IVY TECH COMMUNITY COLLEGE

**SDEV 140 / Introduction to Software Development**

**Software Development Project – Part 3**

# Student’s name Blair, William Section M08 Instructor’s name Carver, Steve

If you’re using Visual Studio Community 2015, as requested, the instructions below should be exact, but there may be minor discrepancies that require you to adjust. If you are attempting this assignment using another version of Visual Studio, you can expect differences in the look, feel, and/or step-by-step instructions below and you’ll have to determine the equivalent actions or operations for your version on your own.

**INTRODUCTION**: In this assignment, you will expand the DroneDogs program in response to a Project Enhancement Request submitted by DroneDogs.

You have received the following request from your client, the owners of DroneDogs, as part of your ongoing contract to provide system upgrades and maintenance. An edited version of the ‘visual details’ that DroneDogs refers to in their request comes later in this assignment.

We love the new ordering system and the operation of the Order Input form, but would now like to add a Customer Information form to the system. The Customer Information form will display existing customers (first name, last name, and email address) and allow the user to add new customers. Users can select a new or existing customer, and the first name, last name and email address of the selected customer will be automatically transferred back to the Order Input form before the order is submitted.

In specific, we are requesting that you (see figures below for visual details):

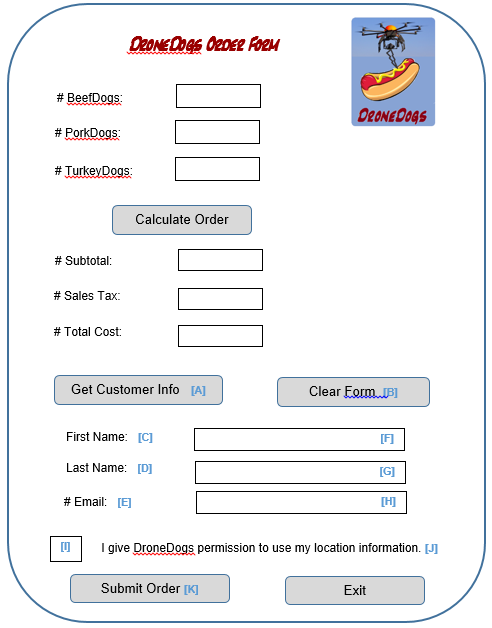
1. Add a Get Customer Info button to the Order Input form. Clicking that button should open the new Customer Information form.
2. Add the fields from the Customer Information form to the Order Input form, so that the information can be transferred there after a customer is selected from the Customer Information form.
3. Add a Clear Form button to the Order Input form, which will clear all the text boxes when the user is ready to add a new order.
4. Add a check box to the Order Input form, allowing DroneDogs to use location services to determine where the drone will deliver the order.
5. Modify the code for the Submit Order button. When the user clicks this button, the program will check to see if:

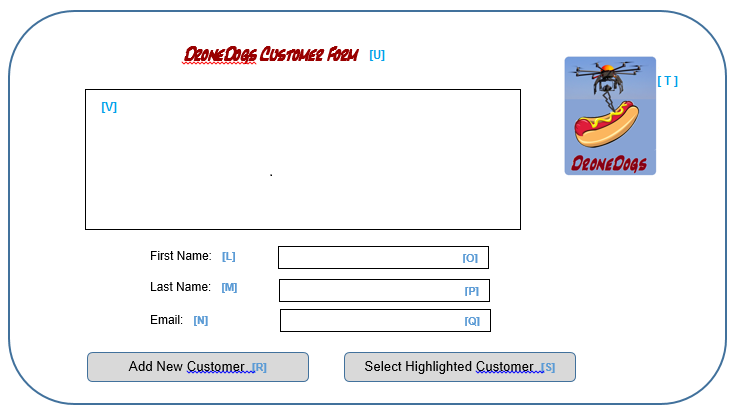
a) the permission check box is checked,   
b) there is something in the total cost text box, and   
c) there is something in the customer email text box.

If any of these are not filled in, the program displays an appropriate error message. If they are all OK, a message box is displayed thanking the user for placing the order.

