Graphics User Defined Forms, Part I

### Quick Start

<table>
<thead>
<tr>
<th>Compile step</th>
<th>Execute step</th>
<th>Submit step</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>mkdir labs</code></td>
<td><code>cd labs</code></td>
<td></td>
</tr>
<tr>
<td><code>mkdir 4</code></td>
<td><code>cd 4</code></td>
<td></td>
</tr>
<tr>
<td><code>cp /samples/csc/156/labs/4/* .</code></td>
<td><code>cp PropertyTax1.java PropertyTax4.java</code></td>
<td><code>emacs PropertyTax4.java &amp; submit csc156 abc 4</code></td>
</tr>
</tbody>
</table>

### Compile step

```bash
mkdir labs
javac PropertyTax4.java
```

### Execute step

```bash
cd 4
java PropertyTax4
```

### Submit step

```bash
emacs PropertyTax4.java & submit csc156 abc 4
```

### Problem Statement and Specifications

These next two programming assignments will be in two parts. The first part of each will focus on the development of a graphic user interface (GUI) that generates a form for the user to interact with. Human Computer Interaction (HCI) is often regarded as the intersection of computer science, behavioral science, design and additional fields of study. It occurs at the user interface and the Java language has a number of tools available to use graphics in a GUI. The second part of the next two exercises will survey the use of methods and introduce you to classes in the Java language. Consequently, there will not be a significant amount of algorithmic development in the first parts of these exercises as the GUI design does have some requirements that some objects be generated before others, but the dependency relationships that we’ll pursue will not be as complicated as the other software tools that we’ve pursued up to this point.

**Assignment 4 Statement:**
Write a program in a file named `PropertyTax4.java` that will display the form shown in [Figure 1](#) for the user to enter data in the appropriate fields to calculate local property and school district taxes. None of the `JTextField`s nor `JButton`s need to work for this assignment, but the display with the appropriate labels must be identical to [Figure 1](#).

**Assignment 4 Specifications:**
- **Set** `JFrame` parameters to determine window
- **Output** `JLabels` to identify `JTextField` I/O opportunities
- **Output** `JButton` labels to identify event options

### Object Analysis and Algorithmic Development

When Java was first introduced, it supported an Abstract Window Toolkit to manipulate graphics that would provide portability for GUI applications, but was rather challenging to use. The Java Swing Toolkit
was designed to provide a less challenging interface to the more popular GUI elements. These objects are best used within a \texttt{JFrame} that can be interpreted as a window with specified size and banner title. The contents of \texttt{PropertyTax1.java} already have these lines coded within the file so we only need to add lines to that file. Let’s examine the contents of Figure 2.

The \texttt{import} statement in \texttt{green} makes the various \texttt{Swing} objects available to us. We’ll describe those in more detail below. The class \texttt{PropertyTax1} in \texttt{magenta} is declared as a subclass of \texttt{JFrame}. Object-oriented languages use an inheritance mechanism so that sub-classes can be created from existing classes. The sub-classes inherit available data attributes and methods that function within the ancestor or super class. For this exercise, we simply need to know that the \texttt{JFrame} sub-class (or \texttt{PropertyTax1}) can control aspects of the window that we’ll be displaying our information in. The next statement declares symbolic constants \texttt{WIDTH} and \texttt{HEIGHT} within the class, but outside any method. We’ll recommend that any other variables and constants be declared in this area.

Then comes the declaration of a special method known as a \texttt{constructor} in black. This method is how we can create instances of a class. Note that it has the same name as the class name (which is the same name as the file name) and has no return value. This is typical of a constructor, but it is not required that constructors do not have parameters as this example shows. All of the code that we’ll be writing for this exercise will occur in the constructor above the lines that are already there. The lines that are in the constructor 1) determine the title of the window that will be displayed in the banner with the \texttt{setTitle()} method, 2) determine the size of the window with the \texttt{setSize()} method, 3) establish the visibility of the window with the \texttt{setVisible()} method and 4) determine that clicking on the boxed \texttt{x} in the upper right-hand corner of the window will close the window. After the constructor comes the \texttt{main} method in \texttt{cyan} which simply invokes the constructor to instantiate an object of type \texttt{PropertyTax1} and store it in a variable named \texttt{proptax}.

