**Introduction to Visual C# Programming**

Course Description:

Programmers start here. In this course, you’ll learn key programming concepts and then apply them using Microsoft® Visual C# ® 2015 Express/Community Edition, the free version of Microsoft’s Visual Studio ® developer toolkit. Complete a series of increasingly complex projects while you learn C#. You will learn two different skills: Designing Windows Forms Applications (the “Visual” part) and C# Programming Language and Syntax. At the end of this course you will have a portfolio of completed Visual C# executable programs.

Prerequisite:

No previous course prerequisites; this course is parallel to *Visual Basic*, so previous completion of *Visual Basic* will enrich the knowledge gained in the course. However, this course covers a lot of similar topics to *Visual Basic*, except in a more complex and rich language.

Skills: Moderate logical skills, some concepts from algebra/pre-algebra such as using equations, types of numbers (decimals, whole numbers, integers), and Boolean logic.

Difficulty Level:

Challenging. Mastery of prerequisites, familiarity with computers, or experience with another programming language will reduce difficulty significantly.

Inability to type/spell with accuracy will greatly harm a student’s ability to perform well in this course.

Picking the Class That is Right for You:

*Introduction to C# Programming* focuses on more complex concepts whereas *Visual Basic* focuses more on user interface design. *Introduction to C# Programming* has a few more “free form” projects, where the students will work by themselves than *Visual Basic.* Students will learn basic UI design (the “Visual” part of “Visual C#”) in this course, however some advanced concepts such as file input/output are not covered.

This course is a great supplement to a student who is planning on taking AP Computer Science or has already taken AP Computer Science.

Class Room Setup:

1. Install Visual Studio 2015, default settings work well.
2. Update Internet Explorer to Version 11.
3. Log into Account: [ABTC.Compucamp2016@outlook.com](mailto:ABTC.Compucamp2016@outlook.com) / Texas2016

Course Overview:

1. Day 1:
   1. Chapter 1
   2. Chapter 2
   3. Chapter 3
      1. Guided Project: Beep
2. Day 2:
   1. Chapter 4
      1. Guided Project: Form Fun
      2. Extra Project 1: Stop Watch
3. Day 3:
   1. Chapter 5
      1. Guided Project: Savings Account
4. Day 4:
   1. Chapter 6
      1. Guided Project: Guess the Number Game
      2. Extra Project 2: Times Tables
5. Day 5:
   1. Chapter 7
      1. Guided Project: Sandwich Maker
6. Day 6:
   1. Chapter 8
      1. Guided Project: Blackboard Fun
7. Day 7:
   1. Chapter 9
      1. Guided Project: Card Wars
      2. Extra Project 3: Dice Rolling
8. Day 8:
   1. Chapter 10
      1. Guided Project: Beach Balls
      2. Extra Project 5: Memory Game

Comments to the instructor are in [brackets]. This is usually a warning or some tips.

**Day 1:**

Daily Class Goals:

* Introduce Instructor and Students
* Familiarize Students with Visual Studio & Visual C#
* Open an existing project in VC#
* Successfully Compile a Project in VC#
* Complete a Simple Project, with a Button Control that runs code.

Daily Plan:

* **Lesson 1: Sample**
  + Simple example project to show students how to load existing projects.
  + It will provide about 5 minutes of “exploration.”
* **Lesson 2a: FirstTry**
  + [This is a Project where we will primarily just demonstrate the tools and windows in Visual C#.]
  + Explain the structure of a C# program (2-2): Solution, Project, Form, Controls.
  + Explain the concept of an “event.” (2-3)
  + Create an example project with the students, and call it “FirstTry” (2-6).
  + Explain Code Window and Designer. (2-9)
    - Drag a button on a Form, resize it, change some of the properties.
  + Explain the Toolbox and Properties Windows
  + Open the “sample” project. (2-13)
  + Open the Code Window and look at the code with the students. (2-13)
    - Focus on trying to demonstrate how the properly named variables and methods can help make the code very straight forward (2-17).
* **Guided Project 3: Beep**
  + Create a new project.
    - We will be bouncing between these steps and the Step-By-Step project below.
  + Explain the process of Dragging and Dropping controls.
  + Review the Naming Conventions for Controls (3-11)
    - btn, lbl, txt, chk, rdo
  + Explain how to change properties with code (3-13)
  + Examine the structure of an Event (3-17): visibility, return types, and parameters. (Use Sample for this).
  + Explain how to Add a new Event (3-21).
  + Explain how Intellisense will help complete code (3-26)

