Project #2 - FIGHT!

Learning Objectives

- Apply object-oriented programming concepts in C++

Overview

In this project, you will build a combat simulator for a simplified roll playing game (RPG). You will create a C++ class for characters in the game, including the character's name, hit points, armor class, and so forth. You will then create two character objects and simulate a combat between them.

The Character Class

You should define a C++ class called Character that implements an RPG character in the files `character.h` and `character.cpp`

Character Data

Your character class should contain the following *private* member data:

- Name (string) – the character's name.
- Role (string) – the type of character (e.g. wizard, fighter, rogue, etc.).
- Hit points (int) – this is the character's current health. When it reaches zero, the character is dead.
- Attack bonus (int) – this value is added to a character's attack roll to determine if an attack hits.
- Damage bonus (int) – this value is added to a character's damage roll when attacking.
- Armor class (int) – this value determines how hard a character is to hit.

Character member functions

Your Character class should implement, minimally, the following member functions (methods):

- `print(ostream&)` – print the character to the given ostream
- `attack(Character &otherCharacter)` – attack another character (see below for details)
- `damage(int amount)` – subtract *amount* from the character's current hit points *(Note: the minimum hit point value should be zero, so you will need to check for negative results)*
- `getHealth()` – return the character's current health
- `getName()` – return the character's name
- `getRole()` – return the character's role

Attacking another character

You should implement a member function for attacking another character. To have character1 attack character2, for example, you would call `character1.attack(character2);` When the attack member function is called, the following steps should be taken:

1. Roll a 20-sided die (i.e. generate a random number from 1 to 20) and add the attacking character's attack bonus.
2. If the result is equal to or higher than the opponent's armor class, then the attack hits. Otherwise it misses.
3. *If* the attack hits, roll a 10-sided die and add the attacking character's damage bonus. Subtract the resulting amount from the opponent's hit points using the damage() method.

Program Operation

Your main program should be implemented in the file `project2.cpp`. Your program should prompt the user for character details (name, role, hit points, etc.) for two characters. It should then simulate combat between the two characters until one of the two characters reaches zero hit points.

Example of Program Execution

```
First character name?
Uglar

Uglar's role?
Barbarian

Uglar the Barbarian's hit points?
80

Uglar the Barbarian's attack bonus?
5

Uglar the Barbarian's damage bonus?
5

Uglar the Barbarian's armor class?
24

Character summary
-----------------
Uglar the Barbarian
HP: 80
AB: 5
DB: 5
AC: 24

Second character name?
Zimzizz

Zimzizz's role?
Wizard

Zimzizz the Wizard's hit points?
40

Zimzizz the Wizard's attack bonus?
5

Zimzizz the Wizard's damage bonus?
15

Zimzizz the Wizard's armor class?
18

Character summary
-----------------
Zimzizz the Wizard
HP: 40
AB: 5
DB: 15
AC: 18

Simulated Combat:
Uglar attacks!
Attack roll: 14 + 5 = 19 --> HIT!
Damage: 9 + 5 = 14
Zimzizz has 26 hit points remaining

Zimzizz attacks!
Attack roll: 17 + 5 = 22 --> MISS!

Uglar attacks!
Attack roll: 13 + 5 = 18 --> HIT!
Damage: 8 + 5 = 13
Zimzizz has 13 hit points remaining

Zimzizz attacks!
Attack roll: 19 + 5 = 24 --> HIT!
Damage:  8 + 15 = 23
Uglar has 57 hit points remaining

Uglar attacks!
Attack roll: 16 + 5 = 21 --> HIT!
Damage: 9 + 5 = 14
Zimzizz has 0 hit points remaining

Uglar wins!
```

Turn in and Grading

Please upload your `project2.cpp`, `character.h`, and `character.cpp` files. Do not zip, archive, or compress your files.

This project is worth 10 points, distributed as follows:

Task	Points
Program compiles, runs, and produces correct output	4
The character class is correctly defined in a .h and .cpp file as described above	1
The character class includes all member data and member functions described above	2
Program conforms to coding standards (see *CS 3100 Coding Standards* in the Content/Handouts section of our class Pilot page)	3