COMP1004 – Rapid Application Development

Mid-Term Assignment
Character Generator

Due Week #7 (Friday February 24, 2017) @midnight

Value 20%

Character Generator

Maximum Mark: 53

Overview: Create a project that generates the first few pages of an RPG character (similar to Dungeons & Dragons, Traveller, Marvel Super Heroes, etc). This will be a multiform project that will display the results of randomly generated abilities and the users selections on the last form.

Resources:

- You may use course documents uploaded to blackboard, the internet and your notes if desired.
- Some of the Forms for the project are provided as well as certain images (imported in the project's resource folder).
- This Mid-Term Assignment is designed to use the Visual Studio 2015 IDE

Instructions:

1. The program should contain 4 forms: **AbilityForm** (generates the character's abilities), **RaceForm** (user selects the character's race), **JobForm** (user selects the character's Job) and **FinalForm** (the program displays the generated abilities and the user's selections).

   (17 Marks GUI, 25 Marks Functionality, 2 Marks: Program Structure, 5 Marks: Internal Documentation, 4 Marks Revision Control)
2. **AbilityForm:** (Form and basic GUI provided - Subtotal **4 Marks: Functionality**)
   a. Whenever the **RollButton** is clicked, the Character's Abilities (**STR**, **DEX**, **END**, **INT**, **PER**, **CHA**) will display a random number between 3 and 30. Use the **Roll3D10** method provided to generate a pseudo-random number and assign the result to each of the Ability Text Box Controls on the AbilityForm (3 Marks: Functionality).
   b. The **Next** Button opens **RaceForm** and hides **AbilityForm**. (1 Mark: Functionality)
3. **RaceForm**: (Form and some controls provided – Subtotal **7 Marks: Functionality, 4 Marks: GUI**)
   a. Add a Group Box and several Radio Button Controls to the Form (1 Mark: GUI)
   b. The Radio Buttons will allow the user to select **Human**, **Dwarf**, **Elf** or **Halfling** as the Character's **Race**. Each Radio Button Control should be Labelled accordingly. (2 Marks: GUI)

c. When the user selects a **Race**, the portrait in the **CharacterPictureBox** control changes to reflect his choice. **Note:** I have included.png files in the resource folder of the project for your use. (2 Marks: Functionality, 1 Mark: GUI)
d. Depending on which Race the user selects, the Character's Abilities that were previous generated on the AbilityForm will be adjusted and a message will be displayed in the **RacialBonusTextBox** to communicate this information to the player. **Note:** No Ability can be increased higher than 50 or decreased lower than 3. (4 Marks: Functionality).
    i. If the user selects the **Human** Race option, increase **all** of the Character's Abilities randomly generated from the **AbilityForm** by 5 points.
    ii. If the user selects the **Dwarf** Race option, increase the Character's **STR** and **PER** by 20 points each and decrease the Character's **CHA** by 10 points.
    iii. If the user selects the Elf Race option, increase the Character's **DEX** and **CHA** by 15.
    iv. If the user selects the Halfling Race option, increase the Character's **DEX** and **INT** by 20 and decrease the Character's **STR** by 10 points.
e. The **Next** Button opens **JobForm** and hides **RaceForm** (1 Mark: Functionality)

4. **JobForm:** (Form and Next Button provided – Subtotal **3 Marks: Functionality, 4 Marks: GUI**)
    a. Add a Group Box and several Radio Button Controls to the Form (1 Mark: GUI)
    b. The Radio Button Controls will allow the user to choose between **Soldier, Rogue, Magicker or Cultist** as the Character's **Job.** Each Radio Button Control should be Labelled accordingly. (2 Marks: GUI)
    c. Add a **Label** that displays the character's Health Points. (1 Mark: GUI)
    d. Health Points vary according to character Job and of the Abilities that were previously generated (so this will depend on the Job the user selects and an Ability Score from the **AbilityForm**) (2 Marks: Functionality)
        i. **Soldier**: 30 points + END Ability Score
        ii. **Rogue**: 28 points + DEX Ability Score
        iii. **Magicker**: 15 points + INT Ability Score
        iv. **Cultist**: 24 points + CHA Ability Score
    e. The **Next** Button Hides the **JobForm** and opens the **FinalForm** (1 Mark: Functionality)
5. **FinalForm:** (Form Provided – Controls and MenuStrip to be added – Subtotal: **11 Marks Functionality, 9 Marks: GUI**)
    a. Add a **Menu** (as shown below). (1 Mark: GUI)
    b. Add an **Exit** Button. This button will Exit the Application. (1 Mark: GUI).
    c. The **Exit** menu selection and the **Exit Button** will close the Form using a shared event handler. (1 Mark: Functionality)
    d. The **Font** Menu selection will modify the **Label** and **TextBox** controls on the Form**.** Ensure that the user can not increase the Font Size ranges between 10 and 18 points (2 Marks: Functionality)
    e. The **About** Menu will display an **AboutBox** containing the programmer's **Name** and **Student ID**. (2 Marks: Functionality)
    f. The **Print** selection will display a mock **print preview** of the Final Form (use a MessageBox) (1 Mark: Functionality)
    g. Add a **PictureBox** to this Form. The Picture box should contain the same picture reflecting the character's Race from **RaceForm** (1 Mark: GUI, 1 Mark: Functionality).
    h. Add several **Label** and **TextBox** controls that display the character's **Abilities**, his **Race**, his **Job**, and his health score that are a result of randomly generated Abilities and selections made in the **AbilityForm**, **RaceForm**, and **JobForm**. (4 Marks: GUI, 4 Mark Functionality)