Example Demonstrations and Comments:

* **Guided Project 3 Comments: Beep**
  + Steps are listed below.
  + Watch out for rampant double clicking. Double clicking a control will cause VS# to generate an event for that control. Doing so can make your code very bloated and confusing, and in rare cases break functionality.
  + The concept of the “active” control is very important, as it will be the control that the properties are linked to, and also the control that will be the source when using “align” and other similar tools.
  + Demonstrate how it is possible to select a control by clicking on it, or selecting it from the drop down menu in the Properties Panel. (3-10)
  + For the AP Comp Sci people out there, since all (well, the default place) of the code is on the Form, the keyword “this” will refer to the current object, which is the Form. The Form actually has instance variables of the controls (which are also Objects, and their instantiation is hidden from us). It might seem contrary to what is taught in AP Comp Sci, because we are accessing public variables of Objects!
  + Also for the AP Comp Sci people, the events typically are private methods on the Form (because why should an external object be calling an event?). This is further heresy when contrasted to the standard AP lesson of “always public methods, always private variables.” Just a heads up for any students who are took/are taking AP Comp Sci.

**Guided Project 3 Steps:**

1. **Create a new Project, call it Beep. (3-2)**
2. Resize the Window. (3-3)
3. Add a Button. (3-5)
4. Move and Resize the Button. (3-6)
5. Add another Button, and other types of Contols. (3-6)
6. Delete all the Controls except one Button on the Form (3-6)
7. Using the Properties, set the Form’s Width to 400 and Height to 250. (3-9)
8. Using the Properties, set the Button’s Location->Top to 110 and the   
   Location->Left to 170 (3-9)
9. Change the Button’s Text to “Click Me”. (3-10)
10. Change the Form’s Text to “My First Code. (3-19)
11. Change the Button’s Name to “btnBeep.” (3-19)
12. Change *btnBeep*’s Text to “Beep!!” (3-19)
13. Add a new Click event to *btnBeep* (3-22)
14. Code the *btnBeep\_Click* method by adding the code on page 3-23.
    1. **System.Media.SystemSounds.Beep.Play();**
15. Test your Project
16. Add more code to the *btnBeep\_Click* method by adding the code on page 3-25.
    1. **btnBeep.BackColor = Color.Blue;**
17. Test your Project
18. Complete!

**Day 2:**

Daily Class Goals:

* Discuss the idea of “Project Design”
* Learn how to create Events for Controls
* Learn the controls: Forms, Buttons
* Learn the basic datatypes of C#
* Guided Project: Form Fun
* Complete Extra Project 1: StopWatch

Daily Plan:

* **Lesson 4: Experimenting the Properties**
  + Show students how to access Help with F1 (4-5).
  + Explain the important properties of the Form (4-8).
    - Name, Text, BackColor, Icon, Width, Height, FormBorderStyle, StartPosition
  + Explain the important events of a Form (4-10)
    - Click, Load, FormClosing
  + Explain the important properties of a Button (4-12).
    - Name, Text, TextAlign, Font, BackColor, ForeColor, Left, Top, Width, Height, Enable, Visible
  + Explain the important events of a Button (4-17)
    - Click
* **Discuss the strictness of C# programming syntax (4-19)**
  + Case Sensitive, Spelling, Curly Braces, Parentheses for Function Calls, Indenting, and Semicolons.
* **C# Language:**
  + Assignment Operator, = (4-21)
  + Integers, int (4-22)
  + Colors, Color (4-23)
    - Not a Primitive datatype.
  + Booleans, bool (4-23)
    - Minor annoying point, in code, use all lowercase, despite what the Property Viewer might show.
  + Strings, String (4-24)
  + Comments, // (4-25)
* **Guided Project 4: Form Fun**
  + There is a Completed version of this project in the BVCS Projects folder.
  + See the steps below.
* **Extra Project 1: StopWatch**
  + Have the students work on StopWatch after this completing Project 4.

