DUE: 11/7/2016

Project #2—Fuzzy Dice Order Form

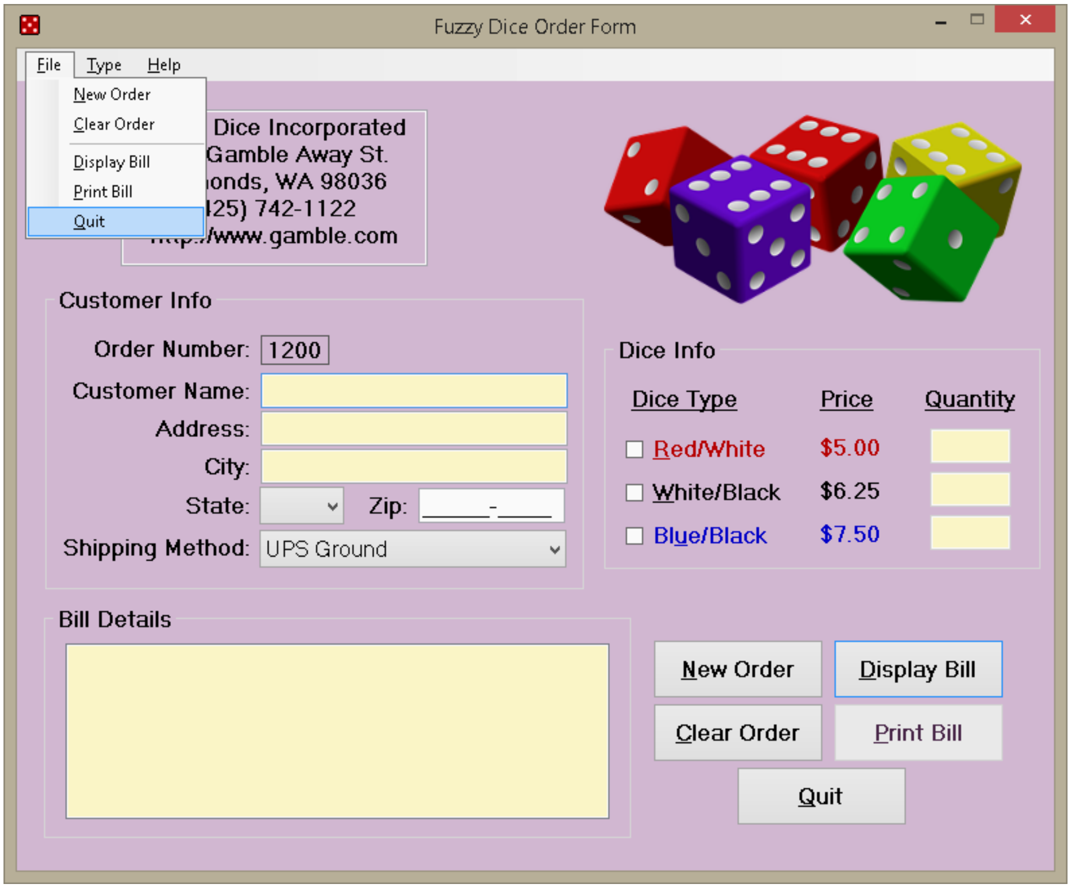
Fuzzy Dice, Inc. produces and sells three types of dice: white/black for $6.25 per pair, red/white for $5.00 per pair, and blue/black for $7.50 per pair. Write a C# Windows application that allows users to process orders for Fuzzy Dice (sample GUI is provided below). The application should calculate the total price of the order, including tax and shipping. Tax is 8.25% and shipping is done via UPS with the following rate table:

Shipping Charges	
UPS Ground	$8.95
UPS 3 Days	$16.95
UPS Next Day	$25.95
For orders over 20: Free UPS Ground Shipping	

Your C# program should display the order form. The order number, customer's name and address, and the quantities for each type of dice should be entered into the program via text boxes (you may use a masked text box for the order number). Use check boxes for the type of dice ordered. Checking a dice check box enables the corresponding text box to allow user to enter quantity ordered. Initially, all check boxes are unchecked with corresponding text boxes disabled. When a Display Bill button is clicked, the customer's name and the invoice should be displayed in the list box, as shown in the sample run below. Note that a discount of 7% applies if customers order $500 or more (before tax and shipping). Furthermore, if a customer orders over 20 pair of dice then the shipping is free if UPS Ground is selected.

1. The project is worth 30 points and it will be graded on accuracy, completeness, presentation, comments, and the like.
2. Make sure that your name, project number, and due date appear as comments in your code (preferably on top). Comments should abound
3. Submit a zipped file with all files and subfolders via the Canvas by the deadline.

EXTRA CREDIT (Optional—3 points). Enhance the form by using a combo box with dropdown for the customers dat. Read the customer information from a text file as the form loads. Consider also adding a separate Invoice class which does all the calculations.