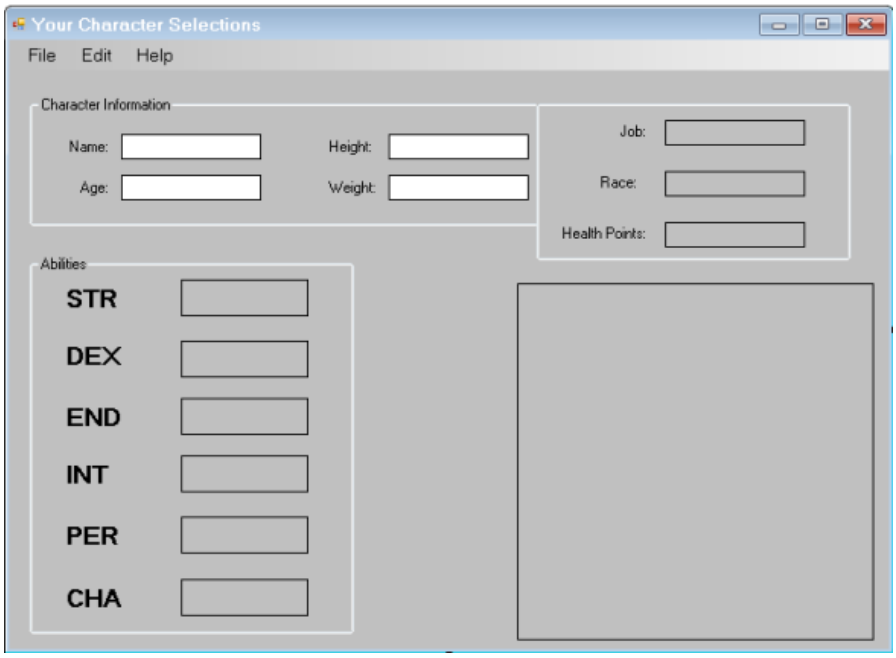
i. Add **Text Boxes** that allows the user to choose the character's **Name**, **Age**, **Height**, **Weight** and any **Titles** he's earned while adventuring. (2 Marks: GUI).

6. **Solution Structure (2 Marks: Program Structure):**
    a. Your solutions should include a Windows Forms named **`AbilityForm.cs`**, **`RaceForm.cs`** and **`JobForm.cs and FinalForm. DO NOT CHANGE THESE`**
    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).
7. Include **Internal Documentation** for your Application **(5 Marks: Internal Documentation):**
    a. Ensure you include a **comment header** for your **C# files** that indicate: the **App name**, **Author's name**, **App Creation Date** and **Student ID** (2 Marks: Internal Documentation).
    b. Ensure you include a **section header** for all of your **Event Handlers, Classes,** and any **functions** (1 Marks: Internal Documentation)
    c. Ensure all your code uses **contextual variable names** that help make the files human-readable (1 Marks: Internal Documentation).
    d. Ensure you include **inline comments** that describe your GUI Design and Functionality. (1 Mark: Internal Documentation).
8. 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).
    b. 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).

The Menu on **FinalForm**

File	Edit	Help
Print	Font	About
Exit		

A Sample **FinalForm**



SUBMITTING YOUR WORK

Your submission should include:

1. A zip archive of your project files. Your Project should be named using the following format: COMP1004-W2017-MidTermAssignment-[YourStudentID]
   (e.g. COMP1004-W2017-MidTermAssignment-200666001)
2. A link to your project files on GitHub.

EVALUATION CRITERIA

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.	17
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.	25
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.	2
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.	5
Version Control	GitHub commit history demonstrating regular updates.	4
Total		**53**

This exam is weighted **20%** of your total mark for this course.

Late submissions:

- Late Submission **will not be accepted** for this Assignment. The Mid-Term Assignment will close promptly at midnight.

External code (e.g. from the internet or other sources) cannot be used for student submissions for this assignment. However, you are allowed to use samples of your own code or any code I have shared with you on GitHub.