Suggestion Demonstration and Comments:

* **Lesson 4: Experimenting the Properties**
  + This project doesn’t have a full set of steps.
  + Add two buttons as a “Start” and “Stop” button. Create code and set initial properties so that Stop is disabled by default, and when you press Start, it enables Stop, but disables Start (4-15).
  + Do the Same with Visible. Add a Label with the text “Running” on this form, which visibility set to false. When you click start, Visible will be True, when you click Stop, visible will be false.
  + When going over the syntax of C#, it might be good to show examples in this project. Could even declare a few variables so students can see how it works.

**Guided Project 4: Form Fun Steps:**

1. Create a new Project. (4-26)
2. Add Six Buttons and Arranged them in a 3x2 grid (4-26)
3. Change the Form.Text to “Form Fun” and Form.StartPosition to “CenterScreen” (4-27)
4. Set the Following Properties on the Buttons (4-27):
   1. button1: Name: btnShrink, Text: “Shrink Form”
   2. button2: Name: btnGrow, Text: “Grow Form”
   3. button3: Name: btnHide, Text: “Hide Buttons”
   4. button4: Name: btnRed, Text: “Red Form”
   5. button5: Name: btnBlue, Text: “Blue Form”
   6. button6: Name: btnShow, Text: “Show Buttons”, Visible: False
5. Create a Click event for *btnShrink*. (4-30)
6. Code *btnShrink\_Click()*: (4-30)
   1. **this.Height = this.Height - 10;  
      this.Width = this.Width - 10;**
7. Create a Click event for *btnGrow* (4-31)
8. Code *btnGrow\_Click()*: (4-32)
   1. **this.Height = this.Height + 10;  
      this.Width = this.Width + 10;**
9. Create a Click event for *btnRed* (4-32)
10. Code *btnRed\_Click()* (4-32)
11. Create a Click event for *btnBlue* (4-32)
12. Code *btnBlue\_Click()* (4-32)
13. Create a Click event for *btnHide* (4-32)
14. Code *btnHide\_Click()* (4-32)
15. Create a Click event for *btnShow* (4-33)
16. Code *btnShow\_Click()* (4-33)
17. Test your Project!
18. Project Complete!

**Day 3:**

Daily Class Goals:

* Discuss the Three Types of Errors
* Learn the controls: Labels and TextBox
* Learn about Working with Variables and Converting Types
* Guided Project: Saving Account

Daily Plan:

* **Lesson 5: Labels and TextBoxes (5-3)**
  + Explain the three types of errors (5-4)
    - Syntax, Run Time, and Logic
  + Explain the important properties of a Label (5-12).
    - Name, Text, TextAlign, Font, BackColor, ForeColor, Left, Top, Width, Height, BorderStyle, Visible, AutoSize
  + Explain the important events of a Label (5-15)
    - Click
  + Explain the important properties of a Text Box (5-17)
    - Name, Text, TextAlign, Font, MultiLine, ScrollBars, MaxLength, BackColor, ForeColor, Left, Top, Width, Height, ReadOnly, TabStop, BorderStyle, Visible
  + Explain the important events of a Text Box (5-20)
    - TextChanged, Leave
  + Discuss what “focus” means in terms of Controls (5-20)
* **C# Language:**
  + Variables (5-22)
    - Naming Rules (5-23)
  + Decimal/Floating Point Numbers, double (5-24)
  + Variable Declaration (5-25)
    - Scope: General Declaration and Local Declaration (5-26)
  + Type Casting (5-28)
    - Explicit Casting to convert type
  + Arithmetic Operators (5-29)
    - Addition, Multiplication, Subtraction, Division, Modulus
  + Operator Precedence (Like PEDMAS), same tier = same time (5-30)
    - 1st: \* and /
    - 2nd: %
    - 3rd + and –
  + Method Calls (5-34)
  + Converting Datatypes (5-33)
    - Convert.ToInt32( str )
    - Convert.ToDouble( str )
    - Convert.ToString( intOrDouble )
  + String Concatenation (5-37):
    - String newStr = str1 + “ and “ + str2

Example Demonstrations and Comments:

* **Lesson 5: Labels and TextBoxes**
  + This project has no individual steps, aside from showing examples of the three kinds of errors.
    - Syntax Error (5-4)
    - Run-Time Error (5-6)
    - Logic Error (5-11)
  + This project has no steps, as it is another “test these things” type of project.
  + Add a label, turn off AutoSize and resize it and move it around. (5-13)
    - Labels typically serve as a means of data output and Form styling.
  + COMMON ERROR: A lot of students will try to use code to change the text of a label without accessing the .Text field. For example this line: lblExample.Text = “My Label Box”; will get incorrectly turned into: lblExample = “My Label Box”;
    - Students think the label is being turned into that text, not the Text property.
    - Also, the Quotes on the String value are only needed when using Code to set the .Text field, not when using the Properties Window.
  + Clicking a Label is an odd event (mostly because the user doesn’t have a clear cue like they do for a button), but it can be useful.
  + Try adding a Text Box to the project. Try having a button that when you click it, it sets the text in the text box to something, or sets a label equal to the stuff in the Text Box. (5-19)
    - Text Boxes typically serve two purposes: input control or a display. (5-19)
    - When using a TextBox as a display, remember to set ReadOnly to True.
  + The TextChanged event happens every keypress, so don’t go crazy with it. (5-20)
  + Once the discussion turns to C# coding, it might not hurt to set up a button that updates a label with some math operations to demonstrate the different arithmetic operators. You might need to use String conversions without explaining them at first. After operators, THEN explain the String conversions.
* **Guided Project 5: SavingsAccount**
  + Steps are below.
  + For an extension, include the ability to use an interest rate and another TextBox and Label to to an input for the interest rate (5-46). See this page in the book to complete these extensions.

**Guided Project 5: SavingsAccount Steps:**

1. Create a new Project called “Savings”. (5-38)
2. Add three Labels and across from each, a TextBox (total of 3) (5-38)
3. Add two Buttons at the bottom. (5-38)
4. Set the Following Properties on the Form:
   1. Form.Text: “Savings Account”
   2. Form.FormBorderStyle: “Fixed Single
   3. Form.StartPosition to “CenterScreen” (5-39)
5. Set the Following Properties on the Labels (5-39):
   1. label1: (The topmost label)
      1. Name: lblDepositHeading
      2. Text: “Weekly Deposit”
   2. label2: (The middle label)
      1. Name: lblWeeksHeading
      2. Text: “Number of Weeks”
   3. label3: (The bottom label)
      1. Name: lblTotalHeading
      2. Text: “Total Savings”
6. Set the Following Properties on the TextBoxes (5-40):
   1. textbox1: (The topmost textbox)
      1. Name: txtDeposit
      2. Text: “Weekly Deposit”
   2. textbox2: (The middle textbox)
      1. Name: txtWeeks
      2. Text: “Number of Weeks”
   3. textbox3: (The bottom textbox)
      1. Name: txtTotal
      2. Text: “Total Savings”
7. Set the Following Properties on the Buttons (5-40):
   1. button1: (The bottom left)
      1. Name: btnCompute
      2. Text: “Compute”
   2. button2: (The bottom right)
      1. Name: btnExit
      2. Text: “Exit”
8. Code a general declaration:
   1. **int deposit;  
      int weeks;  
      int total;**
9. Create a Click event for *btnCompute* (5-43)
10. Code *btnCompute\_Click()*: (5-44)
    1. **deposit = Convert.ToInt32(txtDeposit.Text);  
       weeks = Convert.ToInt32(txtWeeks.Text);  
       total = deposit \* weeks;  
       txtTotal.Text = "$" + Convert.ToString(total);**
11. Create a Click event for *btnExit* (5-44)
12. Code *btnExit\_Click()*: (5-44)
    1. **this.Close();**
13. Test your Project!
14. Project Complete!
15. Try Extension on 5-46 for other Ideas.

**Day 4: Chapter 6: UpDown Control, Decisions, Random Numbers**

Daily Class Goals:

* Discuss the Three Types of Errors
* Learn the controls: UpDown Control
* Learn about If Statements and Random Numbers
* Guided Project: Guess the Number Game
* Extra Project 2: Times Tables

Daily Plan:

* **Lesson 6: Numeric Controls (6-2)**
  + Explain the important properties of a UpDown Controls (6-2).
    - Name, Value, Increment, Maximum, Minimum, TextAlign, Font, BackColor, ForeColor, Left, Top, Width, Height, ReadOnly, Enable, Visible
  + Explain the important events of an UpDown Control (6-5)
    - ValueChanged
* **C# Language:**
  + Logic Expressions (6-6)
    - Comparison Operators (6-8)
      * Equal to: ==
      * Not Equal to: != (6-9)
      * Greater Than: >
      * Less Than: <
      * Greater Than or Equal to: >= (6-10)
      * Less Than or Equal to: <=
    - Logical Operators (6-11)
      * And: &&
      * Or: || (6-12)
      * Not: !
      * Xor: ^
      * Operator Precedence: Not, And, Or, Xor (6-14)
  + If-Statements and If-Else Statements (6-16)
  + If-Else-If Statements (6-19)
  + Random Numbers (6-22)
    - Random Number Object Instantiation (6-22)
    - .Next( limit ) command (6-23)
    - Example of a die roll (6-24)
* **Guided Project 6: Guess the Number Game (6-25)**
  + The Steps are included below.
  + The extensions include adding a “cold / warmer / hot” system that tells you how close you are to the number, in addition to the higher or lower system. (6-37)
    - This uses Math.Abs ( num ).

Project Notes and Comments:

* **Lesson 6: Numeric Controls**
  + This project has no individual steps, aside from demonstrating the UpDown control and Random Numbers.
  + Insert an UpDown control and change the settings (6-4)
    - Keep in mind that the UpDown’s value is a Double, so if you plan to use it as an Int, you would need to Cast it as an Int.
  + During the C# Language discussion, use the project to demo the different logical expressions.
  + For if-statements, the Lemonade Stand example could be added to this project (6-17)
    - Watch out for the typo with the capital Else. Else should be lowercase in the code.
    - The logic error on (6-21) is a great example to demonstrate to the students of else-if chaining.
  + Add a button and a label and test changing the label to a random number by pressing the button (6-23)
* **Guided Project 6: Guess the Number Game (6-25)**
  + Steps are below.
  + Extension: Textbox to set the max value (6-37)
  + Extension: Absolute value Cold / Warm / Warmer hints
* **Extra Project 2 – Times Tables**
  + This a good project to reinforce random numbers.
  + The only odd parts of this project come from the *txtAnswer\_KeyPress()* method.
    - The first part (the e.KeyChar part) is checking for only numeric key presses and if the enter key is pressed.
    - The other confusing part is the String.Format command on the percentage.

**Guided Project 6: Guess the Number Game:**

1. Create a new Project called “GuessNumber”. (6-26)
2. Add a TextBox, a Numeric UpDown, and Three Buttons (6-26).
   1. The TextBox will display the result, the UpDown is the guess, and the buttons will be a “Guess” button, a “New Number” button, and an “Exit”
3. Set the Following Properties on the Form: (6-27)
   1. Form.Text: “Guess the Number”
   2. Form.FormBorderStyle: “Fixed Single
   3. Form.StartPosition to “CenterScreen”
4. Set the Following Properties on the TextBox and UpDown (6-27):
   1. textBox1:
      1. Name: txtMessage
      2. TextAlign: Center
      3. Font: Arial
      4. Font Size: 16
      5. BackColor: White
      6. ForeColor: Blue
      7. ReadOnly: True
      8. TabStop: False
   2. numericUpDown1:
      1. Name: nudGuess
      2. Font: Arial
      3. Font Size: 16
      4. BackColor: White
      5. ForeColor: Red
      6. TextAlign: Center
      7. Value: 50
      8. Minimum: 0
      9. Maximum: 100
      10. Increment: 1
      11. ReadOnly: True
      12. Enabled: False
5. Set the Following Properties on the Buttons (6-28):
   1. button1:
      1. Name: btnCheck
      2. Text: “Check Guess”
      3. Enable: False
   2. button2:
      1. Name: btnPick
      2. Text: “Pick Number”
   3. button3:
      1. Name: btnExit
      2. Text: “Exit”
6. Create a Click event for all three Buttons (6-30)
7. Code a general declaration: (add these after the Form1 constructor)
   1. **int theNumber;  
      int myGuess;  
      Random myRandom = new Random();**
8. Code *btnPick\_Click()*: (6-33)
   1. **if (btnPick.Text == "Pick Number")  
      {  
       theNumber = myRandom.Next(101);  
       txtMessage.Text = "I’m thinking of a number between 0 and 100";  
       nudGuess.Value = 50;  
       nudGuess.Enabled = true;  
       btnCheck.Enabled = true;  
       btnPick.Text = "Show Answer";  
      }  
      else  
      {   
       txtMessage.Text = "The answer is " + Convert.ToString(theNumber);  
       nudGuess.Value = theNumber;  
       nudGuess.Enabled = false;  
       nudGuess.Enabled = false;  
       btnPick.Text = "Pick Number";  
      }**
9. Code *btnCheck\_Click()*: (6-34)
   1. **myGuess = (int) nudGuess.Value;  
      if (myGuess == theNumber)  
      {  
       txtMessage.Text = "Correct!!";  
       nudGuess.Enabled = false;  
       btnCheck.Enabled = false;  
       btnPick.Text = "Pick Number";  
      }  
      else if (myGuess < theNumber)  
      {   
       txtMessage.Text = "Too low!";  
      }  
      else  
      {  
       txtMessage.Text = "Too high!";  
      }**
10. Code *btnExit\_Click()*: (5-44)
    1. **this.Close();**
11. Test your Project!
12. Project Complete!

