# Unit 6 Assignment

## GENERAL

1. Save each class in a separate Java source code file and name it *StuPrefixClassName.java*. Follow the coding style as explained in Unit 1 Assignment.

2. Overall comment your program appropriately (file prolog comment, method prolog comment, and comments on the code). Pay attention to the standard stuff like coding style, meaningful identifier names, indention, and locations of braces.

3. When you’re done with an exercise, include a screenshot of the execution of the program in your assignment report document. This is to remind you to verify the execution result.

## EXERCISES

We will build a GUI conference registration program.

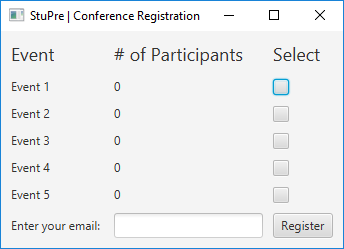


Figure 1. Startup Screen.

* The conference has multiple events a participant may register for.
* A registration requires a valid email address and selecting at least one event.
* The program will track the # of registrations per event.
* After the Register button is clicked,
  + Check and report any invalid registration attempt:
    - No event was selected
    - no email address or an invalid email address was provided
  + Errors should be reported. Processing will end once an error is identified.
  + A successful registration should be confirmed.

Reminder: all GUI components you use should come from the JavaFX GUI toolkit, i.e. all your GUI related imports should be import javafx.xx.xx… Do NOT use the AWT GUI (your import will say import java.awt.xxx) or the Swing GUI (your import will say import javax.swing.xxx).

Build your program following those exercises:

1. Use package for your project. Name the package with your last name. For example, student John Doe will name his package doe.
2. (Pick a proper name for your program and add the StuPre prefix). Set up the GUI first.

Your GUI needs to be similar to the sample screenshot in Figure 1, but not necessarily 100% match (label vs. read-only textfield, font, padding, spacing, etc.). The sample GUI is set up with a GridPane, row 0 for the header, row 1 ~ n for the number of events we have, and row n+1 for email entry (TextField) and the button.

Include your StuPre prefix in the window title.

The last column (of the “grid”) uses CheckBox:

*checkBoxObj* = new CheckBox(); // a checkbox without any text

* ch6.4.3 of <http://math.hws.edu/javanotes/c6/s4.html>
* <https://openjfx.io/javadoc/11/javafx.controls/javafx/scene/control/CheckBox.html>

This program will be extended later to load/save event data from/to file and handle any # of events and different events, so the event data is stored in arrays and made instance members:

private String[] eventNameList = {"Event 1", "Event 2", "Event 3", "Event 4", "Event 5"}; // feel free to use actual event names of your choice

private int[] countList = new int[eventNameList.length]; // count of participants for each event

Your GUI setup code needs to work with any # of events and event names that’s been loaded into the eventNameList array (i.e. not just 5 events. Maybe 7 events next time, or 10 events).

The labels displaying # of registered participants and the checkboxes will be updated during event handling. It’s also preferred to process them in a loop (therefore can handle any # of events). So those are set up as instance array members too:

private Label[] countLabelList = new Label[eventNameList.length];

private CheckBox[] checkBoxList = new CheckBox[eventNameList.length];

// Hint: array created, but elements are null.

// Need to create individual GUI components

You may wait until the event handling step, but figure out any additional instance data members the program may need.

Be sure to run your program to test the GUI before moving to the next step. Take a screenshot of the execution of your program at this point and include it in the assignment report document.

1. Now add the event handling code. The event handling code must be put in a private helper method called processRegistration():

*someButton*.setOnAction( e -> processRegistration() );

This helper method should follow this algorithm:

|  |
| --- |
| // Step 1.  Check if at least one checkbox has been selected.  If not, report error and end processing.  /\* Hint: use "return;" to end a void method early \*/    Figure 2. Report no events were selected. (Include such a screenshot of your execution) |
| // Step 2.  Check if a valid email address has been input.  If not, report error and end processing.  /\*  Set up a boolean helper method to decide whether a string contains a valid email address. Return true if valid.    For the purpose of this program, an email address input is valid if it has only one '@', has at least one '.', and the last occurrence of '.' is after the '@'.  You may know a more advanced Java technique to validate an email address, but you must write this method as required.  You should also check for an empty input, including inputs with only spaces.  \*/    Figure 3-1. Report an empty input with only spaces (see next figure). Pay attention to the header text and content text with each alert. (Include such a screenshot of your execution)    Figure 3-2. After user closes the alert window, the email textfield should gain focus and its content highlighted. You don’t have to, but the sample program also clears all checkbox selections.    Figure 3-3. An invalid email entered. After user closes the alert window, the email textfield should gain focus and its content highlighted (). You don’t have to, but the sample program also clears all checkbox selections. |
| // Step 3. Now we have a valid registration.  // Process  increment countList[i] of each selected event.  Update the labels to display the new counts.  Clear the email input.  Clear all checkbox inputs. // may be completed in an earlier step  Confirm the registration.  /\*  This program doesn’t check repeated registration of the same event from an email address. That would be in the next phase of this project.  \*/    Figure 4-1. A valid registration input.    Figure 4-2. Confirmation for the valid input in Figure 4-1. Note that the email address and event names are included in the content text. (Include such a screenshot of your execution)    Figure 4-3. Processing result for the valid input in Figure 4-1.    Figure 4-4. Another valid input.    Figure 4-5. Processing result for the valid input in Figure 4-4 (after a similar confirmation as shown in Figure 4-2). |

Take screenshots of the final execution of your program as in Figure 2, 3-1, and 4-2 and include them in the assignment report document.

1. Reflection: answer those questions AFTER you’ve completed this assignment:
2. What’s the hardest part of this assignment for you? Please explain.
3. Identify at least one thing that you like or dislike about GUI programs. Provide a brief explanation.

## SUBMISSION

Submit one .java files + one word/PDF document. Please put screenshots and answers to questions into your word/PDF document.

* Exercise 2: one screenshot showing GUI step work.
* Exercise 3: three required screenshots of execution and a java file.
* Exercise 4: Assignment reflection
* Check the completeness of your work against the rubric before turning it in.

## Rubric: Unit 6 Assignment

| **Criteria** | **Ratings** | | | **Pts** |
| --- | --- | --- | --- | --- |
| **Exercise 1 (package setup)** |  | 0.5 pts. Correct package statement in source code. | 0 pts. Incorrect or no submission. | 0.5 |
| **Exercise 2**  **(GUI setup)** | 5.5 pts. Correct JavaFX GUI meeting all requirements. | 5.0 ~ 1 pts. At least one requirement is incorrect or missing:  (0.5 pts) the # of events matches the length of the event name instance array member;  (1 pts) each event name is pulled out of the instance array member;  (1 pts) a proper UI control is used to display event name of each event;  (1 pts) a proper UI control is used to display the # of registered participants (initial value 0) of each event;  (1 pts) a proper UI control is used for user to select each of the events;  (0.5 pts) a proper UI control for email address input;  (0.5 pts) a button to register. | 0 pts. Incorrect or no submission | 5.5 |
| **Exercise 3 (Event handling)** | 11 pts. Correct event handling meeting all requirements. | 10 ~ 1 pts. At least one requirement is incorrect or missing:  (1 pts) event handling code in a private helper method;  (2 pts) identify and report with a proper Alert window if none of the checkboxes were selected;  (2 pts) identify and report with a proper Alert window if an invalid email address;  (1 pts) a private boolean helper method to decide whether a string contains a valid email address;  (1 pts) processing of a valid registration attempt: update the instance countList array;  (1 pts) processing of a valid registration attempt: update the labels displaying the participant counts;  (2 pts) processing of a valid registration attempt: a confirmation through a proper Alert window displaying email address and registered events;  (1 pts) processing of a valid registration attempt: email input and checkbox selections cleared. | 0 pts. Incorrect or missed all requirements; no submission | 11 |
| **Exercise 4. Reflection** | 1 pts. Answered both questions. | 0.5 pts. Answered only one question. | 0 pts. Didn’t answer the questions or no submission. | 1 |
| **Required program name (StuPre part) + Style Points (Proper comments; meaning identifier names; consistent indentation)** | 2 pts. Correctly named program and proper style in all three areas of style points. | 1 pts. Problems in one of the following areas: program name, three areas of style points. | 0 pts. Problems in two or more of the following areas: program name, three areas of style points. | 2 |
|  |  |  | *Total Points* | 20 |