CSE1322L - Lab 12

<u>Concept Summary:</u>

Using Java-Fx and / or Windows Forms

This week we are moving on from text-based programs to programs that have a GUI (Graphical User Interface). What does this mean? A GUI is what is referred to as the *front-end* of a program i.e. it is what you see and interact with when you use a program on your computer that is not text-based (like windows, buttons, scrollbars, textboxes, etc.…) and a *back-end* which is a regular code text file that contains all the methods that each detail how any of the GUI objects should act and variables that store any necessary values for the program to function as intended.

Topic Listing

1. Pages 1-5 Discuss Important Details for Java Students
2. Pages 6-8 Discuss Important Details for C# Students
3. Page 9 is the Lab

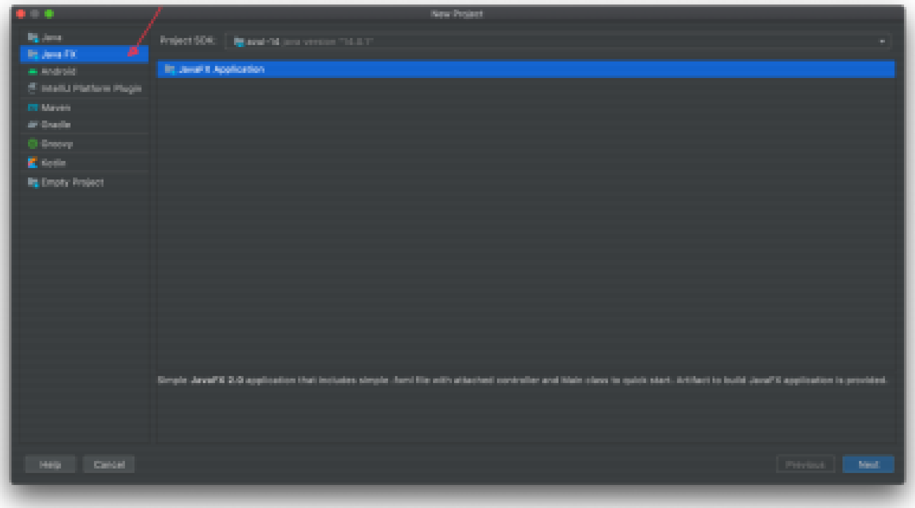
<u>Specifications: - Installing Java-FX</u>

To be able to build these programs with a GUI in Java we will be making use of a Java Library called *JavaFX* in IntelliJ.

Hopefully you already have IntelliJ installed and working.

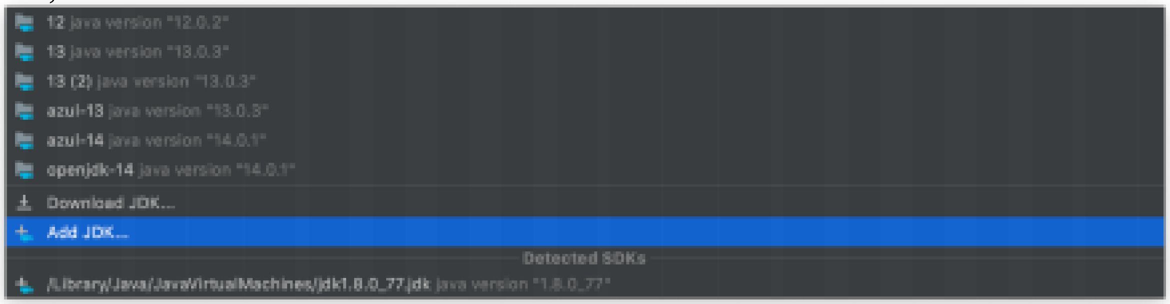
You'll need to add a new JDK to your IntelliJ install. Follow these steps:

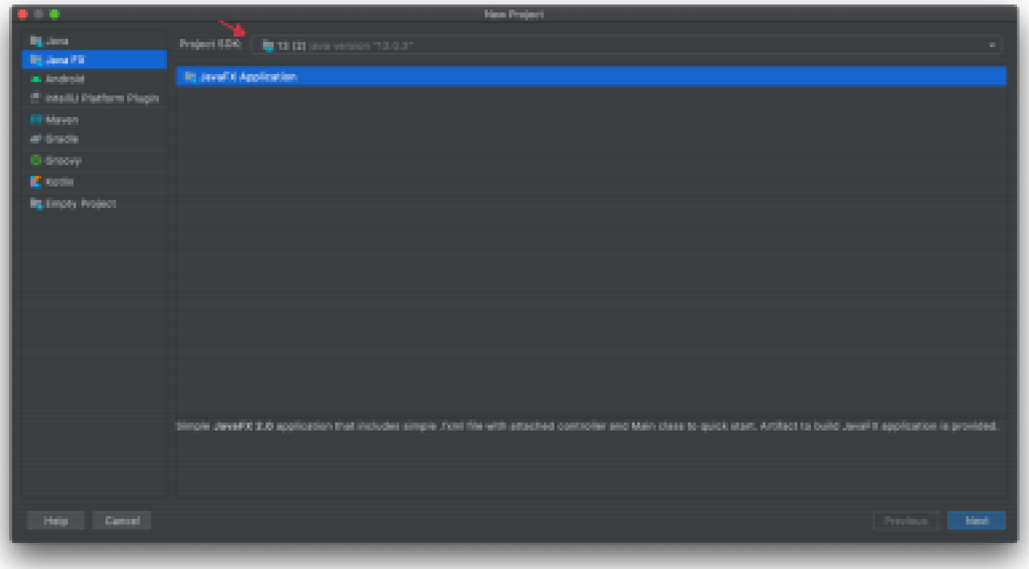
1) Download the appropriate JavaFX JDK from this link: https://www.azul.com/downloads/zulu-community/?version=java-13-mts&package=jdk-fx
    a) Ensure you get a version 13 JDK-FX (There are also JDK, which won't work).
    b) Download it and unzip it into a folder. Never delete it, as it'll be used for each program you write in JavaFX. **Make sure the folder is unzipped before proceeding.**
2) Open Intellij, create a new project, select JavaFX on the left:



Simple JavaFX 2.0 application that includes simple .fxml file with attached controller and Main class to quick start. Artifact to build JavaFX application is provided.

3) When prompted for JDK, choose "Add JDK" from the drop down, select the folder you unzipped the above download into. You should now see Java 13 or Zulu 13 in the drop down, select it.


12 java version "12.0.2"
13 java version "13.0.3"
13 (2) java version "13.0.3"
azul-13 java version "13.0.3"
azul-14 java version "14.0.1"
openjdk-14 java version "14.0.1"
Download JDK...
Add JDK...
Detected SDKs
/Library/Java/JavaVirtualMachines/jdk1.8.0_77.jdk java version "1.8.0_77"



