CS_162_400_S2019
Project3
Susan Hibbert
ID# 933913975

Fantasy Combat Game

Design Description

After reading through the project specifications and 'Suggestion' section, my initial plan is to create the Character abstract base class first, then create the subclass Barbarian and test its functionality before moving onto the other Character subclasses. After writing the Barbarian subclass, I will write the menu function and test the general menu setup and input validation. Then I will try playing a game between 2 Barbarians. If I can play the game for a single round and the attack and defence rolls are as expected and the damage calculation is accurate then I will implement the gameplay (e.g. attacker and defender switch within a round, game continues until one player has negative strength points, and so on). Once this is working I will add in the other subclasses, one at a time, checking functionality and special characteristics are working before moving onto the next subclass.

After I have written and compiled the various modules of my program, I plan to implement the test cases listed in the Test Cases table to detect any runtime issues with my code.

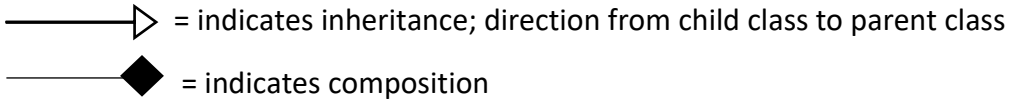
CS_162_400_S2019
Project3
Susan Hibbert
ID# 933913975

