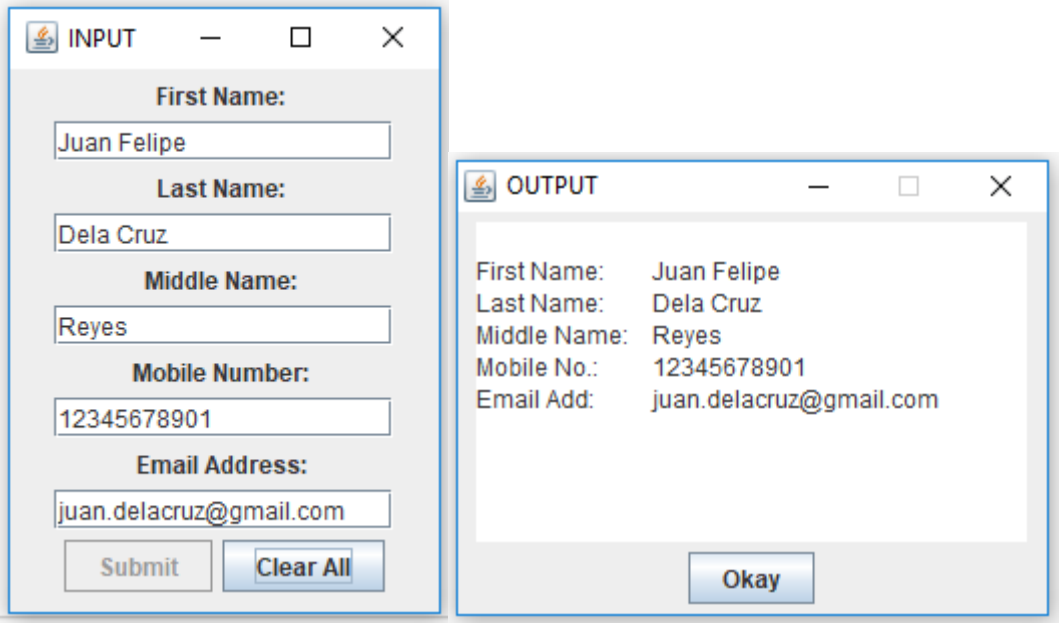

Figure 2. After clicking the submit button, the program should generate the output window (image on the right) and disable the submit button (image on the left).

Note: *Clicking the* ***Okay*** *and* ***Clear All*** *buttons will refresh the program and generate the image on the left under* Figure 1.

7. Use your personal information in filling out the input window. Screenshot your code and output.

GRADING RUBRIC (Application Design):

Criteria	Performance Indicator	Points
Correctness	The code produces the expected result.	40
Logic	The code meets the specifications of the problem.	30
Syntax	The code adheres to the rules of the programming language.	30
	TOTAL	**100**