Activity No. 2	
Laboratory Activity 2 - Inheritance, Encapsulation, Abstraction	
Course Code: CPE009B	**Program:** Computer Engineering
Course Title: Object-Oriented Programming 2	**Date Performed:** 9/27/24
Section: CPE21S4	**Date Submitted:** 9/29/24
Name(s): Santos, Andrei R.	**Instructor:** Professor Maria Rizette Sayo

5. Procedure

Character.py

```python
class Character():
    def __init__(self, username):
        self.__username = username
        self.__hp = 100
        self.__mana = 100
        self.__damage = 5
        self.__str = 0
        self.__vit = 0
        self.__int = 0
        self.__agi = 0

    def getUsername(self):
        return self.__username

    def setUsername(self, new_username):
        self.__username = new_username

    def getHp(self):
        return self.__hp

    def setHp(self,new_hp):
        self.__hp = new_hp

    def getDamage(self):
        return self.__damage

    def setDamage(self, new_damage):
        self.__damage = new_damage

    def getStr(self):
        return self.__str

    def setStr(self, new_str):
        self.__str = new_str

    def getVit(self):
        return self.__vit

    def setVit(self, new_vit):
        self.__vit = new_vit

    def getInt(self):
        return self.__int

    def setInt(self, new_int):
        self.__int = new_int

    def getAgi(self):
        return self.__agi

    def setAgi(self, new_agi):
        self.__agi = new_agi

    def reduceHp(self, damage_amount):
        self.__hp = self.__hp - damage_amount

    def addHp(self, heal_amount):
        self.__hp = self.__hp + heal_amount

```

```
In [14]: %runcell -i 0 'C:/Users/Andrei/Documents/oopfa1_Santos, A/Character.py'
```

Character.py (w/ the added under code)

```python
class Character():
    def __init__(self, username):
        self.__username = username
        self.__hp = 100
        self.__mana = 100
        self.__damage = 5
        self.__str = 0
        self.__vit = 0
        self.__int = 0
        self.__agi = 0

    def getUsername(self):
        return self.__username

    def setUsername(self, new_username):
        self.__username = new_username

    def getHp(self):
        return self.__hp

    def setHp(self,new_hp):
        self.__hp = new_hp

    def getDamage(self):
        return self.__damage

    def setDamage(self, new_damage):
        self.__damage = new_damage

    def getStr(self):
        return self.__str

    def setStr(self, new_str):
        self.__str = new_str

    def getVit(self):
        return self.__vit

    def setVit(self, new_vit):
        self.__vit = new_vit

    def getInt(self):
        return self.__int

    def setInt(self, new_int):
        self.__int = new_int

    def getAgi(self):
        return self.__agi

    def setAgi(self, new_agi):
        self.__agi = new_agi

    def reduceHp(self, damage_amount):
        self.__hp = self.__hp - damage_amount

    def addHp(self, heal_amount):
        self.__hp = self.__hp + heal_amount

        character1 = Character("Your Username")
        print(character1._username)
        print(character1.getUsername())
```

```
In [16]: %runcell -i 0 'C:/Users/Andrei/Documents/oopfa1_Santos, A/Character.py'
```

Novice.py

```python
from Character import Character

class Novice(Character):
    pass
    def basicAttack(self, character):
        character.reduceHp(self.getDamage())
        print(f"{self.getUsername()} performed Basic Attack! -{self.getDamage()}")
```

```
In [17]: %runcell -i 0 'C:/
Users/Andrei/Documents/
oopfa1_Santos, A/Novice.py'
```

Novice.py (w/ the added under code)

```python
from Character import Character

class Novice(Character):
    pass
    def basicAttack(self, character):
        character.reduceHp(self.getDamage())
        print(f"{self.getUsername()} performed Basic Attack! -{self.getDamage()}")

        character1 = Novice("Your Username")
        print(character1.getUsername)
        print(character1.Hp())
```

```
In [23]: %runcell -i 0 'C:/
Users/Andrei/Documents/
oopfa1_Santos, A/Novice.py'
```