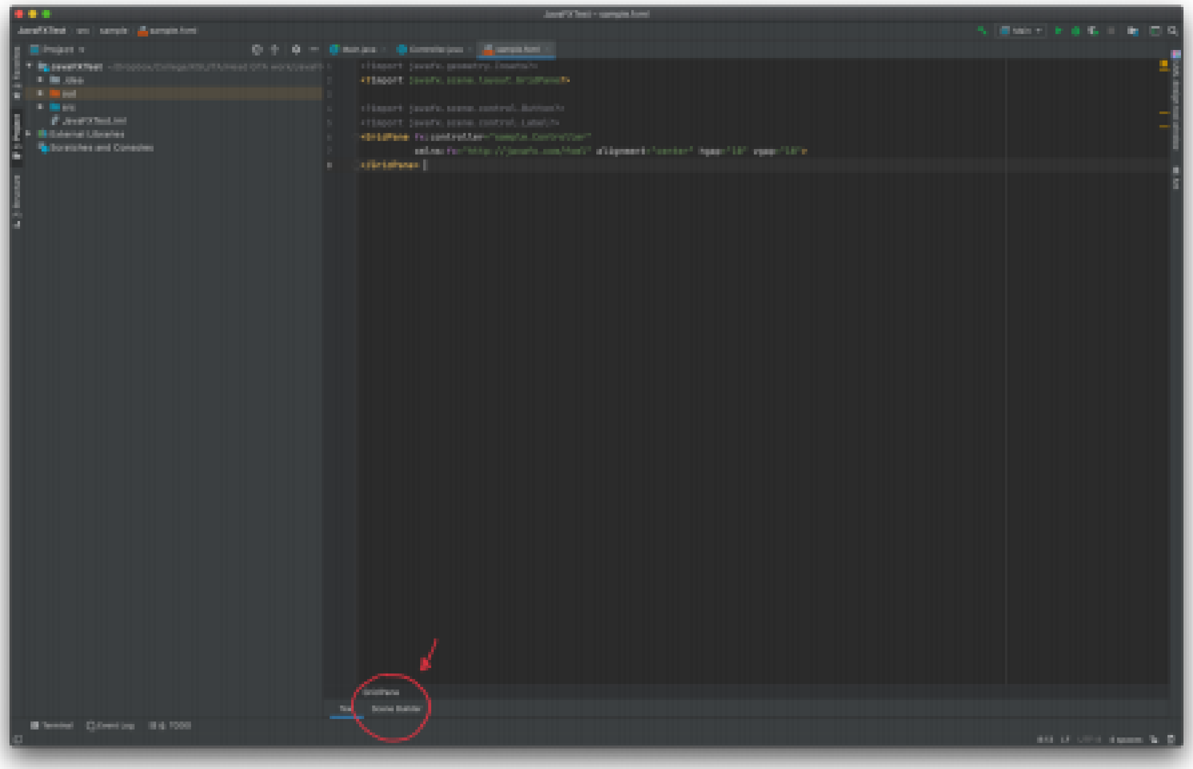
Project SDK: 13 (2) java version "13.0.3"

Simple JavaFX 2.0 application that includes simple .fxml file with attached controller and Main class to quick start. Artifact to build JavaFX application is provided.

4) If you did all the steps correctly as soon as the project loads you should be able to compile and run, and you should see a Hello World window pop up. If not, repeat the steps and ask for help. Nothing will work if that doesn't work, so don't go past this step until you have it working.
5) If you are on a mac, you may have to add the JavaFX library to your path. Follow the "Add JavaFX Library" section instructions: https://www.jetbrains.com/help/idea/javafx.html#add-javafx-lib
6) **Every time you make a new GUI program make sure you select 13 from the dropdown menu, you will never need to point IntelliJ to that folder again as long as you do not delete it**

Installing SceneBuilder (Java):

After creating your first test program; click on the *sample.fxml* file. On the bottom you will find a tab called "Scene Builder" click that.



```
<?import javafx.geometry.Insets?>
<?import javafx.scene.layout.GridPane?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<GridPane fx:controller="sample.Controller"
          xmlns:fx="http://javafx.com/fxml" alignment="center" hgap="10" vgap="10">
</GridPane>
```

You will immediately see a red exclamation mark with text asking you to install the "Scene Builder Toolkit" and a link. Please click that link which will install the Scene Builder for you automatically. The Scene Builder allows you to create these GUI apps by dragging and dropping components on a window like buttons or otherwise; makes it easier to design how your program will look while you implement functionality of these buttons with code; **this will only need to be done once**

4. Please refer to this link to create a test Program (which has code and step by step instructions provided to you). Please use this as a means to test if everything is set-up correctly:
   https://www.jetbrains.com/help/idea/developing-a-javafx-application-examples.html

NOTE - Please see below for instructions on how to add code to the Buttons using Java-FX within IntelliJ

Making Variables for the Controller – Labels and Text Fields (Java)