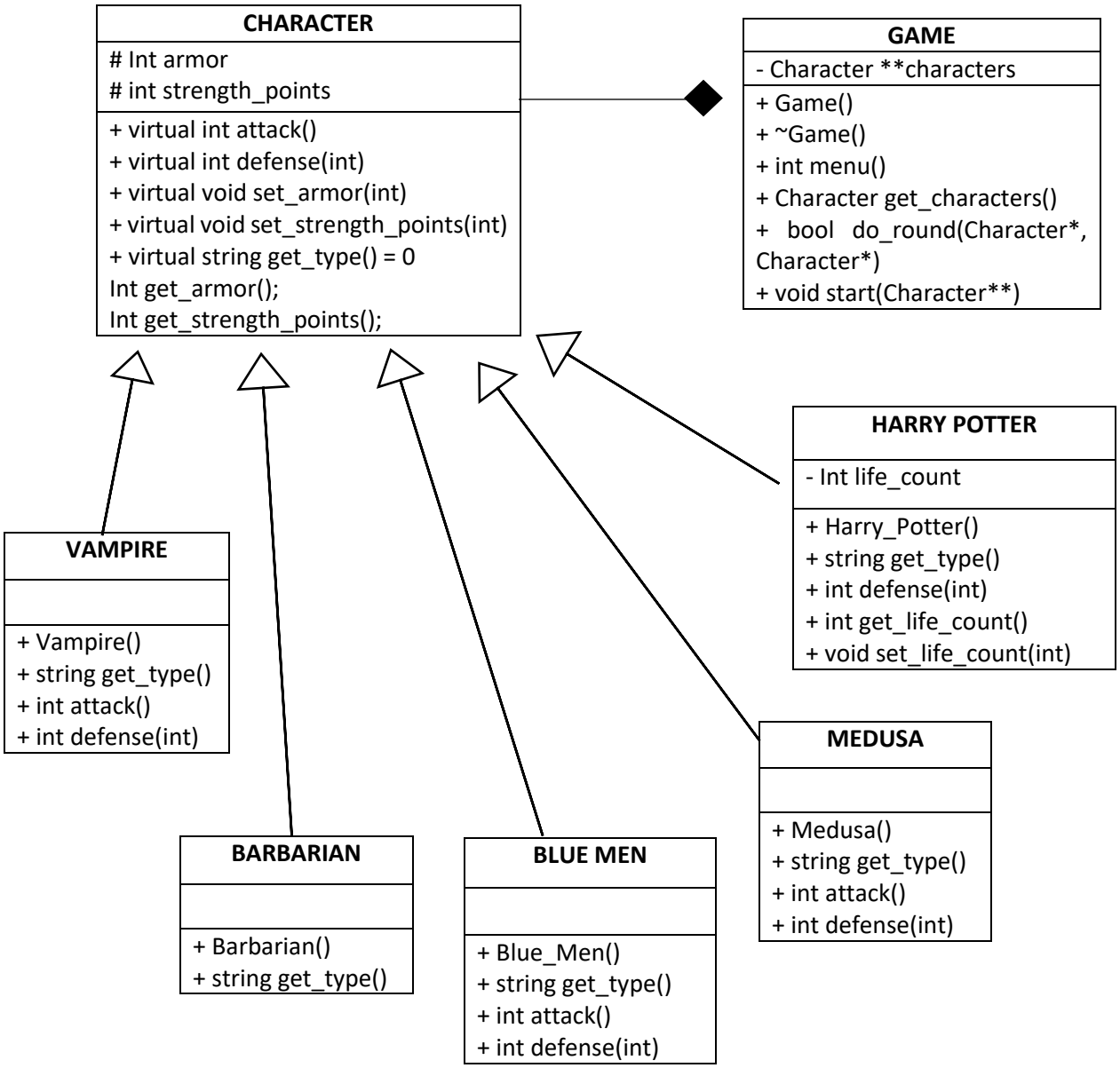
Class Hierarchy Diagram

= protected

+ = public

- = private





CS_162_400_S2019
Project3
Susan Hibbert
ID# 933913975

Test Cases

Test Case	Input Values	Driver Functions	Expected Outcomes	Observed Outcomes
START OF GAME				
User does not enter an integer when prompted to select a character at the start of the game	Input is a not an integer. May be a character, float, or a mix of characters and integers including spaces	Int_input_val() function Do-while loop prompts user to enter an integer	User is prompted to enter an integer until an integer is entered by the user	User is prompted to re-enter an integer until integer is entered. Chars, floats and input containing spaces are rejected
User enters an integer other than 1 – 5 when prompted to select a character start of game	Input is an integer other than 1-5	Display_menu() function called by menu function of Game class While loop prompts user to enter an integer between 1 and 5	Loop back to the question prompting user to enter a number between 1 and 5	Program loops back to the question prompting user to enter an integer between 1 and 5, as expected.
GAME PLAY				
Check attacker and defender switch positions during each round	n/a –check that each character gets to attack during each round	Start() function of the Game class While loop checks both characters are still alive using Bool flag before switching their attack positions	Attacking and defending characters switch positions within the round, as displayed in the stats after each character attack/defense	Attacking and defending characters switch positions within the round and each character gets a chance to attack, as expected
Check the first character to attack is chosen at random	n/a – check the character chosen to attack first is chosen at random	Start() function of Game class Rand() function and if statements	The attacking character is not always the first or second opponent and it differs between games, as displayed in the stats after the first attack of the game	The attacking character is not always the first or second opponent and it differs between games as expected, for each character type

CS_162_400_S2019
Project3
Susan Hibbert
ID# 933913975

Check damage calculation is correct following an attack	n/a – check the correct amount of damage is subtracted from the defending character's strength points. Damage cannot be a negative number	Default Defense() function of Character class used by Barbarian and each overridden Defense() function of the Character subclasses If statement to check if damage is a negative number	Damage is calculated as follows: Attack roll – defense roll – defender's armor. If damage is negative it is set to 0. Damage is displayed in the stats after each character attack/defense	Damage is calculated correctly for each character. If negative damage is incurred it is set to 0 and the defending character's strength points do not change, as expected
Check defender's strength points are updated correctly following an attack	n/a – check the defending character's strength points are updated after an attack with the result of the damage. Strength points cannot be a negative number. If strength points are 0 the character dies and the game ends	Default Defense() function of Character class used by Barbarian and each overridden Defense() function of the Character subclasses If statement to check if updated strength points is a negative number Do_round() function of the Game class If statement checks if defending character's strength points equal 0	Defender's updated strength points is calculated as follows: Defender's strength points – damage. If the updated strength points are negative they are set to 0 and the character dies, ending the game. The defender's updated strength points are displayed in the stats after each character attack/defense	Defender's strength points are updated correctly following an attack for each type of character. If the updated strength points are negative they are set to 0 and the character dies, ending the game as expected
Check each character starts off with its correct strength points and armor as listed in the attributes data table	n/a – check stats of each opponent when displayed at the start of the game and after every attack	Default constructor of each Character subclass	Each character starts off with its correct strength points and armor as displayed in the stats after the first attack	Each character starts off with its correct strength points and armor, as expected