Order Number: 1200

Customer Name:

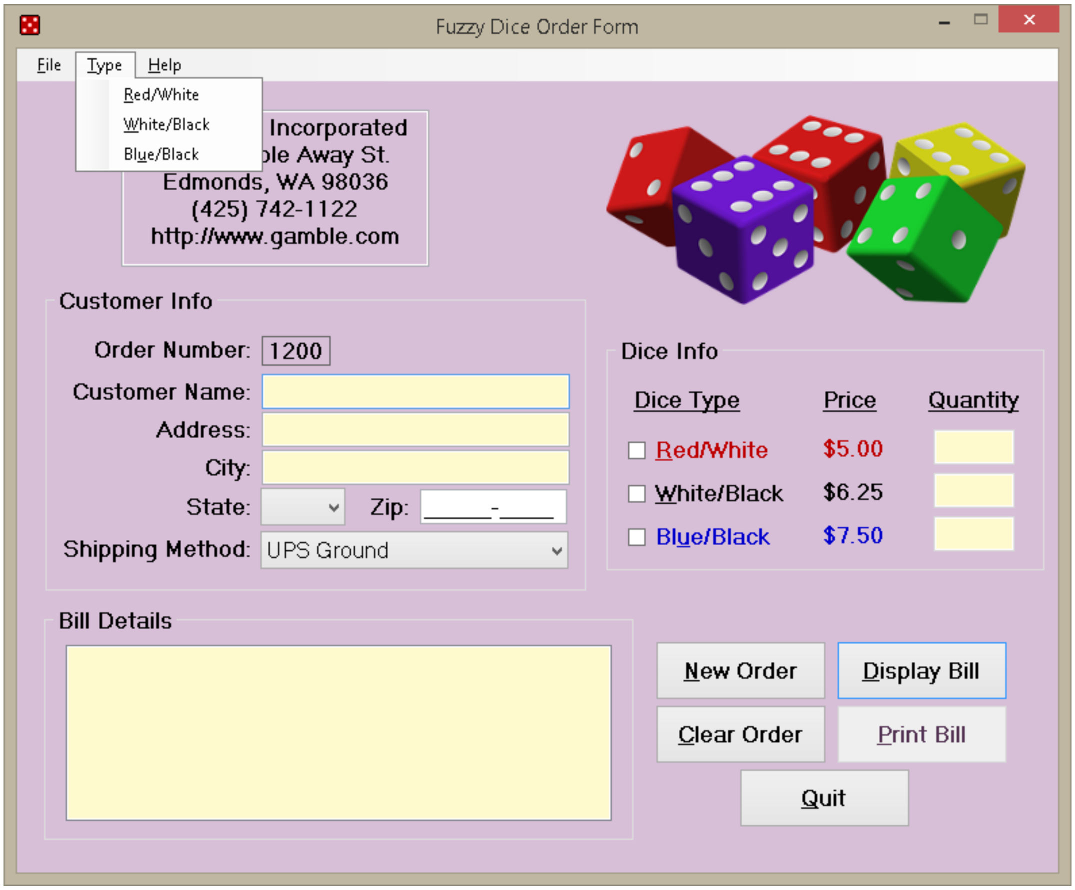
Address:

City:

State: Zip: _____-____

Shipping Method: UPS Ground

Dice Type	Price	Quantity
□ Red/White	$5.00	
□ White/Black	$6.25	
□ Blue/Black	$7.50	



Order Number: 1200

Customer Name:

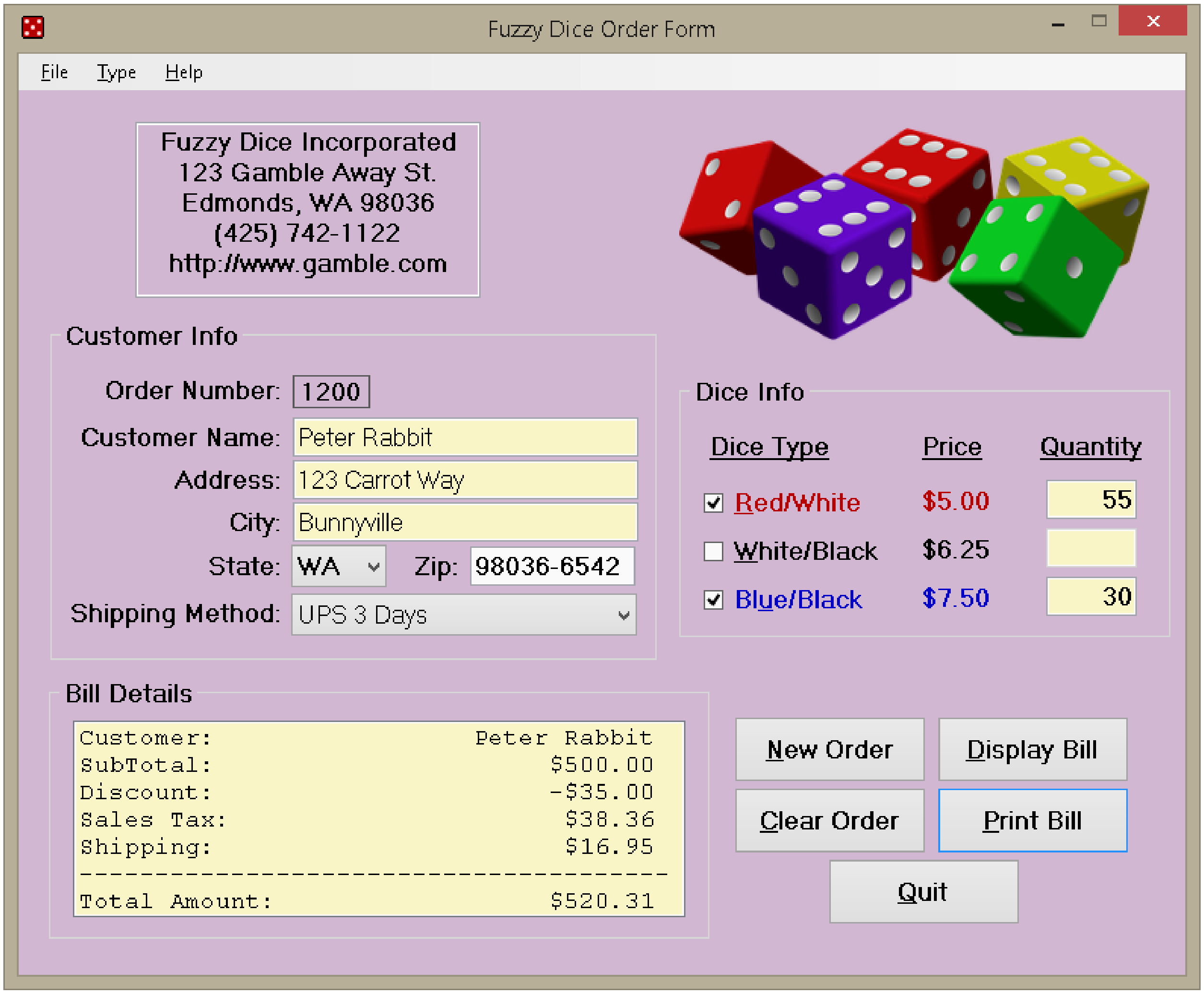
Address:

City:

State: Zip: _____-____

Shipping Method: UPS Ground

Dice Type	Price	Quantity
□ Red/White	$5.00	
□ White/Black	$6.25	
□ Blue/Black	$7.50	



Fuzzy Dice Incorporated
123 Gamble Away St.
Edmonds, WA 98036
(425) 742-1122
http://www.gamble.com

Order Number: 1200
Customer Name: Peter Rabbit
Address: 123 Carrot Way
City: Bunnyville
State: WA Zip: 98036-6542
Shipping Method: UPS 3 Days

Dice Type	Price	Quantity
☑ Red/White	$5.00	55
□ White/Black	$6.25	
☑ Blue/Black	$7.50	30

```
Customer:                  Peter Rabbit
SubTotal:                       $500.00
Discount:                       -$35.00
Sales Tax:                       $38.36
Shipping:                        $16.95
----------------------------------------
Total Amount:                   $520.31
```

Grading Guidelines: Your project will be graded on the following:

- Correct solution to the proposed problem.
- GUI design (as outlined above).
- Elegance, simplicity, style, and readability of the algorithms and code.
- Comments:
  - ✓ Use of header comments section.
  - ✓ Description of project.
  - ✓ Description of methods.
  - ✓ Description of complicated code.

Hints for Project #2: *Given that your GUI resembles the given one, follow these guidelines:*

1. Use a masked text box with appropriate mask for the order number and zip.
2. Set Courier New for font in the list box and use a formatting string to display the bill with the appropriate justification as shown in the sample GUI.
3. Show the price of each type die at form_load using predefined class-level constants. This allows for easy change of prices.