**Notes to Students:**

All new items are labeled below – old items should already exist in your project from previous work. The small box at the bottom of the Order Form is a CheckBox (the text is a property of the CheckBox, not a label), and the large box near the top of the Customer Form is a ListBox. Also, you can use the same names for the First Name, Last Name and Email controls on both forms.





**ASSIGNMENT PART I:**

**Enter the control names you would assign to all new items, using the same Hungarian Notation we’ve been using:**

**A: btnGetCustomerInfo**

**B: btnFormClear**

**C: and L: lblFirstName /**

**D: and M: lblLastName /**

**E: and N: lblEmail /**

**F: and O: txtFirstName /**

**G: and P: txtLastName /**

**H: and Q: txtEmail /**

**I: and J: (part of same control) chkConsent**

**K: btnSubmitOrder**

**R: btnAddNewCustomer**

**S: btnSelectHighlightedCustomer**

**T:**

**U:**

**V:**

**ALGORITHM DEVELOPMENT DISCUSSION:** Now is the time we’ll implement the work requested by DroneDogs

You will need to create the Customer Form based on the visual template above. This form will have the same DroneDogs heading and logo as the order form, and will use the names you assigned earlier to the form components. This includes the new ListBox, which will hold the information for pre-loaded customers and new customers added by the user.

You will also need to make changes to the Order Form.

* 1. You will need to add a Clear Form button to the Order form – it will clear out all text boxes when clicked.
  2. You will need to add a CheckBox to the Order form - the user checks it to give DroneDogs permission to use location services to find the location of the user for delivering the order.
  3. You will need to revise the Submit Order button to check for three things: the permission check box is checked, there has been at least one item ordered, and customer information has been entered. If all these things are OK, then the message thanking the user can be displayed.
  4. You will add the labels and text boxes for the customer information, so that you can transfer the selected information from the Customer form to the Order form.

**ASSIGNMENT PART II**

1. Go to the folder containing your your DroneDogs project from the previous module. Copy that to a new folder (perhaps named SDEV140\_M13\_LastnameFirstname).
2. In Visual Studio, click File | Open and select your the DroneDogs.sln solution file from your new folder.
3. Bring up the Solution Explorer window. There may be a tab for it already visible near the Properties window, or you may need to click View / Solution Explorer.

You will create a structure to hold each customer. Structures are very similar to classes. In Solution Explorer, right click on the project name DroneDogs, and select Add / Class. Name the class **Customer.vb**, click Add, and in the code window, delete all the lines that are there, and create a structure by entering the following code, which defines a structure and provides a nicely formatted display for the list box. When finished, click the Save All button, or File | Save All through the menus.

Public Structure Customer

'Public members

Public FirstName As String

Public LastName As String

Public Email As String

'Formats the display

Public Overrides Function ToString() As String

Return FirstName & " " & LastName & " (" & Email & ")"

End Function

End Structure

1. Now you need to create the Customer form. Right-click the project name DroneDogs in the Solution Explorer, and select Add / Windows Form. Change the File Name of the form **CustomerForm.vb**, and click Add. Change the Text property of the form to **your** name. Use the Visual Studio Toolbox and the form/control names you made earlier in this lab to create the Customer Information form. Remember that the big box under the label for the heading is a ListBox control.
   1. Put labels, text boxes, buttons and a picture box onto the form in the appropriate locations. Change the Name property of each control to the name you assigned it earlier in the lab, and change the Text property of the labels and buttons to the appropriate text from the visual template shown earlier in this assignment.
   2. If you haven’t already, download the image file **DroneDogs.png** attached to this assignment. For the picture box, specify the Image as **DroneDogs.png** by browsing to the location where you downloaded it to and importing the file into the project.
   3. Click View / Code and make sure that you’re viewing the code in the DroneDogs Customer form (not the Order form) by checking the name in the title of the tab you are viewing. We are going to use an ArrayList to keep track of all the customers in the program. Just under the top line of the code (Public Class CustomerForm), enter the comment **'Form level member**, and on the next line, declare a new ArrayList member with this statement: **Private objCustomers as New ArrayList**
2. Go back to the DroneDogsOrder form (in Design view), enlarge the size of the form, and add the new buttons, labels, text boxes and a check box from the visual template in this lab.
   1. To code the Get Customer Info button, double-click it, enter an appropriate comment, and type this statement: **CustomerForm.Show()**
   2. Go back to Design view for the Order form, and code the Clear Form button by double-clicking it, which takes you back to Code view. Enter an appropriate comment, then set the Text property of EACH of the text boxes to an empty string.