CS_162_400_S2019
Project3
Susan Hibbert
ID# 933913975

Check attack and defence dice rolls for each character are correct as listed in the attributes data table	n/a – each character uses different combinations and types of defence and attack die as per the attributes data table	Default Defense() function used by Barbarian and default attack() function used by Barbarian and Harry Potter of Character class Overridden defense() functions and overridden attack() functions for each Character subclass	Each character type rolls the attack and defense die specific to its type, as displayed in the stats after each character attack/defense	Each character type rolls the attack and defense die specific to its type, as displayed in the stats after each character attack/defense, as expected
Ensure rounds continue while both characters have positive strength points	n/a – check that the combat between the opponents continues until one of them dies, represented by a strength point of 0 following an attack	Do_round() function of Game class If statement, bool flag Start() function of Game class While loop	Combat between the two opponents continues, round by round, until one of them loses all their strength points and dies. A character's current strength points is displayed in the stats after each character attack/defense	With each character type, combat between the two opponents continues round by round until one of them loses all their strength points and dies, as expected
Check Charm special ability of Vampire	n/a – check that for each attack there is a 50% chance of Charm activating, resulting in their opponent being unable to attack them and a damage of 0	Overridden defense() function of Vampire class – rand() function to implement charm and if statement	Vampire's defense roll with Charm activated will always be higher than opponent's attack roll, offsetting any damage inflicted by the opponent's attack and damage will be 0. The defense roll and damage inflicted is displayed in the stats after each	With each character type, when Charm is activated the Vampire's defense roll is very high and offsets the opponent's attack roll, resulting in no damage being inflicted to the Vampire and its strength points remain unchanged

CS_162_400_S2019
Project3
Susan Hibbert
ID# 933913975

			character attack/defense	
Check Glare special ability of Medusa	n/a – check that when Medusa rolls a 12 when attacking the opponent dies/ instantly gets turned into stone and Medusa wins	Overridden attack() function of Medusa class – rand() functions to calculate dice roll and if statement	Medusa's attack roll with Glare activated will be very high, inflicting a lot of damage to the opponent. This will reduce the defending character's strength points to 0 and kill them (i.e. turn them to stone), allowing Medusa to win the game. The attack roll and damage inflicted is displayed in the stats after each character attack/defense	With each character type, when Glare is activated Medusa's attack roll is very high and inflicts a lot of damage to the opponent, resulting in their strength points dropping below zero thus killing them and Medusa wins the game
Check Vampire's Charm trumps Medusa's Glare	n/a - When a Vampire and a Medusa are fighting, and the Vampire's Charm ability activates when Medusa uses Glare, the Vampire's Charm trumps Medusa's Glare	Overridden defense() function of Vampire class – rand() function to implement charm and if statement Overridden attack() function of Medusa class – rand() functions to calculate dice roll and if statement	Vampire's defense roll with Charm activated will always be higher than Medusa's attack roll with Glare activated, offsetting any damage inflicted by Medusa's attack and damage will be 0. Ultimately Vampire will win. Defense roll, attack roll and damage inflicted is displayed in the stats after each character attack/defense	To test this, I hard coded Vampire's defense function so Charm would be activated at every round, and I hard coded Medusa's attack function so Glare would be activated at every round. Vampire's Charm always trumps Medusa's Glare and Vampire wins, as expected