Swordman.py

```python
from Novice import Novice

class Swordman(Novice):
    def __init__(self, username):
        super().__init__(username)
        self.setStr(5)
        self.setVit(10)
        self.setHp(self.getHp() +self.getVit())

    def slashAttack(self,character):
        self.new_damage = self.getDamage() + self.getStr()
        character.reduceHp(self.new_damage)
        print(f"{self.getUsername()} performed Slash Attack! -{self.new_damage}")
```

```
In [26]: %runcell -i 0 'C:/
Users/Andrei/Documents/
oopfa1_Santos, A/Swordman.py'
```

Archer.py

```python
from Novice import Novice
import random

class Archer(Novice):
    def __int__(self, username):
        super().__init__(username)
        self.setAgi(5)
        self.setInt(5)
        self.setVit(5)
        self.setHp(self.getHp() + self.getVit())

    def rangedAttack(self, character):
        self.new_damage = self.getDamage() + random.randint(0, self.getInt())
        print(f"{self.getUsername()} performed Slash Attack! -{self.new_damage}")
```

```
In [27]: %runcell -i 0 'C:/
Users/Andrei/Documents/
oopfa1_Santos, A/Archer.py'
```

Magician.py

```python
from Novice import Novice

class Magician(Novice):
    def __init__(self, username):
        super().__init__(username)
        self.setInt(10)
        self.setVit(5)
        self.setHp(self.getHp() + self.getVit())

    def heal(self):
        self.addHp(self.getInt())
        print(f"{self.getUsername()} Performed Heal! +{self.getInt()}")

    def magicAttack(self, character):
        self.new_damage = self.getDamage() + self.getInt()
        character.reduceHp(self.new_damage)
        print(f"{self.getUsername()} performed Magic Attack! -{self.new_damage}")
```

```
In [28]: %runcell -i 0 'C:/
Users/Andrei/Documents/
oopfa1_Santos, A/Magician.py'
```

Test.py

```python
from Swordman import Swordman
from Archer import Archer
from Magician import Magician

Character1 = Swordman("Royce")
Character2 = Magician("Archie")
print(f"{Character1.getUsername()} HP: {Character1.getHp()}")
print(f"{Character2.getUsername()} HP: {Character2.getHp()}")

Character1.slashAttack(Character2)
Character1.basicAttack(Character2)

print(f"{Character1.getUsername()} HP: {Character1.getHp()}")
print(f"{Character2.getUsername()} HP: {Character2.getHp()}")

Character2.heal()
Character2.magicAttack(Character1)

print(f"{Character1.getUsername()} HP: {Character1.getHp()}")
print(f"{Character2.getUsername()} HP: {Character2.getHp()}")
```

```
In [37]: %runcell -i 0 'C:/
Users/Andrei/Documents/
oopfa1_Santos, A/Test.py'
Royce HP: 110
Archie HP: 105
Royce performed Slash Attack!
-10
Royce performed Basic Attack!
-5
Royce HP: 110
Archie HP: 90
Archie Performed Heal! +10
Archie performed Magic
Attack! -15
Royce HP: 95
Archie HP: 100
```

Test.py (changed the magic attack to slash attack)

```python
from Swordman import Swordman
from Archer import Archer
from Magician import Magician

Character1 = Swordman("Royce")
Character2 = Magician("Archie")
print(f"{Character1.getUsername()} HP: {Character1.getHp()}")
print(f"{Character2.getUsername()} HP: {Character2.getHp()}")

Character1.slashAttack(Character2)
Character1.basicAttack(Character2)

print(f"{Character1.getUsername()} HP: {Character1.getHp()}")
print(f"{Character2.getUsername()} HP: {Character2.getHp()}")

Character2.heal()
Character2.slashAttack(Character1)

print(f"{Character1.getUsername()} HP: {Character1.getHp()}")
print(f"{Character2.getUsername()} HP: {Character2.getHp()}")
```

```
  File ~\AppData\Local\spyder-6\Spyder-6-new\envs\spyder-
runtime\Lib\site-packages\spyder_kernels\customize\utils.py:209
in exec_encapsulate_locals
    exec_fun(compile(code_ast, filename, "exec"), globals)

  File c:\users\andrei\documents\oopfa1_santos, a\test.py:17
    Character2.slashAttack(Character1)

AttributeError: 'Magician' object has no attribute
'slashAttack'
```

(Note : There is no slash attack given from the magician, that is why it is an error.)