Depending on what you named the text boxes, the first statement might be:  
**txtBeefDogs.Text = ""**

* 1. In Design view, change the Read-Only property of the customer text boxes (first name, last name, email) on the original Order Input form to True (these will be filled in from the Customer Information form, not by the user).
  2. To modify the code for the Submit Order button, double-click it, and in Code view, use a series of IF statements to test whether: the permission check box has been checked, there is something in the total cost text box, and there is something in the customer email text box.

Your code, after you enter your descriptive comments, should look similar to this (depending upon the names you used for your controls):

If chkPermission.Checked = False Then

MessageBox.Show("ERROR...You must check the location permission check box.")

ElseIf txtTotalCost.Text = "" Then

MessageBox.Show("ERROR...You must order at least one item.")

ElseIf txtEmail.Text = "" Then

MessageBox.Show("ERROR...Please get customer information for this order.")

Else

MessageBox.Show("Thank you for ordering from DroneDogs!")

End If

1. Now, go back to the new CustomerForm in Design view. To code the Add New Customer button, double-click it, and the code header will appear in the code window.
   1. Enter a comment that you will be getting the new customer information from the text boxes.
   2. You need to declare string variables to hold the customer’s first name, last name, and email, and initialize them to the Text values entered in the text boxes. Use Dim statements to declare the three string variables, and use the Text properties of the corresponding text boxes to initialize each of the string variables.

Depending on what you named your variables and text boxes, your first statement might be:  
**Dim strFirstName = txtFirstName.Text**

* 1. Next, you need to create a new Customer record from the information. The easiest way is to call a procedure named **CreateCustomer**, sending it the three string variables as arguments.(We’ll define that procedure shortly.) To call a procedure, just enter a descriptive comment about what you’re doing, then type the procedure name, and the arguments in parentheses.

Depending on what you named your variables this might be like:  
**CreateCustomer(strFirstName, strLastName, strEmail)**

1. Now, to define that new procedure, stay in the Code View window, but click the cursor UNDER the **End Sub** line for the Add Customer procedure you just finished. Since this procedure (Sub) is not associated with an event on a control, the ‘stub’ isn’t automatically created by you double-clicking on an item in the Design View. You’ll create the Sub in it’s entirety.

You will need to declare a new Customer object, assign the customer variables to its members, add the new customer to the ArrayList, and also display it in the ListBox on the form. The code is given to you here, but make sure that all names match the names you assigned (like the name of the ListBox. To create the procedure (Sub) you can type or paste this code:

Public Sub CreateCustomer(firstName As String, lastName As String, email As String)

'Declare a customer object

Dim objNewCustomer As Customer

'Create the new customer

objNewCustomer.FirstName = firstName

objNewCustomer.LastName = lastName

objNewCustomer.Email = email

'Add the new customer to the list

objCustomers.Add(objNewCustomer)

'Add the new customer to the ListBox control

lstCustomers.Items.Add(objNewCustomer)

End Sub

1. We’re getting closer! To make things easier, we are going to pre-load three customers into the Customer Form when it is first loaded. Go back to Design view for the CustomerForm, and double-click in any open area of the form. This should take you into code view for the form and into a procedure called **CustomerForm\_Load**. The Load event is automatically executed when the form is first loaded.

