Matthew Lonis

Jeff Whitmer

CSCI-A 290

December 5th, 2016

***MINI 3***

**Topic 1: Form Communication**

**3 URLs**

1. <http://stackoverflow.com/questions/4438654/best-way-to-communicate-between-forms>
2. <http://stackoverflow.com/questions/1665533/communicate-between-two-windows-forms-in-c-sharp>
3. <http://stackoverflow.com/questions/27855243/communicating-between-two-forms-in-c-sharp>

**Precis of Sources**

All URL’s describe different, yet similar methods for communication between a parent and a child form. Two main suggestions were offered. The first option was to pass specific data between the forms by including it as a parameter in the creation of the child form. This method allows you to send rudimentary data such as strings, integers, etc. but is inherently limited. The second options presented is far more useful. Showcased in URL #2, the entire parent form is passed as a parameter in the initialization of the child form. In the child form code, a variable has to be set to be equal to the passed for and is ‘called as’ the form class name.

**Summary of Topic**

URL #2 is the URL that showed me how to add communication between two forms. The benefit of this method is that you can access anything that is public in the parent form from the child form. I’ve implemented this as far as allowing a button in a child form to cause a button click event in the parent form. The benefit of this method can be easily explained by referring to the options form in the A290 Buffet project. By using this method, allowing the options form to communicate with the parent form, once you change the background color in the options form, you can set it to also change the background color in the parent form; and it updates it live without having to close the child form. This is also useful for accessing or changing variables in the parent form that the user will change in the child form, especially if the variables need to be accessed by future child forms. For example, I set a color variable in my main form so that when a child form is initialized, the child form sets its BackColor to be the same as the parent form (useful when you change a BackColor in an options page and then go to open another page).

**JIT 3**

This is important and useful to know, because form communication is essential for a fluid and dynamic app.

**Topic 2: Dynamic ComboBox**

**3 URLs**

1. <https://msdn.microsoft.com/en-us/library/aa983551(v=vs.71).aspx>
2. <http://stackoverflow.com/questions/2152705/how-to-dynamically-add-combobox-in-windows-formsc-and-bound-it-to-a-column-of>
3. <http://stackoverflow.com/questions/3063320/combobox-adding-text-and-value-to-an-item-no-binding-source>

**Precis of Sources**

The URLs all describe ways to add or remove items from a ComboBox (CB) in C#. There are several different ways to add items to a CB. You can simply just add a string with the .Add() method, you can insert a string at a desired location in the CB using the .Insert() method, or you can assign an entire array to a CB using the .AddRange() method. Removing items from a CB are similar to adding items. To remove an item with index [0] use the .RemoveAt(0), to remove the currently selected item use the .Remove(comboBox1.SelectedItem), and to remove a specific string use .Remove(“string”) method. There is also a usefull .Clear() method which will clear out the entire contents of the CB with one method.

**Summary of Topic**

In C#, there are built in methods to inset or remove ComboBox items. The methods for adding items to the CB are Add(), Insert(), and AddRange(). To remove items from a combo box the methods are RemoveAt(), Remove(), and Clear().

**JIT 3**

This is a useful thing to know because this opens the door to creating dynamic ComboBox dropdowns that can respond and change based on variables, user input, or events.

**Topic 3: Custom Classes & Structures**

**3 URLs**

1. <https://msdn.microsoft.com/en-us/library/x9afc042.aspx>
2. <https://www.tutorialspoint.com/csharp/csharp_struct.htm>
3. <http://stackoverflow.com/questions/521298/when-to-use-struct>

**Precis of Sources**

Classes in C# are declared in a very similar way to that as seen in Java. The class keyword is necessary to create the class and in order to create an object, one must use the new keyword followed by the class name with parentheses for arguments. Classes can inherit from other classes and to do this one must simply put a semicolon next to the class name followed byt the class it is inheriting from. Classes differ from C# structs because a class can be used to make objects and have associated methods with that object. A struct on the other hand is simple a way to define a variable that contains variables of different types within it. Structures are defined very similarly to C or Java structs and uses the struct keyword.

**Summary of Topic**

The difference between classes and structs are important to know. Classes are like structures in some ways but they can make objects and have associated methods with those objects. A struct on the other hand is more like a convenient way to hold multiple related variables in one variable. No methods are associated with a struct unlike a class.

**JIT 3**

This is useful to note because one doesn’t want to use a class when they could easily use a struct and vice versa. Data size of the variables are also important and knowing how to pass each type into a function is useful as well.