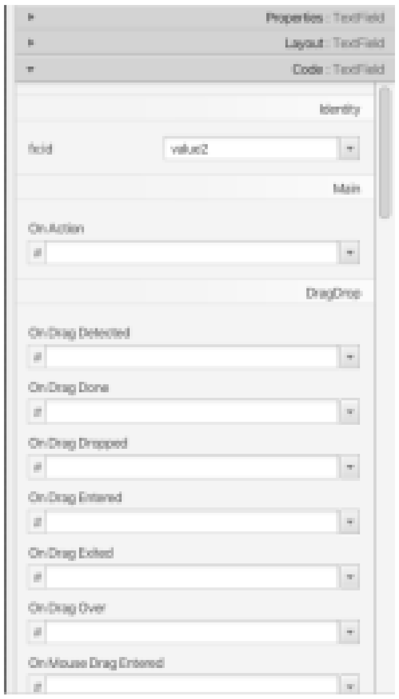
The variables in the Controller class are responsible for storing the values that are specific to your program's intent. For example, if we were to implement a calculator, there would exist a variable to store the value of the first number the user entered, another to store the second value, and a third to store the result. These variables would be the same ones that you are used to; of type integer, double etc…

Since, in a JavaFX GUI program, there exist no way to traditionally input and output information to and from the user we instead have two *special* variables to take care of something similar.

New variable types that you may need to use are the:

1- (Input) TextField variable which is a variable the holds the text entered into a text field you would have in your GUI

The primary way that your GUI program will take in information from the user would be through the use of a GUI text field linked to the TextField variable; you will need to link the GUI text field to a text field variable once declared in the Controller using the Scene Builder. When the user runs the program and types in text, the TextField variable will store the typed text to be used for a specific purpose that you specify.



fx:id value2

2- (Output) Label variable which would represent the text shown on a label in your GUI

The primary way that your GUI will output information to a user would be through the use of a GUI label linked to the Label variable. You will need to link the variable to its GUI element so that it reflects what the variable stores. When the user runs the program and interacts with it, this field should reflect some sort of output; for example, the result in the case of a calculator.

How to link a TextField or Label?

Once the either variable is declared in the Controller and GUI counterpart is created in the Scene Builder, click on the label/text field in the Scene Builder go the properties pane on the right hand side and open the “Code” section; specify the variable associated in the “fx:id” section.

<u>Methods (Event Handlers) – Specifying the Actions of Buttons</u>

Each button will need an event handler method for when they are clicked; you will make these event handlers in the Controller class before assigning them to their respective GUI button in the Scene Builder. The Event Handlers mostly look and work like regular methods. An Event Handler is responsible for handling the action that the program will undertake when a button is pressed; hence the name the event (button press) handler. This idea can be expanded to objects that are not buttons but the idea remains the same. The parameter of these methods should be (ActionEvent actionEvent); this parameter represents the button press.

For example:

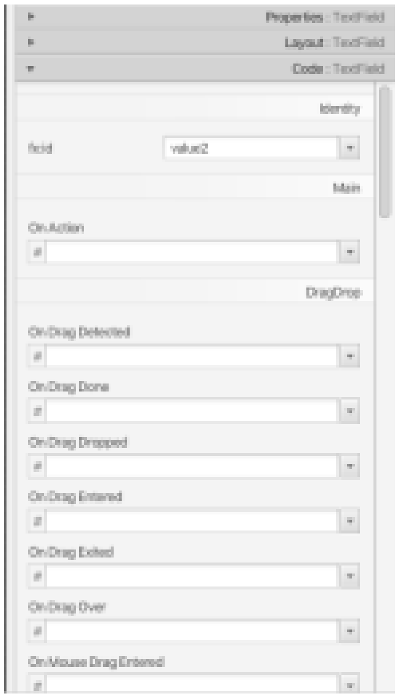
```
public void add(ActionEvent actionEvent) {
 //YOUR CODE
}
```

The function variableName.getText() can be used in an event handler for a button action to retrieve the value entered by the user from a text field variable associated with a GUI text field.

The function variableName.setText() can be used in an event handler for a button action to change the value in a label variable associated with a GUI label.

How to link an Event Handler to its GUI Button?