Try Extension on 6-37 for other Ideas.

-Use a textbox to set the max value of the game

-Add a freezing / cold / warm / hot system using Absolute value command Math.Abs(int)

**Day 5: Chapter 7: Icons, Group Boxes, Check Boxes, Radio Buttons**

Daily Class Goals:

* Discuss the Importance of grouping controls and how that can help create more complex interactivity between input and output (as opposed to just simple buttons).
* Learn the controls: Icons, Group Boxes, Check Boxes, Radio Button
* Learn about Switch Statements
* Guided Project: Guess the Number Game
* Extra Project 2: Times Tables

Daily Plan:

* **Lesson 7: Grouping Controls (7-2)**
  + Explain how .ico (**Icon**) files work (7-2).
    - Assigning Icon Files (7-7).
  + Explain the important properties of a **Group Box Control** (7-8).
    - Standard Properties such as Name
    - Putting other controls into a Group Box and Moving them (7-10).
  + Explain the important properties of a **Check Box Control** (7-13).
    - ***Checked***: True or False if this box is checked
    - When using check boxes, it is recommended to always use a Group Box.
    - Events: ***CheckedChanged***: When the checked property changes.
  + Explain the important properties of a **Radio Button Control** (7-17).
    - ***Checked:*** True or False if this is selected—only one Radio Button per group can be Checked.
    - Radio Buttons are best used with Group Boxes because they will exhibit mutually exclusive behavior automatically if in a group.
    - Should always default one Radio button to True.
    - Events: ***CheckedChanged***: When the checked property changes (from either setting to the other)
* **C# Language:**
  + Logic Expressions (7-22)
    - Switch (7-22)
      * Case
      * Break
      * Default
* **Guided Project 7: Sandwich Maker (7-25)**
  + The goal of this project is to create a computerized sandwich ordering menu. The design will allow for the user to pick one bread (white, wheat, rye), any number of meats (roast beef, ham, turkey, pastrami, salami), one cheese (none, American, Swiss), and any of six condiments (mustard, mayonnaise, lettuce, tomato, onion, pickles).
  + The extension adds the ability to calculate price and change.
  + The Steps are included below.

Project Notes and Comments:

* **Lesson 7: Grouping Controls (7-2)**
  + This project has no individual steps, aside from demonstrating the Group Boxes, Check Boxes, and Radio Buttons.
  + Insert Group Control and other controls (7-11).
    - Keep things to demonstrate:
      * Deleting the group, deletes all the controls
      * Setting Enable or Visible for the group will change all of the controls inside.
  + When using switch statements, break; is required, otherwise it will cascade into the next case.
* **Guided Project 7: Sandwich Maker (7-25)**
  + They use the CheckChanged feature of Radio Buttons to determine which type of bread or cheese is selected (by changing the variable). Technically, you can also do this just with a bunch of if-statements when they click the “order” button (checking each radio button for Checked). Using CheckedChanged event is more efficient.
  + The logic explanations on 7-36, 7-39, and 7-42 are great to include in your guided discussion.
  + Steps are below.