4. The Clear button differs from the New Order button in that the Clear button clears everything but not the order number and not the customer information. The New Order button clears everything, including the customer information and increments the current order number by one.
5. Disable initially all quantity text boxes and enable each one with focus set to it only when a corresponding check box is selected (or chosen from the Type menu). Similarly, un-checking a check box should clear and disable the corresponding text box.
6. The main form has menu items File (with New Order, Clear Order, Print Bill, and Quit), Type (Red/White, White/Black, and Blue/Black), and Help (with About).
7. Synchronize checking and unchecking of the check boxes with the corresponding choices in the Type menu item.
8. Calculate subtotal first, based on number of dice ordered—note that each color combination dice have a different price. Make sure you apply discount first (7% off the subtotal if it is >= $500), if any, before calculating the tax (8.25%).
9. Calculate the shipping charges next—it helps to draw a flowchart here. Essentially, consider the shipping choice in the combo box (check for the text property of the combo box). If it is "UPS Ground" then the shipping charge will be 0 if more than 20 items are ordered else it is $8.95. Otherwise, if the shipping choice is "UPS 3 Days", the shipping charge is $16.95, else it is $25.95.
10. Calculate the total by adding to the subtotal the tax and the shipping and subtracting the discount. Use a formatting string for displaying columns with print zones and justification in a list box.
11. Provide input validation. Do not allow wrong type (Try/Catch blocks) or excessively large number (you decide what is that maximum) or negative number of dice, for example.
12. Declare constants for all constants used (such as 8.25% tax, 7% discount, dice prices, shipping charges, etc., including maximum number of dice to order, minimum constants for discount or for free shipping).
13. Make the check boxes and the shipping combo box dynamic—any change in their state should automatically recalculate the invoice without having to press the Display Bill button.
14. Make sure the menus are functional without repeating the code. The sample GUI has a File, Type, and Help main menu items. File contains Clear, New Order, Separator, Print, and Quit. Type contains White, Red, and Blue, and Help contains About for an optional About form. All menu choices can be activated with a corresponding "Hot" key combination, such as Alt-F for File.
15. Set the Tab Order index appropriately so that the form can be filled in proper order. You can invoke the Tab Order from View menu.
16. For test purposes do not require completion of customer information.

17. Add an About form which will describe the project: Explaining that each price is for a pair of dice—thus a quantity of one (1) yields two dice, describing shipping charges and discounts, copyright and version, author, etc.
18. Add a Splash Screen with a progress bar and with the same logo picture as in the main form and the About form.
19. The Print Bill button should print the form as an image; the print menu item should print the content of the list box.

Pay attention to the following criteria for grading:

```
/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'Comments by the prof:
'Great effort.  Here are suggestions for improving:
'1. Use comments through the program, not just at the top heading
'Use comments to describe each method and complicated code sections.
'2. Convert entries in textboxes from strings to numbers with the appropriate convert functions:
Convert.ToByte, Convert.ToDecimal, etc.
'3. Program crashes on empty.
'4. Make the Display Bill button default.
'5. Correct the formulas—results are incorrect.
'6. Apply 8.25% tax to bill.
'7. Use Const definition for all constants and follow naming convention of ALL CAPITALS.
'8. Apply 7% discount on orders of $500 or more.
'9. Provide Display Bill, Clear Order, New Order, Print Bill, and Quit buttons.
'10. Use Application.Exit() to exit program.
'11. Use a ComboBox with DropDownList property for the three UPS shipping rates.
'12. Free shipping applies if order has over 20 items and UPS ground is chosen for shipping.
'13. Name the project/form appropriately; a generic WindowsApplication1, Project1 or Form1 just won't do.
'14. Use the Heading.txt heading for the project.
'15. Quantity textboxes are enabled only if corresponding checkbox is checked.
'16. Display bill as currency.
'17. Follow the naming convention for naming variables and constants.
'18. Avoid excessive use of class variables-declare variables inside procedures.
'19. Clean up the empty procedures code.
'20. Separate code in main procedure into 4 section: Declaration, Assignment, Calculation, and Display.
'21. Implement input validation for all text boxes (avoid -3 dice, for example).
'22. Select UPS Ground as default text for shipping combo box.
'25. Declare and use variables for input.
'26. Use the Clear() method to clear text in a text box.
'27. Nice effort on the printing--printing the bill alone would suffice (without all controls).
'28. Text boxes should be empty initially.
'29. Dice quantities should be integers; all $$ variables should be Decimal type.
'30. Replace if (chkRed.Checked = true) with if chkRed.Checked.
'31. Implement menus & printing. Synchronize Types in menu with check boxes.
```

```
'32. Use mono-spaced font such as Courier for the list box--it helps set columns correctly.
'33. Use print zones with string formatting to set up columns in display list box.
'34. Add an image and icon to the form.
'35. Check the Tab Order on your form.
'36. Clear should clear check boxes, list box and quantity text boxes—leave customer information intact.
'37. Disable maximization of form.
'38. Make check boxes and combo box dynamic.
'39. Missing About form or Splash Screen.
'40. Why require order of all type of dice? Allow for orders with only one type of dice.
'
'The ones that apply to your project are:
'3, 7, 15, 21, 30, 31, 40
'
'24+2=26/30
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/
```