COMP123 – The Hero Class

Assignment 1
Implement the Hero Class

Due Week #4 (Friday June 3, 2016) @ midnight.

Value 10%

The Hero Class **Maximum Mark: 40**

Overview: Using MS Visual Studio, Build and Implement a **Hero** Class. The **Hero** class must include the following **Instance variables**, **Properties** and **Methods**:

Class
Hero
private **Instance Variables**
- _strength : int
- _speed : int
- _health : int
- _name : string
public **Properties**
+ Name : string
private **Methods**
- _generateAbilities(): void
- _hitAttempt(): bool
- _hitDamage(): int
public **Methods**
+ Fight(): void
+ Show(): void

Assignment Instructions:
(26 Marks: Functionality, 4 Marks: Program Structure, 6 Marks: Internal Documentation, 4 Marks: Version Control)

1. The Hero Class must include the following **Properties** and **Methods**: **(23 Marks: Functionality):**
    a. *Private* **Instance Variables**: **strength** (int), **speed** (int), **health** (int) and name(string) (3 Marks: Functionality).
    b. *Public* **Properties**: **Name** (string). The value of the *public* **Name** property will be stored in the *private* **name** Instance Variable (1 Marks: Functionality).
    c. The **constructor Method** should take one parameter, **name** (string) and pass its value to the *private* **name** Instance Variable. It will also call the **generateAbilities** method (3 Marks: Functionality)

d. A *private* **generateAbilities** Method randomly generates the ability numbers for the **strength**, **speed** and **health** properties. Each ability will be an integer between 1 and 100 (4 Marks: Functionality).
e. A *public* **Fight** Method calls the *private* **hitAttempt** method. If **hitAttempt** returns `true`, then it will call the *private* **hitDamage** method. The damage amount will then be **displayed** in a message on the Console. Ensure that you use appropriate spacing and padding to make the output as pretty as possible (4 Marks: Functionality).
f. A *private* **hitAttempt** Method that will randomly determine if the **Hero** hits (this should be only 20% of the time) otherwise it will return `false`. (3 Marks: Functionality).
g. A *private* **hitDamage** Method that calculates the damage the Hero causes to the target on a hit. The damage will be calculated by multiplying the Hero's strength property by a number between 1 and 6. The method will return this value. (3 Marks: Functionality).
h. A *public* **Show** Method that will display the Hero's ability scores to the console. Ensure that you use appropriate spacing and padding to make the output as pretty as possible (2 Marks: Functionality).

2. The Main method of your driver class will be structured as follows (3 Marks: Functionality):
    a. The **Main** method of your driver class (**Program**), implements the **Hero** class by creating a new **hero** object (1 Mark: Functionality).
    b. Have the **hero** object call its *public* **Show** method to display the hero's abilities. (1 Mark: Functionality).
    c. Then, have the hero object call its *public* **Fight** method to demonstrate functionality for the *private* **hitAttempt** and *private* **hitDamage** classes. Please test your output thoroughly to ensure that there are no errors. (1 Mark: Functionality).
3. Program Structure (4 Marks: Program Structure):
    a. Your "Driver" class will be named **Program.** You will use this class to call the **Hero** class (1 Mark: Program Structure).
    b. Your "Hero" class will be named **Hero**. This class will be contained in a separate file named **Hero.cs** (2 Marks: 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. Include Internal Documentation for your program **(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 (4 Marks: Documentation).
    b. Ensure your program uses contextual variable names that help make the 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).

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).

OPTIONAL PROGRAM FEATURES (Bonus Points)

- Build a unit test for the **generateAbilities** method (2 bonus points: Functionality).
- Build a unit test for the **hitAttempt** method (2 bonus points: Functionality).
- Build a unit test for the **hitDamage** method (2 bonus points: Functionality).
- Build a unit test for the **fight** method (2 bonus points: Functionality).
- Build a unit test for the **show** method (2 bonus points: Functionality).

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
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.	26
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.	4
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		**40**

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

Late submissions:

- 20% deducted for each additional day.

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.