**01.1205.OOP2 :: Orville's Oldtime Pop (OOP) :: Phase 2**

## .1) Assigning Variables (Text Boxes) ::

OK, you now have a Screen with Controls accessible via their Unique IDs (**IDC_** stands for ID, Control).

Our next step is to create Variables for the appropriate controls. Note that we will not be creating variables for EVERY control since as you'll soon see there is a relationship between sets of controls.

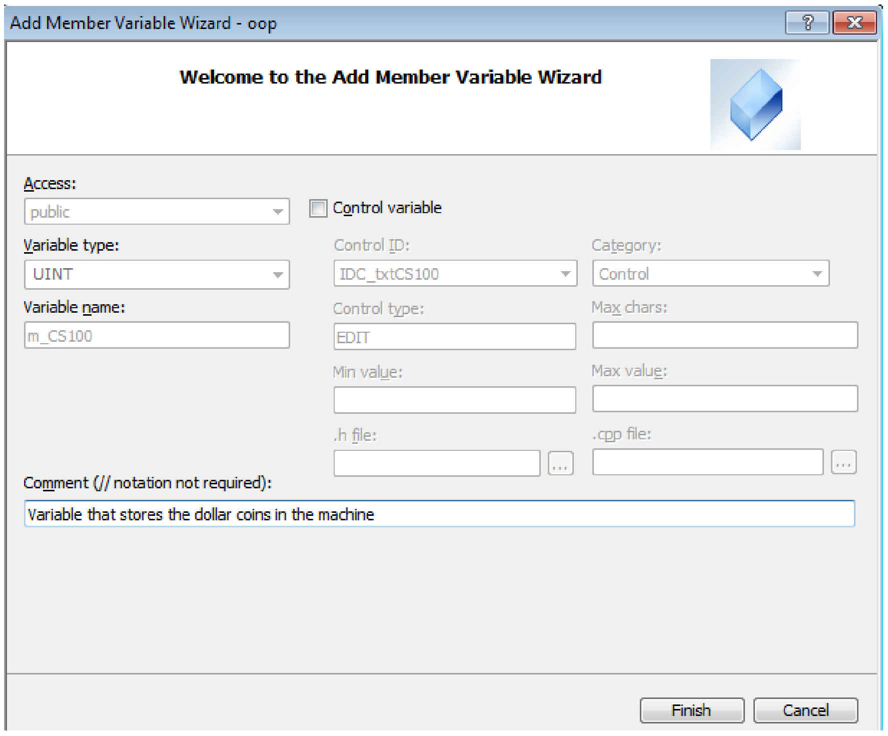
To begin with, we will create a variable for each of the ***Text Boxes*** (Coin Stacks, Coins Entered, Dispensers, Dispense Slot, and Coin Return boxes). Don't worry about Pop Price , History , and Service Check Box for now!

In all, you should be creating 20 variables... double check your screen layout with your instructor if you're confused. START BY filling out the paperwork provided earlier, and have the instructor check that page before continuing.

Click on the ***ResourceView*** tab, expand the ***Dialog*** folder, and open **IDD_???OOP_DIALOG** . (where ??? should be your last name)

Select one of your ***Text Boxes***, right click on it, and select ***Add Variable...***

You should then get the following Dialog.


Add Member Variable Wizard - oop
Welcome to the Add Member Variable Wizard
Finish
Cancel

Access: public

Control variable

Variable type: UINT

Control ID: IDC_txtCS100

Category: Control

Variable name: m_CS100

Control type: EDIT

Max chars:

Min value:

Max value:

.h file:

.cpp file:

Comment (// notation not required): Variable that stores the dollar coins in the machine

Select an appropriate ***Member Variable Name***... the **m_** denotes it is a **Class Member Variable** so keep that. Other than that, be appropriate!!! (hint... if I was setting up a variable for **IDC_txtCoinStack100**, I'd probably call it **m_txtCoinStack100** ...but you're free to keep track of as many different names as you wish!)

***Category*** will be **Value** for everything we do here.

Pay attention to what you select as ***Variable type*** !!!

Most of our variables will be **UINT** (Unsigned Int), but the $ fields will be **double** and the Dispense Slot will be **CString**. (please recall you are in a case sensitive language).