When visiting the Scene Builder and clicking to edit a button, on the left “Properties” pane please open the “Code” section, the “onAction” portion is where you will assign the event handlers that you have already made to their respective buttons. Please refer to the figure on the right.



fx:id value2

__Final Comments__

When making a JavaFX project in IntelliJ, the GUI component is handled by a file (by default) called the “sample.fxml” file; this is where you will find the Scene Builder. It will appear as text at first, please click the Scene Builder button on the bottom.

The Controller is the file called “Controller.java”. Please pay no mind to the “Main.java” file.

Please go to page 9 to start your first GUI Lab Exercise

C# - Creating an Example GUI program

Creating a Window

The first step to creating a GUI application is to use the correct project type. From within Visual Studio please select the "Windows Form Application". Now that we have a project set up to work with a GUI, let us get into the specifics on how to make a window.

NOTE: After the project finishes being made, the first page that should appear is the one with the window already made. You should see a blank white window in the center panel instead of the code you would normally find there when making a console application.

In order to see the code and be able to change it, please find the "Solution Explorer" panel which should be on the right. Find the "Form1.cs" file and click on the dropdown arrow to see the "Form1.Designer.cs" file and double click on it to open it.

NOTE: We still have a "Program.cs" file, however if you open it, it will only contain a few lines of code inside the Main method.

Now you should see the code behind the design of the window. However, do NOT write any code here yet.

We have already made our window, congratulations! (Feel free to change the look and size of the window using the "Properties" panel.)

Creating Buttons

We can make many types of interactable objects in our window, but we will keep it simple and use a clickable button.

1. First, go back to the "Form1.cs" file so that we can see the blank white window.
2. Please find the toolbox pop-out menu on the left side bar. Click on it to pop-out the menu.
3. Go to the menu tab called "All Windows Forms" and click once on the choice/item "Button".
4. Move the mouse cursor back over the blank white page and you should notice the cursor changes into a point with the "Button" icon on the lower right.
5. Clicking on the window will create a default button with a default size on that window. You may also click and drag to create a button with the size of the area your cursor moved over. Make sure to add the remaining buttons.

NOTE: The code in our "Form1.Designer.cs" file has changed, by adding a button object at the bottom of the code.

Creating an Event (Event Handler) (C#)

At this point, we will want to create an event object every time the button is pushed. This way we will be able to tell when the button is pushed and what to do when it is pushed.

1. To add an event object for clicking on the button, simply double click on the button in the window.
2. This will immediately take us to the “Form1.cs” file where a new method will have been created for us named button1_Click.
3. We have added the ability of the button to create events after being clicked (even without writing anything into the new click method)!

Creating an Event Listener

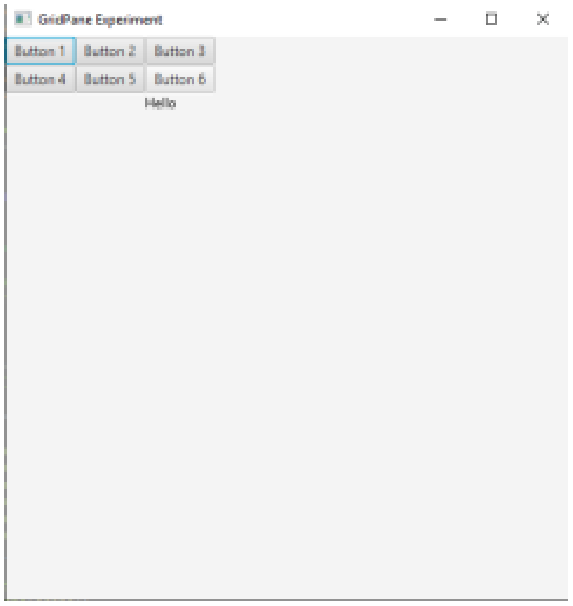
Lastly, we will create an event listener to handle the events (buttons being pushed) and display a message, on the GUI, saying which number button was pushed.