Examining the contents of Figure 1 we can see that it is organized into 7 rows where each row has 2 columns. The last row is easiest to discuss in that both objects in the last row are \texttt{JButton} objects. \texttt{JButtons} have labels that provide directions to the user indicating what will happen when the user uses the mouse to click on the \texttt{JButton}. In this manner, the \texttt{JButton} is an output object as it directs the user what will happen if the \texttt{JButton} is clicked and it is also an input object as it is only activated if the user uses the mouse and clicks on it. The six objects in the left-hand column above the \texttt{JButton} are \texttt{JLabel} objects. These are simply output objects that are used to describe other objects that are near them. Each \texttt{JLabel} object in the left-hand column of the first six rows corresponds to a \texttt{JTextField} object in the right-hand column of the first six rows. Each of the \texttt{JTextField} objects can be used for textual input and output either by the user or the program. In this particular application, the top three \texttt{JTextField} objects are intended for the user to enter information regarding a home’s assessed value, regional school tax rate and county tax rate. The bottom three \texttt{JTextField} objects are used for the calculated school taxes, county taxes and total taxes based upon the input that the user had
import javax.swing.*;

public class PropertyTax1 extends JFrame
{

private static final int WIDTH = 400, HEIGHT = 300;

public PropertyTax1()
{
    setTitle("Calculation of Property Taxes");
    setSize(WIDTH, HEIGHT);
    setVisible(true);
    setDefaultCloseOperation(EXIT_ON_CLOSE);
} // end of the constructor method

public static void main(String[] args)
{
    PropertyTax1 proptax = new PropertyTax1();
} // end of the main method
} // end of the class

Figure 2: Lab 4 JFrame Shell

entered in the top three JTextField areas.
We’ll identify those objects that we need as we describe our algorithm for this program (note that this algorithm will accommodate the next two assignments.) Let’s start where we always have, with the IPO model.

1. Input
2. Process
3. Output

As was the case with the JOptionPane methods, all of the input and output will be interpreted as String values, so we’ll need to utilize some of the same numeric conversion and formatting methods that we did in previous exercises.

1. display the form and wait for the user to create a JButton event
2. process if the user clicks on the exit JButton
   (a) process to shut down the window
3. process if the user clicks on the calculate JButton
   (a) input the 3 String values from the JTextField objects that represent the assessed value, the regional school tax rate and the county tax rate
   (b) process to convert the 3 JTextField String objects to double
   (c) process to calculate the regional school tax, county tax and total tax
   (d) display those three tax values in the appropriate JTextField objects
CSC 156 - Assignment 4 PropertyTax

To clarify, step 2.(a) only occurs if the exit JButton is clicked. Likewise, steps 3.(a), 3.(b), 3.(c) and 3.(d) will only execute if the calculate JButton is clicked. The tax values will be recalculated each time that the calculate JButton is clicked on. Of course, once the exit JButton is clicked on, the window shuts down. This algorithm will be explored more fully in the next assignment. Our goal for this lab is to discuss step 1, how to create the form and display it. Line that are displayed in blue represent areas where code will need to be written or modified to complete the assignment.

### Pseudo code for Creating and Displaying the GUI Form

<table>
<thead>
<tr>
<th>Statement</th>
<th>Data objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) declare &amp; initialize 6 labels</td>
<td>JLabel, String &amp; Swing.Constants objects</td>
</tr>
<tr>
<td>2) declare &amp; initialize 6 text I/O fields</td>
<td>JTextField &amp; int objects</td>
</tr>
<tr>
<td>3) declare &amp; initialize 2 button objects</td>
<td>JButton &amp; String objects</td>
</tr>
<tr>
<td>4) declare &amp; initialize a Container</td>
<td>Container object &amp; getContentPane() method</td>
</tr>
<tr>
<td>5) configure the Container</td>
<td>Container.setLayout() method, GridLayout &amp; int objects</td>
</tr>
<tr>
<td>6) place GUI objects in the Container</td>
<td>JLabel, JTextField, JButton objects and Container.add() method</td>
</tr>
<tr>
<td>7) set the dimensions of the window</td>
<td>setSize() method &amp; int objects</td>
</tr>
<tr>
<td>8) set the title of the window</td>
<td>setTitle() method &amp; String object</td>
</tr>
<tr>
<td>9) set the visibility of the window</td>
<td>setVisible() method &amp; boolean object</td>
</tr>
<tr>
<td>10) set the default close operation</td>
<td>setDefaultCloseOperation() method &amp; int objects</td>
</tr>
</tbody>
</table>