**Guide Project 7: Sandwich:**

1. Create a new Project called “Sandwich”. (7-26)
2. Add three Radio Buttons and group them together (Group 1) in the top left of the UI. (7-26)
3. Add five Check Boxes and group them together (Group 2) in the bottom left of the UI (7-26)
4. Add three Radio Buttons and group them together (Group 3) in the top middle of the UI (7-26)
5. Add six Check Boxes and group them together (Group 4) in bottom middle of the UI (7-26)
6. Add a Label and Textbox in the top right (7-26).
7. Add two Buttons to the bottom right (7-26)
8. Set the Following Properties on the Form: (7-27)
   1. Form.Text: “Sandwich Maker”
   2. Form.FormBorderStyle: “Fixed Single”
   3. Form.StartPosition to “CenterScreen”
   4. Form.Icon to one of your choice
9. Defaults for Groups:
   1. Group.FontSize = 10
   2. Group.FontStyle = Bold
10. Defaults for Controls:
    1. Control.FontSize = 8
    2. Control.FontStyle = Regular
11. Names for Group 1: (7-27)
    1. groupBox1:
       1. Name: grpBread
       2. Text: “Bread”
    2. radioButton1: rdoWhite, “White”, True
    3. radioButton2: rdoWheat, “Wheat”
    4. radioButton3: rdoRye, “Rye”
12. Names for Group 2: (7-28)
    1. groupBox2:
       1. Name: grpMeats
       2. Text: “Meats”
    2. checkBox1: chkRoastBeef, “Roast Beef”
    3. checkBox2: chkHam, “Ham”
    4. checkBox3: chkTurkey, “Turkey”
    5. checkBox4: chkPastrami, “Pastrami”
    6. checkBox5: chkSalami, “Salami”

1. Names for Group 3: (7-29)
2. groupBox3:
   * 1. Name: grpCheese
     2. Text: “Cheese”
3. radioButton4: rdoNone, “None”, True
4. radioButton5: rdoAmerican, “American”
5. radioButton5: rdoSwiss, “Swiss”
6. Names for Group 4: (7-30)
7. groupBox4:
   * 1. Name: grpCondiments
     2. Text: “Condiments”
8. checkBox6: chkMustard, “Mustard”
9. checkBox7: chkMayo, “Mayonnaise”
10. checkBox8: chkLettuce, “Lettuce”
11. checkBox9: chkTomato, “Tomato”
12. checkBox10: chkOnions, “Onions”
13. checkBox11: chkPickles, “Onions”
14. Set the Following Properties on Label and TextBox (7-31):
15. label1:
16. Name: lblOrder
17. Text: “Order: ”
18. FontSize: 10, FontStyle: Bold
19. textBox1:
20. Name: txtOrder
21. MultiLine: True
22. ScrollBars: Vertical
23. Set the Following Properties on the Buttons (7-32):
24. button1:
25. Name: btnOrder
26. Text: “Order”
27. Enable: False
28. button2:
29. Name: btnClear
30. Text: “Clear”
31. Create a Click event for both Buttons (7-34)
32. Code a general declaration: (add these after the Form1 constructor)
33. **int breadChoice;  
    int cheeseChoice;  
    int numberMeats;  
    int numberCondiment;**
34. Code *Form1\_Load(object sender, EventArgs e)*: (7-36)
35. **breadChoice = 1;  
    cheeseChoice = 0;**

*Set up the bread Radio Buttons:*

1. Code *rdoWhite\_CheckedChanged*: (7-37)
2. **breadChoice = 1;**
3. Code *rdoWheat\_CheckedChanged*: (7-37)
   1. **breadChoice = 2;**
4. Code *rdoRye\_CheckedChanged*: (7-37)
   1. **breadChoice = 3;**

*Set up the Cheese Radio Buttons:*

1. Code *rdoNone\_CheckedChanged*: (7-38)
2. **cheeseChoice = 0;**
3. Code *rdoAmerican\_CheckedChanged*: (7-38)
   1. **cheeseChoice = 1;**
4. Code *rdoSwiss\_CheckedChanged*: (7-38)
   1. **cheeseChoice = 2;**