Boss.py

```python
from Swordman import Swordman
from Archer import Archer
from Magician import Magician

class Boss(Swordman, Archer, Magician):
    def __init__(self, username):
        super().__init__(username)
        self.setStr(10)
        self.setVit(25)
        self.setInt(5)
        self.setHp(self.getHp() + self.getVit())
```

```
In [41]: %runcell -i 0 'C:/
Users/Andrei/Documents/
oopfa1_Santos, A/Boss.py'
```

Test.py (Final Output)

```python
from Swordman import Swordman
from Archer import Archer
from Magician import Magician
from Boss import Boss

Character1 = Swordman("Royce")
Character2 = Boss("Archie")

print(f"{Character1.getUsername()} HP: {Character1.getHp()}")
print(f"{Character2.getUsername()} HP: {Character2.getHp()}")

Character1.slashAttack(Character2)
Character1.basicAttack(Character2)

print(f"{Character1.getUsername()} HP: {Character1.getHp()}")
print(f"{Character2.getUsername()} HP: {Character2.getHp()}")

Character2.heal()
Character2.basicAttack(Character1)
Character2.slashAttack(Character1)
Character2.rangedAttack(Character1)
Character2.magicAttack(Character1)

print(f"{Character1.getUsername()} HP: {Character1.getHp()}")
print(f"{Character2.getUsername()} HP: {Character2.getHp()}")
```

```
In [46]: %runfile 'C:/Users/Andrei/Documents/oopfa1_Santos, A/
Test.py' --wdir
Reloaded modules: Character, Novice, Swordman, Archer,
Magician, Boss
Royce HP: 110
Archie HP: 140
Royce performed Slash Attack! -10
Royce performed Basic Attack! -5
Royce HP: 110
Archie HP: 125
Archie Performed Heal! +5
Archie performed Basic Attack! -5
Archie performed Slash Attack! -15
Archie performed Slash Attack! -9
Archie performed Magic Attack! -10
Royce HP: 80
Archie HP: 130
```

6. Supplementary Activity

```python
from Novice import Novice
from Swordman import Swordman
from Archer import Archer
from Magician import Magician
from Boss import Boss
import random


class Game:
    def __init__(self):
        self.wins_single_player = 0
        self.wins_player_vs_player = 0

    def select(self):
        while True:  # 1 and 2 only accepts to go next
            mode = input("\n(1) for Single Player \n(2) for Player vs Player \nSelect mode: ")
            if mode == "1":
                self.single_player()
                break  # After playing, it will exit since it is only single player mode
            elif mode == "2":
                self.player_vs_player()
                break  # It will break if there is already a winner
            else:
                print("Incorrect mode, please try again.") # If error mode was enetered, this line will be printed

    def single_player(self): #For single player mode
        player = Novice("Player")
        opponent = Boss("Monster")

        while True:
            self.play_match(player, opponent)

            if opponent.getHp() <= 0:
                print(f"{player.getUsername()} is the winner. Nice!")
                self.wins_single_player += 1
                if self.wins_single_player >= 2:
                    print("Congratulations, you leveled up! You can now choose different roles!")
                    player = self.select_role("Player")
                opponent = Boss("Monster")  # Reset opponent HP for the next match
            else:
                print(f"{opponent.getUsername()} wins. Try again, and keep it up!")
                break
```

```python

    def player_vs_player(self): #For player vs player mode
        player1_name = input("Enter name for player 1: ")
        player2_name = input("Enter name for player 2: ")
        player1 = self.select_role(player1_name)
        player2 = self.select_role(player2_name)

        while True:
            self.play_match(player1, player2)
            if player1.getHp() <= 0:
                print(f"{player2.getUsername()} is the winner!")
                self.wins_player_vs_player += 1
                break
            elif player2.getHp() <= 0:
                print(f"{player1.getUsername()} is the winner!")
                self.wins_player_vs_player += 1
                break

    def select_role(self, name):
        print(f"{name}, select a role: \n1. Swordsman \n2. Archer \n3. Magician")
        role_choice = input("Enter your chosen role: ")

        if role_choice == "1":
            return Swordman(name)
        elif role_choice == "2":
            return Archer(name)
        elif role_choice == "3":
            return Magician(name)
        else:
            print("Invalid choice of role, your role : Novice.")
            return Novice(name)

    def play_match(self, player1, player2):
        players = [player1, player2]
        random.shuffle(players)
        current_player = players[0]
        opponent = players[1]

        while player1.getHp() > 0 and player2.getHp() > 0:
            damage = random.randint(5, 10)  # Random damage number between  and 10
            opponent.setHp(opponent.getHp() - damage)  # The updated health / hitpoints of the player
            print(f"{current_player.getUsername()} attacks {opponent.getUsername()} for {damage} damage!")
            print(f"{opponent.getUsername()} HP: {opponent.getHp()}")

            # Switch turns
            current_player, opponent = opponent, current_player

    def play(self):
        while True:
            self.select()

if __name__ == "__main__":
    game = Game()
    game.play()