1. First, we will need to create a text box on the window so that we can display our message. Follow similar steps for creating the buttons, but this time look for the item “TextBox”.
2. Return to the “Form1.cs” file with our button1_click method. Using the new textbox object and the parameter object sender, write some code into the method so that the number of the sender (which was the button) displays in the text region of the textbox.

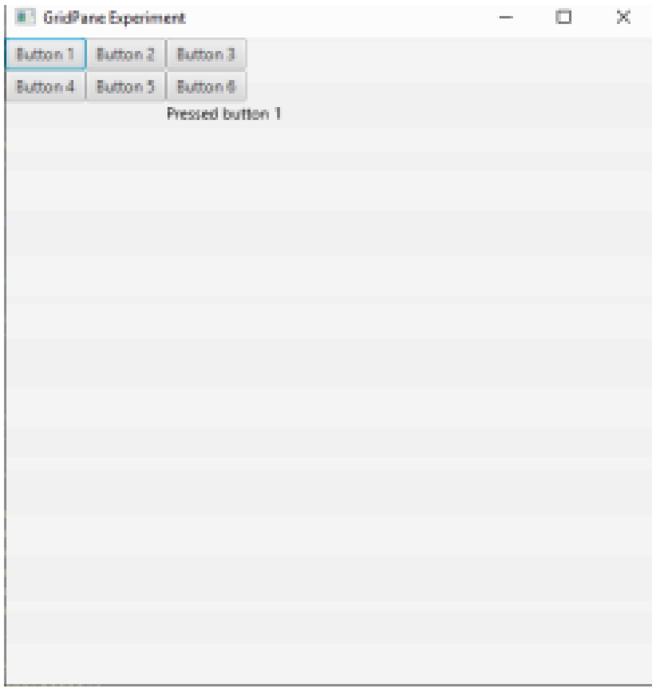
NOTE: The method button1_click is the event listener and it is handling events after the button is clicked and creates that EventArgs e event that is the second parameter in that method.

Sample Output (C#):

Before button 1 is clicked:



After button 1 is clicked:



Please go to page 9 to start your first GUI Lab Exercise

FIRST GUI LAB Exercise (Common for C# and Java)

NOTE (for Java):

1. **The "sample.fxml" file comes with a default GridPane object, please do not delete it as it will unlink the "sample.fxml" from the "Controller.java" file**
2. **When submitting, please upload all the files in the "src" folder including the Controller.java, Main.java, sample.fxml, and *projectName*.iml**

NOTE (for C#):

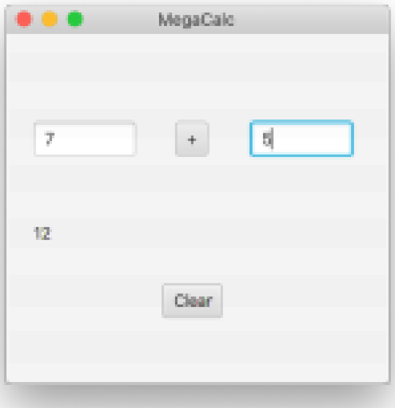
1. **Please submit the entire project Folder**

Lab Instructions:

1. Create and design a program called MegaCalc! which should be a simple calculator that has a front-end graphical user interface (GUI).
2. Design the window of size 400x300 pixels.
3. Add two text fields in which the user types the operands to a binary + operation.
4. Next add the '+' button

   Note: When the user presses the the program treats the text in the two fields as two integers, adds them together

5. Finally, the resulting integer should be displayed on the screen as the text of a result label.



Submission Guidelines:

Please upload the following files onto Gradescope:

Java

Controller.java
Main.java
Calc.java
Lab12.fxml

C#

Form1.cs,
Form1.Designer.cs,
Program.cs
CalculatorClass (that which implements the interface)

Please follow the posted submission guidelines here:
https://ccse.kennesaw.edu/fye/submissionguidelines.php

Ensure you submit before the deadline listed on the lab schedule for CSE1322L here:
https://ccse.kennesaw.edu/fye/courseschedules.php