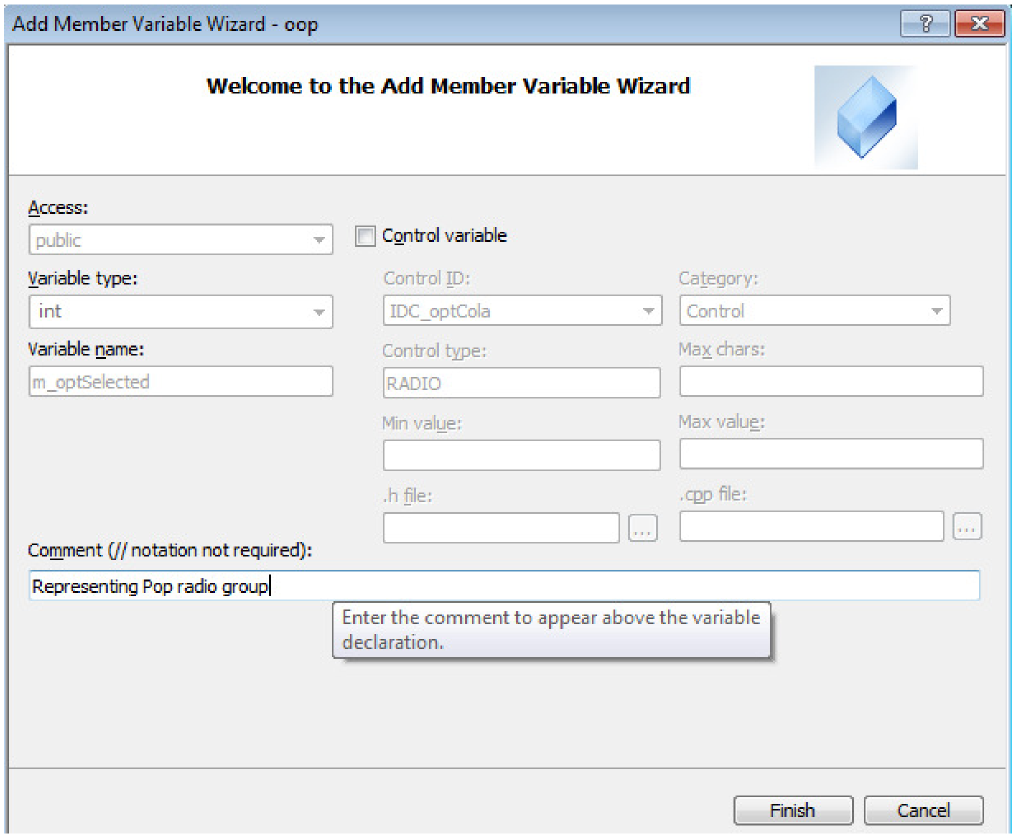
OK, set up 20 Variables for the 20 Edit Boxes mentioned above.

## .2) Assigning Variables (Radio Buttons) ::

***Radio Buttons*** are rather different from ***Text Boxes***. Whereas ***Text Boxes*** each have separate and distinct variables assigned to them, ***Radio Buttons*** share a variable between ***Groups***.

***Groups*** are also how VC++ handles the situation of distinguishing between the different arrangements of ***Radio Buttons***.

To create a ***Group*** , click on the first button in your set of ***Radio Buttons***. Set the ***Group Property*** of that button to TRUE. All the other radio buttons should have Group Property set to FALSE.


Add Member Variable Wizard - oop

Welcome to the Add Member Variable Wizard

Access: public

Control variable

Variable type: int

Variable name: m_optSelected

Control ID: IDC_optCola

Category: Control

Control type: RADIO

Max chars:

Min value:

Max value:

.h file:

.cpp file:

Comment (// notation not required): Representing Pop radio group

Enter the comment to appear above the variable declaration.

If you had other sets of radio buttons, you would repeat the process, setting the Group Property of the first button of each set to TRUE.

To assign a single variable to a ***Group*** of ***Radio Buttons***, right click on the first ***Radio Button*** in a particular arrangement, and select ***Add Variable...***

See image for typical settings. You "set" the ***Group*** by checking ***Control Variable***. In the example shown, **m_optSelected** would be the single variable attached to ALL radio buttons that follow, UNTIL it hits another ***Group***. Also note that int is selected as variable type… you would select that instead of using the BOOL type the wizard may suggest.

For future reference, this variable now has a unique relationship with the buttons. Setting it to 0, 1, 2, 3, etc. will cause the first, second, third, etc. button is be "set". Conversely, querying the variable will enable you to determine the ***Group State*** (which button is selected).

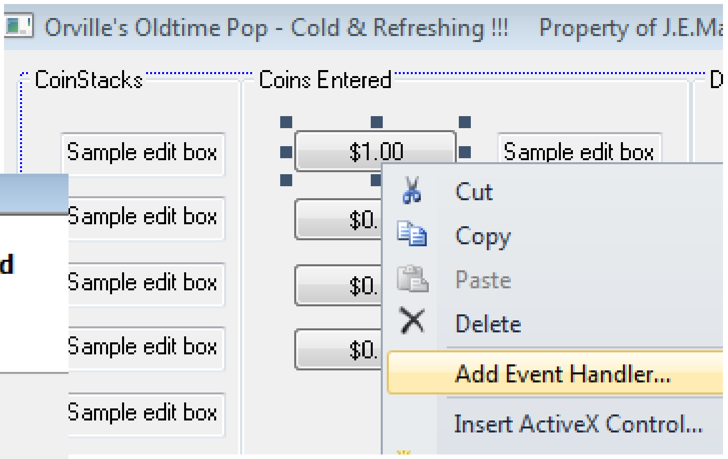
## .3) Generating a Function for a Control & Adding Appropriate Code ::