### Coding for Compilation

Logon to your csc.oakton.edu account and create a new subdirectory of the labs directory called 4 to work on your assignment.

```bash
cd labs
mkdir 4
cd 4
```

A shell of a main program has already been included in the file /samples/csc/156/labs/4/PropertyTax1.java that you can copy over to your local directory by issuing the following commands.

```bash
cp /samples/csc/156/labs/4/* .
cp PropertyTax1.java PropertyTax4.java
eemacs PropertyTax4.java &
```

The contents of this file appear in Figure 2. After you change every occurrence of PropertyTax1 to PropertyTax4, compile and execute your program with the following commands.

```bash
javac PropertyTax4.java
java PropertyTax4
```

The output of this should match the window shown in Figure 3.

### Declaration of Variables

The scope of a symbol is the set of lines that can access that symbol. To maximize scope, it will make sense in this instance to declare the variables within the class, but outside the constructor method.
Step 1
// declare and initialize 6 JLabels
There are 6 JLabel fields that we must accommodate and the text that they display should be right-justified. So, in the area within the class we could write the following declaration.

    JLabel assessL = new JLabel("Assessment Home Value", SwingConstants.RIGHT);

Do this for the other 5 JLabel objects as well.

Step 2
// declare and initialize 6 JTextFields
In the area where you’ve declared the JLabel objects, declare the JTextField objects in a manner similar to below.

    JTextField assessTF = new JTextField(10);

Similarly, you’ll need to do this for 5 additional JTextFields.

Step 3
// declare and initialize 2 JButtons
Once again in the area where the JLabel and the JTextField declarations are, place the following declaration.

    JButton exit = new JButton("Exit");

Similarly, you’ll need to do this for a JButton that should be named calculate.

Steps 4-5
// declare and initialize a Container
// set the container layout
These are statements that do not rely upon the new operator. They will retrieve the content pane and introduce a design pattern so that we can build the GUI. Consequently, it is inappropriate to place this code above the
constructor, so we'll place it within the constructor, near the top. It is important to note that anything that we intend to display must be executed before the setVisible(true) command is executed. In order to use the Container class, it will be necessary to place a import java.awt.*; statement at the top of your file. It is the GridLayout() method invocation that creates the 7 rows by 2 columns pattern.

```java
Container pane = getContentPane();
pane.setLayout(new GridLayout(7,2));
```

**Step 6**

// set GUI objects in the container

This is simply the execution of the following command repeatedly for each GUI object.

```
pane.add(assessL);
```

Each pair of the repetitions of the add() method will fill a row. When you have added 8 of the 14 GUI objects to be displayed, we might verify that our GUI construction is proceeding in the manner that we expect.

**Checkpoint**

Although we're almost done, at this point, it might be a good idea to compile this much before we build more. Save your program to disk by choosing the Save command from the Files menu of your emacs session. Then, compile your program by choosing the Compile... command from the Tools menu and change the make -k that is displayed to javac PropertyTax4.java. Compiler errors can be parsed with the keystroke C-x ' and need to be repaired before your program can execute. When your program has compiled, click on your xterm window to access your command line prompt, and issue the command java PropertyTax4.

The output displayed should look in the same manner as Figure 4. Once you are sure that you're on the right track, you should use the add() method to place the remaining objects into the content pane.

**Steps 7-10**

// set the dimensions of the window
// set the title of the window
// set the visibility of the window  
// set the default close operation.

These statements were already in the original PropertyTax1.java file. After you’ve loaded all of the GUI objects into the content pane, your output should look like that of Figure 1. Note that it may be necessary for you to halt your program using the ps and kill commands as you did in the last exercise.

Testing for errors

At this point, there is not a great amount that you can do with your program as it only displays the GUI. We’ll have more opportunity to test this code in the next exercise. There is a correctly functioning version of the program for PropertyTax4.java at this link.

Printing and submitting

Once you are satisfied with the correctness of your program, print it as you did with previous assignments by using the following command that assumes that you are working in the room 1234 at Oakton. Retrieve your copy from the printer.

    printer 1234 PropertyTax4.java

Finally, submit your program with the following command that assumes that you are registered in section abc of CSC 156.

    submit csc156 abc 4