Class Assignment – Yahtzee
Part B

Due Date: 7th June, 2016 – No late submissions accepted. Weighting: 40%

The Objective of Part B

The objective of Part B of the assignment is to code the methods and instance variables of the **`Form1`** class. Below is the class diagram for that class. You must use the names for variables, methods and parameters as specified.

```
Form1
----------------------------------------------
-dice : Array of Label
-scoreButtons : Array of Button
-scoreTotals : Array of Label
-checkBoxes : Array of CheckBox
-game : Game
----------------------------------------------
-InitializeLabelsAndButtons()
+GetDice() : Array of Label
+GetScoresTotals() : Array of Label
+ShowPlayerName(string name)
+EnableRollButton()
+DisableRollButton()
+EnableCheckBoxes()
+DisableAndClearCheckBoxes()
+EnableScoreButton(ScoreType combo)
+DisableScoreButton(ScoreType combo)
+CheckCheckBox(int index)
+ShowMessage(string message)
+ShowOKButton()
+StartNewGame()
```

The class diagram does not name the event handlers that will be required. **Do not add event handlers to the code until told to do so.**

The following section provides a brief description of the members of this class as specified by the above UML class diagram.

Where to start.

First use the UML diagram to populate **Form1.cs** with the instance variables and method headings**.** Complete the body of all methods as described below. You should add constants to avoid the proliferation of magic numbers. Make sure that your code continues to compile without errors.

Instance variables

dice holds the labels representing the five dice.

scoreButtons is an array to hold the buttons for the scoring combinations. Though there are 13 buttons, this array should be declared with 16 elements to enable you to use the values of the enumeration **`ScoreType`** defined in **Game.cs** to access that button easily. Hover over the value **`Yahtzee`** in the enumeration to see why 16 is the “magic” number. Not all elements of this array will be used as some elements do not correspond to a scoring combination button.

scoreTotals is an array of the 19 labels which will show the score for each of the combinations and the various totals.

checkBoxes is an array to hold the checkboxes which activate/deactivate the dice.

game is the link to the current **`Game`** object so that the methods of that object can be called from **`Form1.`** **Game.cs** has the empty body of class **`Game`**. This class will be completely defined in Part C.

Methods

InitializeLabelsAndButtons method will put all the buttons, labels and checkboxes into the correct array so that the individual controls can be accessed easily.

GetDice will return the array representing the five dice.

GetScoreLabels will return an array representing all of the scores on the form.

ShowPlayerName will display the specified player’s **name** on the form.

EnableRollButton and **DisableRollButton** will enable or disable the "***Click to roll dice***" button

EnableCheckBoxes will allow the checkboxes to be checked, and **DisableAndClearCheckBoxes** will clear any ticks and disable all of the checkboxes.

EnableScoreButton and **DisableScoreButton** will enable or disable the specified button corresponding to the **`ScoreType`** parameter. **`ScoreType`** is an enumeration already defined in **Game.cs**. **Do not change this enumeration in any way.**

CheckCheckBox will "tick" the checkbox with the **index** in the array of checkboxes.

ShowMessage will display the supplied **message** on the ***message label***.

ShowOKButton will display the **OK button** and make it available to be clicked. At the moment this button has not been placed onto the form. **So for Part B the body of this method is empty.**

StartNewGame will need to instantiate a new **`Game`** object. Initially the body of this method can be simply the following assignment: `game = new Game();`

You may need to add a default constructor to the `Game` class so that your code compiles.