```

Input and output for single player mode:

```
(1) for Single Player
(2) for Player vs Player
Select mode: 1
Monster attacks Player for 7 damage!
Player HP: 93
Player attacks Monster for 8 damage!
Monster HP: 132
Monster attacks Player for 9 damage!
Player HP: 84
Player attacks Monster for 6 damage!
Monster HP: 126
Monster attacks Player for 10 damage!
Player HP: 74
Player attacks Monster for 8 damage!
Monster HP: 118
Monster attacks Player for 10 damage!
Player HP: 64
Player attacks Monster for 5 damage!
Monster HP: 113
Monster attacks Player for 7 damage!
Player HP: 57
Player attacks Monster for 5 damage!
Monster HP: 108
```

```
Monster attacks Player for 10 damage!
Player HP: 47
Player attacks Monster for 8 damage!
Monster HP: 100
Monster attacks Player for 5 damage!
Player HP: 42
Player attacks Monster for 5 damage!
Monster HP: 95
Monster attacks Player for 8 damage!
Player HP: 34
Player attacks Monster for 7 damage!
Monster HP: 88
Monster attacks Player for 8 damage!
Player HP: 26
Player attacks Monster for 8 damage!
Monster HP: 80
Monster attacks Player for 9 damage!
Player HP: 17
Player attacks Monster for 8 damage!
Monster HP: 72
Monster attacks Player for 10 damage!
Player HP: 7
Player attacks Monster for 7 damage!
Monster HP: 65
```

```
Monster attacks Player for 8 damage!
Player HP: -1
Monster wins. Try again, and keep it up!
```