*Set up the main order button, read the explanation on 7-41:*

1. Code *btnOrder\_Click*: (7-39)

**txtOrder.Text = "Sandwich Order:\r\n";**

**switch (breadChoice)**

**{**

**case 1:**

**txtOrder.Text = txtOrder.Text + "White Bread\r\n";**

**break;**

**case 2:**

**txtOrder.Text = txtOrder.Text + "Wheat Bread\r\n";**

**break;**

**case 3:**

**txtOrder.Text = txtOrder.Text + "Rye Bread\r\n";**

**break;**

**}**

**// Add and count meats**

**numberMeats = 0;**

**if (chkRoastBeef.Checked)**

**{**

**numberMeats = numberMeats + 1;**

**txtOrder.Text = txtOrder.Text + "Roast Beef\r\n";**

**}**

**if (chkHam.Checked)**

**{**

**numberMeats = numberMeats + 1;**

**txtOrder.Text = txtOrder.Text + "Ham\r\n";**

**}**

**if (chkTurkey.Checked)**

**{**

**numberMeats = numberMeats + 1;**

**txtOrder.Text = txtOrder.Text + "Turkey\r\n";**

**}**

**if (chkPastrami.Checked)**

**{**

**numberMeats = numberMeats + 1;**

**txtOrder.Text = txtOrder.Text + "Pastrami\r\n";**

**}**

**if (chkSalami.Checked)**

**{**

**numberMeats = numberMeats + 1;**

**txtOrder.Text = txtOrder.Text + "Salami\r\n";**

**}**

**// If no meats picked, say so**

**if (numberMeats == 0)**

**{**

**txtOrder.Text = txtOrder.Text + "No Meat\r\n";**

**}**

**// Add cheese type**

**switch (cheeseChoice)**

**{**

**case 0:**

**txtOrder.Text = txtOrder.Text + "No Cheese\r\n";**

**break;**

**case 1:**

**txtOrder.Text = txtOrder.Text + "American Cheese\r\n";**

**break;**

**case 2:**

**txtOrder.Text = txtOrder.Text + "Swiss Cheese\r\n";**

**break;**

**}**

**// Finally, add and count condiments**

**numberCondiments = 0;**

**if (chkMustard.Checked)**

**{**

**numberCondiments=numberCondiments+1;**

**txtOrder.Text = txtOrder.Text + "Mustard\r\n";**

**}**

**if (chkMayo.Checked)**

**{**

**numberCondiments=numberCondiments+1;**

**txtOrder.Text = txtOrder.Text + "Mayonnaise\r\n";**

**}**

**if (chkLettuce.Checked)**

**{**

**numberCondiments=numberCondiments+1;**

**txtOrder.Text = txtOrder.Text + "Lettuce\r\n";**

**}**

**if (chkTomato.Checked)**

**{**

**numberCondiments=numberCondiments+1;**

**txtOrder.Text = txtOrder.Text + "Tomato\r\n";**

**}**

**if (chkOnions.Checked)**

**{**

**numberCondiments=numberCondiments+1;**

**txtOrder.Text = txtOrder.Text + "Onions\r\n";**

**}**

**if (chkPickles.Checked)**

**{**

**numberCondiments=numberCondiments+1;**

**txtOrder.Text = txtOrder.Text + "Pickles\r\n";**

**}**

**// If no condiments picked, say so**

**if (numberCondiments == 0)**

**{**

**txtOrder.Text = txtOrder.Text + "No Condiments\r\n";**

**}**

**}**

1. Code *btnExit\_Click()*: (7-43)

**// Set bread to white**

**breadChoice = 1;**

**rdoWhite.Checked = true;**

**// Clear all meat choices**

**chkRoastBeef.Checked = false;**

**chkHam.Checked = false;**

**chkTurkey.Checked = false;**

**chkPastrami.Checked = false;**

**chkSalami.Checked = false;**

**// Set cheese to none**

**cheeseChoice = 0;**

**rdoNone.Checked = true;**

**// Clear all condiment choices**

**chkMustard.Checked = false;**

**chkMayo.Checked = false;**

**chkLettuce.Checked = false;**

**chkTomato.Checked = false;**

**chkOnions.Checked = false;**

**chkPickles.Checked = false;**

**// Clear text box  
txtOrder.Text = "";**

1. Test your Project!
2. Project Complete!

Try Extension on 7-45 for other Ideas.

-Create an Exit Button

-Calculate Price

-Enter a payment amount and calculate change

-Super challenge: calculate the bills and coins required

**Day 6:**

Daily Class Goals:

* 1

Pre-Project Recommended Discussion and Demonstrations:

* Lesson 8
  + 1

Project Notes and Comments:

* Project 8a
  + 1

**Day 7:**

Daily Class Goals:

* 1

Pre-Project Recommended Discussion and Demonstrations:

* Lesson 9
  + 1

Project Notes and Comments:

* Project 9a
  + 1

**Day 8:**

Daily Class Goals:

* 1

Pre-Project Recommended Discussion and Demonstrations:

* Lesson 10
  + 1

Project Notes and Comments:

* Project 10a
  + 1