**INFSCI 0201 (Spring 2024)**

**Lab 6: OOP Project**

This week’s lab builds on the object-oriented programming approach you learned last week. **Lab Report**

\*) Post the GitHub folder URL to Canvas

Student 1: \_\_\_\_\_\_

Student 2: \_\_\_\_\_\_

*\*) Both students can submit the same report to Canvas*

GitHub project folder Url: \_\_\_\_\_

*\*) For all labs, you can use monorepo or a single project in GitHub. Submit the folder, e.g., https://github.com/numpy/numpy/tree/main/numpy*

## Overview

You must 1) write a Python program that meets the requirements outlined in the Tasks section and 2) answer reflection questions related to the program.

* Suggested filename for the project **/labs/lab06/oop\_project.py**
  + You can also organize the project into multiple files and use “import”
* Suggested filename for the reflection questions **/labs/lab06/reflection.md**

Study material

* Watch and discuss with your teammate; feel free to ask the instructor or TA to clarify.
  + [Learn Python Classes With a Text-Based Battle - OOP Tutorial](https://www.youtube.com/watch?v=cM_ocyOrs_k&ab_channel=OrkSlayerGamedev)

## Expected Output for the Project

The **oop\_project.py** should cover the following operations:

1. A “protagonist” object from “Hero” class, inherited from “Character” class
2. An “ally” object from “Enemy” class, inherited from “Character” class
3. A “empty\_handed” object from “Weapon” class that do nothing on each turn
4. A “healing\_staff” object from “Weapon” class that increase health with 3 points on each turn.
5. “protagonist” uses “empty\_handed”, while “ally” uses “healing\_staff”

## Expected Output for the Reflection

1. Create **reflection.md** file
2. Answer these questions:
   1. Explain the concept of **inheritance** that is being demonstrated in the project.
   2. Explain the concept of **association** (this can be **aggregation** or **composition**) being demonstrated in the project.  
      <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-aggregation-vs-composition/>
   3. Explain the concept of **encapsulation** that is being (or should be) demonstrated in the project.

### Project Tasks

Be careful when copy-pasting the code below; it might contain errors, and Microsoft Word automatically converts quotes. The code serves as instruction, not part of the solution.

Setup the game using these functions, based on the instructions 1-5:

empty\_handed = Weapon(name="Empty handed",

weapon\_type="dummy",

damage=0,

value=0)

healing\_staff = Weapon(name="Healing Staff",

weapon\_type="magic",

damage=-3,

value=0)

protagonist = Hero(name="Hero", health=100)

protagonist.health = 10

protagonist.equip(empty\_handed)

ally = Enemy(name="Friendly Enemy", health=100, weapon=healing\_staff)

The main game loop should be similar to the example.

Git Hub Url:

1.

Inheritance in the Project: In the provided code, inheritance is demonstrated through the Hero and Enemy classes, which inherit from the Character class. This allows both Hero and Enemy to share common properties and methods (such as name, health, equip, and take\_turn) that are defined in the Character

2.Association in the Project: The concept of association is demonstrated by the relationship between the Character and Weapon classes. Characters have a relationship with a Weapon through the weapon attribute within the Character class. This is a form of composition, as characters own their weapons, meaning that a Weapon object is part of a Character object.

3.Encapsulation in the Project: Encapsulation is exhibited through the use of class definitions to bundle attributes and methods that operate on the data into a single class. Access to this data is typically provided through methods like equip() and is controlled in a way that internal states of objects are hidden from the outside, only allowing manipulation through the master class methods.