Input and output for player vs player mode:

```
(1) for Single Player
(2) for Player vs Player
Select mode: 2
Enter name for player 1: Andrei
Enter name for player 2: Tiffany
Andrei, select a role:
1. Swordsman
2. Archer
3. Magician
Enter your chosen role: 1
Tiffany, select a role:
1. Swordsman
2. Archer
3. Magician
Enter your chosen role: 2
```

```
Andrei attacks Tiffany for 8 damage!
Tiffany HP: 46
Tiffany attacks Andrei for 5 damage!
Andrei HP: 59
Andrei attacks Tiffany for 6 damage!
Tiffany HP: 40
Tiffany attacks Andrei for 8 damage!
Andrei HP: 51
Andrei attacks Tiffany for 9 damage!
Tiffany HP: 31
Tiffany attacks Andrei for 9 damage!
Andrei HP: 42
Andrei attacks Tiffany for 5 damage!
Tiffany HP: 26
Tiffany attacks Andrei for 7 damage!
Andrei HP: 35
Andrei attacks Tiffany for 6 damage!
Tiffany HP: 20
Tiffany attacks Andrei for 5 damage!
Andrei HP: 30
Andrei attacks Tiffany for 7 damage!
Tiffany HP: 13
Tiffany attacks Andrei for 5 damage!
```

```
Andrei attacks Tiffany for 6 damage!
Tiffany HP: 94
Tiffany attacks Andrei for 9 damage!
Andrei HP: 101
Andrei attacks Tiffany for 10 damage!
Tiffany HP: 84
Tiffany attacks Andrei for 7 damage!
Andrei HP: 94
Andrei attacks Tiffany for 9 damage!
Tiffany HP: 75
Tiffany attacks Andrei for 10 damage!
Andrei HP: 84
Andrei attacks Tiffany for 10 damage!
Tiffany HP: 65
Tiffany attacks Andrei for 6 damage!
Andrei HP: 78
Andrei attacks Tiffany for 6 damage!
Tiffany HP: 59
Tiffany attacks Andrei for 6 damage!
Andrei HP: 72
Andrei attacks Tiffany for 5 damage!
Tiffany HP: 54
Tiffany attacks Andrei for 8 damage!
```

```
Andrei HP: 25
Andrei attacks Tiffany for 6 damage!
Tiffany HP: 7
Tiffany attacks Andrei for 6 damage!
Andrei HP: 19
Andrei attacks Tiffany for 7 damage!
Tiffany HP: 0
Andrei is the winner!
```

Questions

1. **Why is inheritance important?**

Inheritance comes from the word "inherit," which sums up connecting one thing to another. Inheritance simply connects one class to another, making coding simple, neat, clean, and convenient. It allows us to reuse code and avoid redundancy, which saves time compared to creating multiple classes. This is why inheritance is helpful in coding.

2. **Explain the advantages and disadvantages of using applying inheritance in an Object-Oriented Programming.**

The advantages of inheritance in object-oriented programming are code reusability, flexibility in changing data after running the code, and automatic updates in each class when changes are made to the base or overall code. However, as I mentioned earlier, changes to the base code can override data, and we need to re-run the code after each change, such as in test.py. Inheritance can also introduce complexity, making the code harder to maintain as more features are added.

3. **Differentiate Single inheritance, multiple inheritance, and multi-level inheritance.**

Single inheritance involves inheriting data from one class, while multiple inheritance allows a class to inherit from two or more classes, gathering more information. Multilevel inheritance works like a chain, where the first class inherits from the next class, and the second class inherits from another class.

4. **Why is super().__init__(username) added in the codes of Swordman, Archer, Magician, and Boss?**

As we tackle some learnings in the last term, super().__init__(username) is like a superclass (parent class) provides what is inherited by subclasses (child classes). The `super()` function allows the subclass to access and initialize attributes from the superclass before adding its own specific functionality. This promotes code reuse, reduces redundancy, and ensures that changes in the superclass apply consistently across all the subclasses.

5. **How do you think Encapsulation and Abstraction helps in making good Object-Oriented Programs?**

Encapsulation involves grouping data within a class and improving security, as we learned with the use of single (`_`) and double underscores (`__`) for private variables. This is why some code doesn't run outside the class. Encapsulation keeps errors contained within the class, protecting other parts of the code. Abstraction focuses on showing only the necessary details of an object, making the code simple, reusable, and clear.

7. Conclusion

In conclusion, I have learned that inheritance, encapsulation, and abstraction can add layers of complexity to coding, especially in Object-Oriented Programming (OOP). Understanding these concepts is crucial as they help organize and simplify code by defining relationships between classes. The three types of inheritance are single, multiple, and multilevel which each serve different purposes in how properties and behaviors are passed between classes. Additionally, the superclass and subclass play significant roles in modifying and extending code functionality. Applying these OOP learnings not only makes the code more efficient but also enhances clarity, making complex systems easier to manage once understood thoroughly.

8. Assessment Rubric