Fantasy Combat Game

Goals

- Develop classes from program requirements
- Implement characters using polymorphism
- Implement a menu to allow the user to test the characters

In this project, we will develop a fantasy combat game. This is going to be part 1 of the game development in which we develop the characters and a menu to test the characters.

Requirements

Characters

Our game universe contains Vampire, Barbarian, Blue Men, Medusa, and Harry Potter. Each character has attributes of **attack, defense, armor, and strength points**.

The table containing the attributes data is shown below.

Note: “1d12” means rolling one 12-sided die, and “2d6” means rolling 2 6-sided dice, etc.

Type	Attack	Defense	Armor	Str
Vampire	1d12	1d6	1	18
Barbarian	2d6	2d6	0	12
Blue Men	2d10	3d6	3	12
Medusa	2d6	1d6	3	8
Harry Potter	2d6	2d6	0	10/

The characters also have their own **characteristics** as well as **special abilities**:

Type	Characterstics	Special Abilities
Vampire	Suave, debonair, but vicious and surprisingly resilient.	**Charm**: Vampires can charm an opponent into not attacking. For a given attack there is a 50% chance that their opponent does not actually attack them.
Barbarian	Think Conan or Hercules from the movies. Big sword, big muscles, bare torso.	
Blue Men	They are small, 6 inch tall, but fast and tough. They are hard to hit so they can take some damage. They can also do a LOT of damage when they crawl inside enemies' armor or clothing.	**Mob**: Blue Men are actually a swarm of small individuals. For every 4 points of damage, they lose one defense die. For example, if they have a strength of 8, they would have 2d6 for defense.
Medusa	Scrawny lady with snakes for hair which helps her during combat. Just don't look at her!	**Glare**: If a Medusa rolls a 12 when attacking then the target instantly gets turned into stone and Medusa wins! If Medusa uses Glare on Harry Potter on his first life, then Harry Potter comes back to life.
Harry Potter	Harry Potter is a wizard.	**Hogwarts**: If Harry Potter's strength reaches 0 or below, he immediately recovers and his total strength becomes 20. If he were to die again, then he's dead.

Note:

1. If Medusa uses "glare" on Harry Potter on his first life, then Harry Potter comes back to life after using "hogwarts".

2. If the Vampire's "charm" ability activates when Medusa uses "glare", the Vampire's charm trumps Medusa's glare.
3. The sample characters are unbalanced intentionally. This will help you in debugging your program! Some will win a lot, while others won't.

Gameplay

Each combat between 2 characters **ends when one of the characters die**.

Each round consists of **two attacks**, one for each character. For each attack, attacker and defender both generate dice rolls. The type and number of dice is listed above in the table.

The **actual damage inflicted** from the attacker onto the defender is calculated as follows:

Damage = attacker's roll – defender's roll – defender's armor

Then the value of that damage is subtracted from the defender's strength points.

Example: character A attacks with dice roll of: 8 10, which means it has an attack of 18, and character B defends with dice roll of 5 6, which means it has defense of 11, character B also has an armor of 3. So the actual damage inflicted from A to B is 18 – 11 – 3 = 4.

If character B has strength point of 8 during that round, the new strength point would be 8 – 4 = 4, which means next round B will have strength point of 4.

If character B has strength point of 3 during that round, the new strength point would be 3 – 4 = -1, which means character B dies.

Note: Deciding who starts attacking is your own design decision.

Class

The program should contain a **Character base class**. The base class should be an **abstract class**. All the characters should have their own subclass that inherits from the **Character** class.

Each class should only have its own information or data. For example, when A attacks B, the program should call A's **attack function**, which should return the damage attacked. Then O2's **defense function** will take the damage attacked,

and calculate the actual damage inflicted, and apply that damage to the defender’s strength points.

You can add whatever you want to the parent class but it must include an attack function and defense function. Also, the subclass should not have dependencies on which type of character the attacker is. For example, It is not an acceptable solution to have an external function check if the attacker is Medusa if a 12 is rolled; the special traits must happen within the character classes themselves.

Menu

To be able to test and play the game, write a menu. The menu should display five characters by their names, and prompt the user to select two characters to fight one another. You must account for two characters of the same type, so Vampire can fight Vampire, etc. For each round, display the results of each round on the screen so you and your grading TA can verify that calculations are correct and the game is functioning properly.

The following information must be displayed for each attack:

1. Attacker type.
2. Defender type, armor, strength point.
3. The attacker's attack dice roll.
4. The defender’s defend dice roll.
5. The total inflicted damage calculation.
6. The defender’s updated strength point amount after subtracting damage.

After the combat is over, ask users to:

1. Play again
2. Exit the game

Note: This is the **first part** of the larger project, please **do not add** any characters of your own.

Reflection Document

The requirements for the reflection document have been specified in the Project 1 document.

Make sure your document has **design descriptions, test tables, and reflections**. The reflections include: changes in design, problems encountered, and how you solve those problems. Additionally, the reflection document should include a **class hierarchy diagram**.

Suggestion

Create your design before coding anything! You should even be outlining your test plan. The grading is set up to encourage you to develop your program incrementally! Start with the base and Barbarian classes. Create the testing program that runs sample combats for you to test your code. How do you handle random die rolls when testing your code? At each step of the coding process, make notes about what worked, what has been changed from your design. Doing this will make writing the reflections easier.

Note: this project involves a lot of details and many are not specified as requirements here. In this case, they will be your design decisions. You can include these decisions in your reflection documents. If you are still not sure whether something is allowed or which is the proper way to go, post your question on Piazza and TAs will clarify it.

What you need to Submit

- All the program files including header and source files (.cpp/.hpp)
- makefile
- Your reflection pdf file

Important: Put all the files in a single .zip file and submit it on Canvas.

Grading

- Programming style and documentation (10%)
- In each of these, the virtual attack and defense functions must work correctly:
- Create the base class (10%)
- Create the Barbarian class (10%)
- Create the Vampire class, override defense function (10%)
- Create the Blue Men class (10%)
- Create the Medusa class, override attack function (10%)
- Create the Harry Potter class (10%)
- Create a test driver program to create character objects which make attack and defense rolls required to show your classes work correctly (20%)
- Reflections document to include the design description, test plan, test results, and comments about how you resolved problems during the assignment (10%)