CS_162_400_S2019
Project3
Susan Hibbert
ID# 933913975

Check Hogwarts special ability of Harry Potter	n/a – check Hogwarts is implemented when Harry Potter's strength points are less than or equal to 0. Check strength points is set to 20 after Hogwarts is activated. Ensure he cannot use Hogwarts and come back to life more than once	Overridden defense() function of Harry_Potter class If statement	Hogwarts is only implemented when he has not used it before and his strength points are less than or equal to 0. He comes back to life with strength points of 20. If Medusa uses Glare on Harry Potter on his first life, then Harry Potter comes back to life after using Hogwarts. If he were to die again, then he's dead. Strength points are displayed in the stats after each character attack/defense	Against each character type, when Harry Potter's strength reaches 0 or below, he immediately recovers using Hogwarts and his strength points are set to 20. When Medusa uses Glare on Harry Potter on his first life, Harry Potter comes back to life after using Hogwarts. If he dies again during the same game, he actually dies and the opponent wins, as expected
Check Mob special ability of Blue Men	n/a – check that for every 4 points of damage, Blue Men lose one defense die	Overridden defense() function of Blue_Men class if statement	Blue Men start with 3d6 and strength points of 12. When they lose 4 strength points resulting in a strength point of 8 they should have 2d6. When they lose another 4 strength points resulting in a strength point of 4 they should have 1d6. When they lose another 4 strength points (resulting in a strength point of 0) they die. Their strength points	Blue Men start with 3d6 and strength points of 12. At a strength point of 8 they have 2d6, and at a strength point of 4 they have 1d6. When they lose another 4 strength points (resulting in a strength point of 0) they die and the opponent wins. This is true when Blue Men play against all character types including another Blue Men character

			can be tracked in the stats that display after each character attack/defense	
END OF GAME				
User does not enter an integer when prompted to play again or exit program at end of game	Input is a not an integer. May be a character, float, or a mix of characters and integers including spaces	Int_input_val() function Do-while loop prompts user to enter an integer	User is prompted to enter an integer until an integer is entered by the user	User is prompted to re-enter an integer until integer is entered. Chars, floats and input containing spaces are rejected
User enters an integer other than 1 or 2 when asked whether they would like to play again or exit at the end of the game	Input is an integer other than 1 or 2	Display_menu() function called by menu function of Game class While loop prompts user to enter 1 or 2	Loop back to the question prompting user to enter 1 or 2	Program loops back to the question prompting user to enter 1 or 2, as expected.

CS_162_400_S2019
Project3
Susan Hibbert
ID# 933913975

Reflection

In my initial design, I described how I would implement and test the gameplay with the Barbarian subclass (e.g. attacker and defender switch within a round, game continues until one player has negative strength points, and so on) before creating the other Character subclasses. I ended up running into some issues with gameplay, for instance, I could not work out how to switch between attacker and defender within each round. Due to this, I ended up writing all my Character subclasses before fully testing the combat aspect of the game. In hindsight this does not seem like good practice and I should have resolved the issues I had with the test driver program before introducing any new elements to it.

In addition to the above, I encountered some other problems during the assignment. Firstly, I was having an issue where I could not choose characters of 2 different types. Whatever character I chose for the first opponent was automatically selected for the second opponent, regardless of what choice of character I had made for my second opponent. It took me quite a while to identify what the problem was. The issue was that I was passing valid_choice1, the variable representing the user's first choice of character, into the if statements determining the user's second choice of character, when I should have been passing valid_choice2. If my variable names had not been so similar it may not have taken me as long to identify the issue, and for future assignments I will try to chose more distinct variable names for debugging purposes.

I had some confusion over where to implement the damage calculation. Initially I had written a separate function called damage which calculated and returned the attack damage and took the attack roll and defense roll as parameters. However, I ran into problems when trying to code the Mob special ability of the Blue Men subclass. I posted a question to a teaching assistant on Slack who clarified that the defense calculation should be a part of the defense function of the Character class which takes the attack roll as a parameter. I had to re-write a lot of my code but after I made this change it was much easier to override and implement in the subclasses.