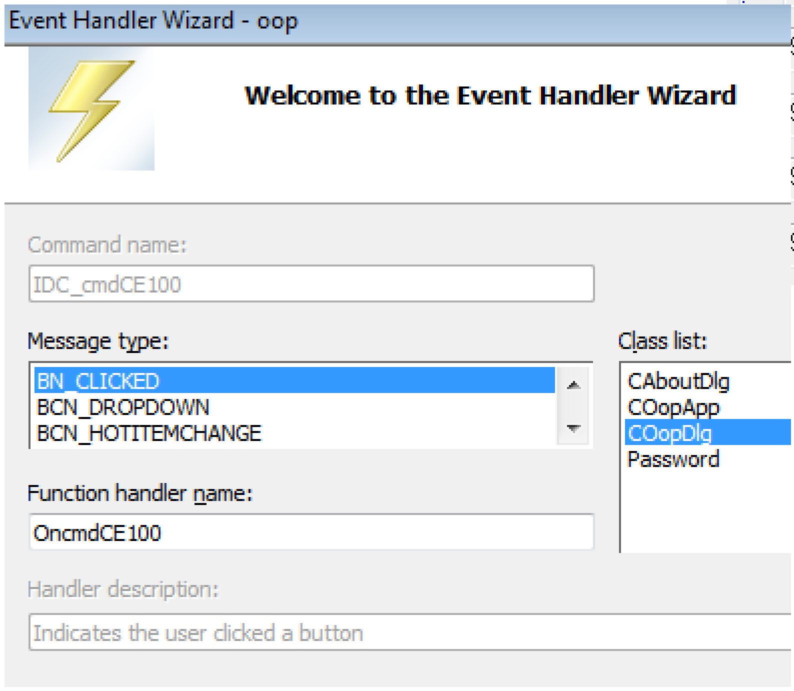
Ok... we're getting close to writing some code! But, the ***IDE (integrated Development Environment)*** will take care of generating all the appropriate Function mechanisms for us first (ie making sure it's in the Class Definition etc.).

If you have become familiar with the working model (LIKE YOU WERE SUPPOSED TO), you'll realize there are relationships between the **Command Buttons** and **Text Boxes** on the Screen. For example, clicking the **$1.00 Button** causes the variable in the **Text Box** next to it to increment.

On this page, I will show you specifically how that was accomplished.

Select the **$1.00 Button** by right-clicking and selecting ***Add Event Handler*** (hope you're catching on to this two-step... I'm going to stop laying it out for you!!!).


Orville's Oldtime Pop - Cold & Refreshing !!! Property of J.E.Ma
CoinStacks
Coins Entered
Sample edit box
$1.00
$0.
Cut
Copy
Paste
Delete
Add Event Handler...
Insert ActiveX Control...


Event Handler Wizard - oop
Welcome to the Event Handler Wizard

Command name:
IDC_cmdCE100

Message type:
BN_CLICKED
BCN_DROPDOWN
BCN_HOTITEMCHANGE

Class list:
CAboutDlg
COopApp
COopDlg
Password

Function handler name:
OncmdCE100

Handler description:
Indicates the user clicked a button

Highlight the **BN_CLICKED** selection. Then, click on ***Add and Edit*** , and voila, that has become a Class Method!

Unfortunately, it doesn't contain code! Sorry, time to put your programming training to use!

Since I don't know what you called your variables, I can't tell you EXACTLY what to add, but it should look something like...

```
m_txtEntered100++;
```

"Basically", clicking the **$1.00 Button** should update the variable attached to the **Edit Box** next to it. Understand? Keep reading to Step 2.4…

## .4) Pushing the Contents of Variables out to Controls ::

Build and run your program! Once you get through your compile errors, you should have a program that runs, but appears to do nothing with you click on any of the buttons ( !? )... whoops.

No fear... you haven't done anything wrong, I'm just illustrating a point! In this GUI environment, you have to keep track of what you're doing (ie manipulating your variable), but then you have to "tell" Windows to "display" your work!!! Ouch... is that hard? No, just a single line of code in the same function...

```
UpdateData(FALSE);
```

....repaints the existing screen!

Although in the last part of Step 2.3 you updated a variable, you need to take the additional step of pushing the new value out to the actual control.

OK, rebuild again and see the fruits of your labor!!! Proud of yourself? Good, do it a total of 16 times to handle the relationships between the 4 "Coins Entered" buttons, and the 12 "Service Machine" buttons.

<u>DON'T WORRY ABOUT</u> the $ total fields which are also affected by certain buttons... we'll handle those next!!!

Have fun (BTW, test thoroughly… careful about "unloading" the CoinStacks (hint, hint)).

## .Z) Checkpoint ::

*At this point, you should be comfortable with generating a Dialog Based Windows project with a complex screen design. As well, you should be familiar with how to assign the IDs of the various controls*, **assign appropriate variables to Text Boxes, generate Class Methods for your Command Buttons, and finally add rudimentary code to manipulate and display those variables .**