COMP123 – Programming 2

Assignment 5
BMI Calculator App

Due Week #14 (Tuesday August 15, 2017) @ midnight

Value 10%

BMI Calculator App **Maximum Mark: 54**

Overview: Using Windows Forms, create a **BMI calculator app** that allows users to enter their **weight** and **height** and whether they are entering these values in **Imperial** or **Metric** units, then calculates and displays the user's body mass index (BMI). The app should also display the following information from the Department of Health and Human Services/National Institutes of Health so the user can evaluate his/her BMI:

BMI SCALE	Result
Underweight	less than 18.5
Normal	between 18.5 and 24.9
Overweight	between 25 and 29.9
Obese	30 or greater

The Formula for BMI is as follows

$$BMI = \frac{weightInPounds \times 703}{heightInInches \times heightInInches}$$

or

$$BMI = \frac{weightInKilograms}{heightInMeters \times heightInMeters}$$

Instructions :

(23 Marks GUI, 18 Marks: Functionality, 3 Marks: Program Structure, 6 Marks: Internal Documentation, 4 Marks: Version Control)

1. Modify the following properties of the BMICalculator **Form (5 Marks: GUI):**
    a. The **Size** property must be set to 320, 480 (width and height) (1 Mark: GUI).
    b. The **Font** property must have a **Font Size** of 20 points (1 Mark: GUI).
    c. The **FormBorderStye** property must be set to **FixedDialog** (1 Mark: GUI).
    d. The **MaximizeBox** property must be set to **False** (1 Mark: GUI).
    e. The StartPosition property must be set to CenterScreen (1 Mark: GUI).
2. The following **UI Controls** should be **added** to the BMICalculator Form **(18 Marks: GUI, 18 Marks: Functionality):**
    a. Your BMICalculator User Interface (UI) will use a **TableLayoutPanel** container (2 Marks: GUI).
    b. A **RadioButton Control** that allows the user to **toggle** between **Imperial** or **Metric Units**. This control will affect the BMI calculation. See above for the BMI formula. (2 Marks: GUI, 2 Marks: Functionality).
    c. A **Label Control** with a **Text** value of "My Height" (2 Marks: GUI).
    d. An **TextBox Control** that allows the user to input his/her height in **inches** or **metres,** depending on the **RadioButton Control** setting. Ensure that appropriate input validation techniques are used to limit the user's entry to numeric data only for this control (2 Marks: GUI, 4 Marks: Functionality).
    e. A **Label Control** with a Text value of "My Weight" (2 Marks: GUI).
    f. An **TextBox Control** that allows the user to enter his weight in **pounds** or **kilograms.** Ensure that appropriate input validation techniques are used to limit the user's entry to numeric data only for this control (2 Marks: GUI, 4 Marks Functionality).
    g. A **BMI TextBox Control** which will display the user's BMI when the **Calculate BMI Button** is pressed. This control will be **disabled** for any user input but will have background and foreground colours adjusted appropriately. (2 Marks: GUI, 2 Marks: Functionality)
    h. A **Button** control with a Text value of "Calculate BMI" that displays the user's calculated BMI in the BMI **TextBox Control.** Any method used to calculate the user's BMI will use either **imperial** or **metric units**, depending on the unit type selected with the **RadioButton Control**. (2 Marks: GUI, 4 Marks: Functionality).
    i. A **Multiline TextBox Control** that displays the user's BMI results as per the BMI Scale above (2 Marks: GUI, 2 Marks: Functionality).
3. **Solution Structure (3 Marks: Program Structure):**
    a. Your solutions should include a Windows Form named **BMICalculator.cs** (1 Mark: Program Structure).
    b. The Windows Form and all attached **UI Controls** must have appropriate Variable names with the following format: **ControlNameUIControlType** (e.g. **CalculateBMIButton**) (1 Mark: Program Structure).

c. Ensure all *private* class member variables (instance variables) use an underscore character (_) at the beginning of the identifier name to signify that they are private (or protected) (1 Mark: Program Structure).

4. **Internal Documentation (6 Marks: Internal Documentation):**
    a. Ensure you include a program header that indicates: The Author's name, Author's student number, Date last Modified, Program description, Revision History for each project in your solution (4 Marks: Documentation).
    b. Ensure your program uses contextual variable names that help make each program human-readable (2 Marks: Documentation).
5. Share your files on **GitHub** to demonstrate Version Control Best Practices **(4 Marks: Version Control).**
    a. Your repository must include **your code** and be well structured (2 Marks: Version Control).

    Your repository must include **commits** that demonstrate the project being updated at different stages of development – each time a major change is implemented (2 Marks: Version Control).

Optional App Features (i.e. Potential Bonus Marks).

A. A Splash Screen that is displayed for 3 seconds and then displays the BMI App (4 Bonus Marks).
B. A colorful Progress Bar that changes in proportion to the user's BMI (4 Bonus Marks).
C. A **Reset** button that reset's the screen to its original state (2 Bonus Marks).
D. Number Keys (0 to 9) and a backspace button that limit the user's entry to improve input validation (6 Bonus Marks).

SUBMITTING YOUR WORK

Your submission should include:

1. A zip archive of your project files
2. A link to your project files on GitHub.

Feature	Description	Marks
GUI / Interface Design	UI Controls meet the application requirements. Display elements are deployed in an attractive manner. Appropriate contrast is applied to application UI Controls and any background colours applied so that all text is legible.	23
Functionality	The program's deliverables are all met and the program functions as it should. No errors appear as a result of execution. User Input does not crash the program.	18
Program Structure	Your main "driver" class is named Program and it creates objects that are defined in other classes. All other classes are contained in their own files. Your classes use **public** properties and related **private** member variables wherever possible.	3
Internal Documentation	A program header is present and includes the name of the program, the name of the student, student number, date last modified, a short revision history and a short description of the program. All methods and classes include headers that describe their functionality and scope and follow commenting best practices. Inline comments are used to indicate code function where appropriate. Variable names are contextual wherever possible.	6
Version Control	GitHub commit history demonstrating regular updates.	4
Total		**54**

This assignment is weighted **10%** of your total mark for this course.

All Assignments are due at the beginning of class.
Late submissions:

- 20% deducted for each additional day late.

External code (e.g. from the internet or other sources) can be used for student submissions within the following parameters:

1. The code source (i.e. where you got the code and who wrote it) must be cited in your internal documentation.
2. It encompasses a maximum of 10% of your code (any more will be considered cheating).
3. You must understand any code you use and include documentation (comments) around the code that explains its function.
4. You must get written approval from me via email.