Enter a comment that you are pre-loading three customers to the form, and type in these three statements, between the procedure heading and the **End Sub** line:

CreateCustomer("Fred", "Garvin", "fgarvin@thiscompanysnl.com")

CreateCustomer("Fran", "Pepper", "fpepper@notthesoftdrink.org")

CreateCustomer("Will", "Robinson", "wrobinson@lostinspacetown.gov")

1. When the user highlights a customer from the ListBox, the index of that customer is stored in a variable called **SelectedIndex**. (If no customer is selected, the variable is set to -1.) We are going to need to provide for a way to capture the information from the highlighted customer. We can do that by creating a **property**. (You can read about properties for items in an ArrayList in the textbook or in numerous online sources – take the time to see if the logic below makes sense to you.) As long as a customer is selected, we can convert the information in the ListBox into a Customer structure.

Stay in code view, and underneath the **End Sub** line for the form load procedure, type or paste this code, changing control names to those you are using, as appropriate:

Public ReadOnly Property SelectedCustomer() As Customer

Get

If lstCustomers.SelectedIndex <> -1 Then

'Return this customer

Return CType(objCustomers(lstCustomers.SelectedIndex), Customer)

End If

End Get

End Property

1. When the user wants to select a customer by clicking the **Select Highlighted Customer** button, the selected customer’s information should be used to fill the text boxes back in the order form. Go back to Design view for the Customer form, and double-click the **Select Highlighted Customer** button. This should take you back into code view and into a procedure called **btnSelectCustomer\_Click** (depending on the name of your button). Enter the code below, which will show an error message if no customer was selected or will fill in the blanks on the order form if there is a selected customer. Make sure this code goes between the procedure heading and the **End Sub** line:

'If no customer is selected, then error and exit

If lstCustomers.SelectedIndex = -1 Then

'Display error message and exit

MessageBox.Show("ERROR...no customer selected.")

Exit Sub

End If

'Get customer info and display it in the order form

Dim objCustomerSelected As Customer = SelectedCustomer

DroneDogsOrder.txtFirstName.Text = objCustomerSelected.FirstName

DroneDogsOrder.txtLastName.Text = objCustomerSelected.LastName

DroneDogsOrder.txtEmail.Text = objCustomerSelected.Email

1. You are finally ready to compile and run this program. If you have errors, read the error messages closely. You can double-click on the error message, and you will be taken to the line of code that caused the error. When the program compiles successfully, click Debug, and Start Without Debugging to run the program.
   1. Enter these numbers to order: **4** beef dogs, **5** pork dogs, and **6** turkey dogs, then click Calculate Order, and ensure that the program correctly computes the subtotal, the amount of sales tax, and the total cost of the order.
   2. Next, click the **Get Customer Info** button. That should bring up the new Customer Information form (it may appear directly on top of the Order Form. Click on one of the three existing customers, and click **Select Highlighted Customer**. That should put the name and email of the selected customer back into the first form (you may have to drag the Customer Information form to the side to see the Order form.).
   3. Now, enter YOUR name and email address as a new customer into the text boxes on the Customer form. Click **Add New Customer** to verify that you’ve been correctly placed into the ListBox. Then, click on the yourself in the ListBox and click **Select Highlighted Customer** as before, and the new customer information should be placed back into the Order form.
2. On the Order form, click the permission check box and click **Submit** Order. If the customer information was transferred correctly and you checked the permission check box, you should get a message box thanking you for placing your order. If you get an error message, you will need to find your error, save the program, rebuild it, and run it again.
3. Note: You don’t have to close the customer form after each order, and in fact, you shouldn’t, because each time you reopen it, any new customers that you added will not reappear. (The customers are pre-loaded each time the form is opened.)
4. When the program works correctly, end your run by clicking the Exit button. Save your project and exit Visual Studio.

**Note**: There is a document in this Module named **VB3SampleCode.docx**, which shows some sample working program code. You can use this to help you design and debug your own program, but you cannot copy it directly into your code, because you don’t have text, you have images of the text. Also, the names for the components, constants and variables in the sample code may be different than what you are using in your lab and some of the other data (for example the three name strings in the default customer info list) are different. Finally, you MUST include comments in your code, which are not present in the sample code. This is just a guide.