Lab 8: GUI

UNK CSIT 150 Object Oriented Programming

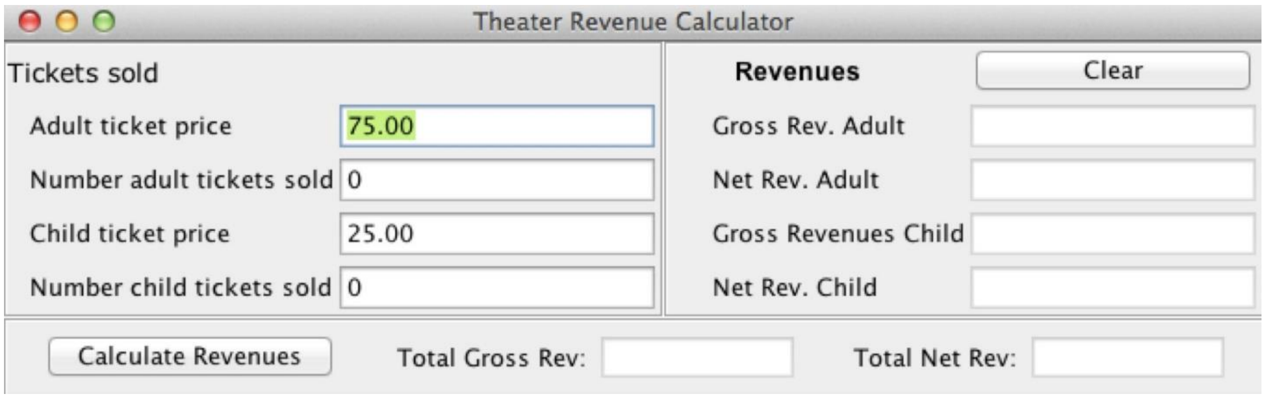
Objectives

- Familiarize yourself with designing & implementing a solution in GUI form
- Exercise good programming style
- Familiarize yourself with the Button listeners.
- Introduce programming of Data Validation with form, featuring intelligent error messages.

A sample application has been implemented as an example of a GUI interface that separates into two classes, the functionality of displaying data, with the actual class responsible for performing the logic of the program.

1. Download the two source code files and put them into a Java package named Lab8:
   - TheaterRevenues
   - TheaterRevenueWindow

2. Compile and run. A Window such as shown below should appear.



Tickets sold

Adult ticket price	75.00
Number adult tickets sold	0
Child ticket price	25.00
Number child tickets sold	0

Revenues

Gross Rev. Adult	
Net Rev. Adult	
Gross Revenues Child	
Net Rev. Child	
Total Gross Rev:	
Total Net Rev:	

Before we dissect the details of the lab assignment, we will discuss the responsibilities in general, for each class:

- **TheaterRevenueWindow**: This program's overall responsibility is to handle anything to do with the display (**view**) of data, entry of the data and the user interface.
- **TheaterRevenues:** This program's overall responsibility is to handle all the theater data (**model**), along with its data calculations, such as the net and gross incomes for both adult and child ticket sales, given the ticket prices and the number of tickets. This class does not need modified for this lab assignment.
- **MyListener:** This class is a private internal class within TheaterRevenueWindow. This class provides the **controller.** It responds to the user's actions. Because it is a private class within the TheaterRevenueWindow class, it has access to all of the TheaterRevenueWindow's data members and methods.

Note: It is important to separate out the display of the data with the logic. Model-View-Control software development architecture.

There are several features to this application to be noticed. The first feature to be discussed are the event listeners. These are added to the code in the method **addButtonListener**(). This method creates an instance of the MyListener class and adds it to individual components we want to "listen" to. At this point, we are listening to two JButtons. It should be noticed that the actions to be performed for each of the individual buttons is determine by the method call within the **actionPerformed**() method of the new listener. We have discussed the action listener previously so the code to create one should be familiar to you.

3. Make the program to respond to user action, by completing the following steps:

- Complete the actionPerformed() method, such that:
- When the user clicks the clearButton, **call the clearAllFields() method** and put the values back to the default values.
- When the user clicks the calcButton:
    - **call the validateFields() method** and proceed to the calculations if all fields are all valid. The data validation methods have been provided for you for this lab.

      Look at the validate methods to determine what they do. Use these methods as samples for data validation on future labs/assignments. Below are some things to consider for input validation.

      1) So, what is a valid field? That is entirely dependent on the actual application and deciding what constitutes a valid field is all part of the design process.

      2) Is the data type restrictive? For instance in this application when it asks for the number of tickets sold, does it make sense to enter 10 ½ tickets or should the number be restricted to positive whole number? Since integers are more restrictive that decimal numbers, the field would have to be checked for both letters and decimal numbers. Should negative numbers be allowed? With numbers these decisions are fairly straight forward, but with text, the best choices are not so obvious. What range of values are acceptable? If there is a maximum value, that should be validated as well.

      3) Is it critical that there be a default value or can the field be empty? What happened if no adult tickets were sold? Does a 0 have to be entered or can the field be zero?

    - **call all of the appropriate calculation methods**. You will need to write these methods as well (they are stubs at this time). In these methods, you will need to set the theaterRev object values appropriately, and set the JTextField values appropriately. For example:

      Call this method to set the theaterRev object's adult price:

      ```
      theaterRev.setAdultPrice(Double.parseDouble(pricePerAdultTxt.getText()));
      ```

      Call this method to set the JTextField for the net child revenue:

      ```
      netChildRevTxt.setText(Double.toString(theaterRev.calcNetChildSales()));
      ```

4. Run and test the program to verify that it works correctly.

5. Add a Menu Bar. Add two menus to the menu bar: File and Help. In the File menu, add two menu items, About and Exit. If the user selects "About", open a dialog box that displays information about this program. If the user selects "Exit", close the program. If the user selects the "Help" menu, open a dialog box that displays helpful information for this program. (See the gui.zip example files.)

6. Bonus: Remove the buttons, and move the control of user actions to menu items. You can choose the appropriate menus and menu items to add to do this.

Lab 8: GUI

Name(s): ____

Evaluation

Requirement	**Possible Points**	**Points Received**	**Comments**
calcButton	1		
actionPerformed	1		
validateFields	1		
the calculation methods	3		
clearButton/ clearAllFields	1		
Menu Bar, menu items	3		

Programming style	**Possible Points**	**Points deducted**	
Inconsistent indentation	**-1**		
Poor use of white space	**-1**		
Heading documentation, includes programmer names, date, algorithm, basic purpose	**-2**		
All sources not cited (Remember to cite all code used, even class demo code.)	**-1**		
JavaDocs not used on each method	**-1**		
Poor variable names	**-1**		
Poor structure/logic issues	**-2**		
Program specifications			
Bonus:			