Lab 5: Objects Relationship

INFSCI 0201
Intermediate Programming (Spring 2025)

The Alpha test for the RPG game you helped develop last week was quite successful, especially regarding the item systems in the Alpha test. As the leading developer for the item system, you are promoted to take on more responsibilities regarding the items. In this week's lab, the design team has some new requirements based on the feedback in the Alpha test. At the same time, your new responsibilities ask you to develop the inventory system (backpack). These tasks will test your knowledge of object relationships and polymorphism.

Submission Details

You are to extend your work in Lab 4 and continue writing a Python program that meets the requirements outlined in the Lab 5 Tasks section.

- As with all programming assignments in this course, you must submit Lab 5 through GitHub in the git repository the instructor created for you. You must place your Lab 5 eclipse project in the folder **/labs/lab05/**
- If you did this all correctly, this project will be in the directory **/labs/lab05/**
- **Remember to also put your code from Lab 4 in this week's directory since you are going to use them.**

Testing

There will be four main tests for this lab, which cover the following operations:

1. The legendary showcase method is correctly implemented.
2. New weapon subclasses are fully implemented according to the requirement
3. The inventory classes have their full functionality
4. The relationship between the inventory and items is implemented properly.

Note: Be sure you're strictly adhering to the Python coding style guide on Canvas.

Lab 4 task

There are two parts of the work you need to do in this week's lab. First, you need to make some modifications to the item system from your work in last week's lab since the design team and the project manager have come up with some new ideas.

- When legendary items are viewed (when the player character inspects the item or shows it off to the friend), it should produce some unique presentation style:
    - In this case, you can use ASCII art, or you can construct a special message.
    - Ultimately, the design team thinks we should make these legendary items stand out from others so that players will pursue them for their flashing presentation and larger numbers.
    - **Hint**: when printing out (using __str__(). method) the legendary items, they should output some unique strings to indicate that they are legendary items.

The animation team wants to rework how weapon items work. They want to have more detailed weapon type separation. They want the following weapon types:

    - Single-handed weapon
    - Double-handed weapon
    - Pike
    - Ranged weapon
- These different types are still items and should also be considered weapons, but each will have its unique attack animations and character movement set.
    - When using the weapon, the method for the weapon's unique attack movement will be called first, and the damage message follows.
    - This week, because you have a lot on your plate, the animation team only asks you to return a string describing the one unique attack animation of the specific weapon.
    - In the future, each weapon type might have a set of unique attack movement
    - **Weapons should have an attack_move() method (possibly with some arguments) as an interface to access these unique attack movements.**
- Upon consulting with the design team, they have the name for the first unique attack movement:
    - Single-handed weapon => _slash()
    - Double-handed weapon => _spin()
    - Pike => _thrust()
    - Ranged weapon => _shoot()

Now that you have managed to fulfill the requirements of other teams. It's time to look at the Inventory system.

- Each player has one inventory named "backpack". The inventory will have a piece of owner information, and the owner can be set to none.
- The purpose of the inventory system is to manage the items the player has collected.
- The inventory class should have methods to add and remove items and to view the individual items in the inventory or a collection of items based on different item types
    - Don't forget to change the item ownership when adding or removing items.

        - When viewing an individual item, the output should be a human-readable string describing the item. (hint: think of using .__str__() method)
        - When viewing a collection of items, the output should be a list of human-readable strings describing the items.
        - The viewing collection should have options to view different types of items and the option to view the entire inventory.
    - On top of that, the inventory class should also be considered an iterator to support batch operation on items down the road.
    - The inventory class should also support the membership operator ‘in’ to check if an item is already in the inventory.

Coding Tasks Hints

- You will design, modify, and implement a group of classes based on your work in Lab 04.
- The relationships between the inventory class and all item classes should be considered as a “has-a” relationship.
- For the detailed weapon types, it is up to you to decide how to create the weapon type separation to fit the animation team’s requirements, the method of achieving this will be left to your discretion. However, since this part of the code will be worked on by various different teams and coders, please avoid using purely implicit ways (only duck typing). You can construct ABCs or use typing.Protocol to have explicit indications of the relationship between these classes.

Example:

You can still use the items created from last week’s testing code. Here are some item’s information for testing:

Single-handed weapon:

    (master sword, legendary, damage: 300, type: 'sword')

Double-handed weapon:

    (muramasa, legendary, damage: 580, type: ‘katana’)

Pike:

    (gungnir, legendary, damage: 290, type: ‘spear’)

Ranged weapon:

    (belthronding, legendary, damage: 500, type: ‘bow’)

```python
beleg_backpack = Inventory(owner = 'Beleg')
beleg_backpack.add_item(belthronding)
beleg_backpack.add_item(hp_potion)
beleg_backpack.add_item(master_sword)
beleg_backpack.add_item(broken_pot_lid)
beleg_backpack.add_item(muramasa)
beleg_backpack.add_item(gungnir)
beleg_backpack.add_item(round_shield)
beleg_backpack.view(type = 'shield')
beleg_backpack.view()
beleg_backpack.drop_item(broken_pot_lid)

if master_sword in beleg_backpack:
    master_sword.equip()
    print(master_sword)
    # message to show off your legendary item
    master_sword.use()
    # Beleg slash using master sword
    # master sword is used dealing 345 damage

for item in beleg_backpack:
    if isinstance(item, Weapon):
        beleg_backpack.view(item = item)
```