W8.2 Applied + W8.3 Post-Class

Create a video game ⚔ □ □□ □□

In this activity, you will implement your own game in Python, based on the provided description of the game. Specifically, your task is to develop the characters of a very simple turn-based video game (as a part of W8.2 Applied), and implement the gameplay (as a part of W8.3 Post-Class).

W8.2 Applied

Your first task is to design and implement the Python classes in the file named `characters.py` following the mechanics and the requirements of the game that are described below.

Question 1 (General Character Class: Constructor and Variables)

Your first task is to design the general character class `GeneralClass` that will be used to implement all other character classes we will see later. Please implement `GeneralClass` class with the following two class variables:

- `character_tuple` (i.e., a tuple of strings denoting the character classes, namely: `('Knight', 'Archer', 'Mage', 'Super Mage')`),
- `move_list` (i.e., a list of string(s) denoting the list of moves available for the character class, namely: `['regular_attack']`).

Moreover, `GeneralClass` should have the following five instance variables:

- `name` (i.e., a string denoting the name of the character which should be initialised as `None` by the constructor),
- `health` (i.e., a non-negative integer number which should be initialised as `None` by the constructor),
- `power` (i.e., a non-negative integer number which should be initialised as `None` by the constructor),
- `move_list` (i.e., a list of strings denoting the list of moves the character can perform which should be initialised as a copy of `move_list` class variable by the constructor),
- `death_cry` (i.e., a string denoting death cry of the character which should be initialised as `None` by the constructor).

The constructor of `GeneralClass` class should not take in any inputs.

▶ Expand

Please click on the **'Mark'** button after solving Question 1.

Question 2 (General Character Class: Simple Methods)

The second task is to write methods that will allow us to interact with the values of a character's attributes. Specifically, we will write the following methods in our implementation of `GeneralClass` class:

- Five methods where each method will return the value of a specific instance variable. Each method should use the following naming convention *'get_nameofthevariable'* where *nameofthevariable* is the name of the instance variable.

  ▶ Expand

- One method, named `get_character_tuple`, for returning the list stored in `character_tuple` class variable.
- Four methods where each method will set the value of a specific instance variable, namely for: `name`, `health`, `power` and `death_cry`. Each method should use the following naming convention *'set_nameofthevariable'* where *nameofthevariable* is the name of the instance variable.

  ▶ Expand

▶ Expand

Please click on the **'Mark'** button after solving Question 2.

Let us now review the dynamics of the game in order to finalise the implementation of `GeneralClass` class.

Question 3 (General Character Class: Advanced Methods for Gameplay)

The third task is to implement the advanced methods that will allow a character to i) randomly choose a move from `move_list`, ii) execute a move on the opponent, iii) receive damage from an opponent's move, and shout out death cry `death_cry` if they die. Specifically, we will write the following methods in our implementation of `GeneralClass` class:

- `rand_move` method (with no inputs) that will return a random move from `move_list`.

  ▶ Expand

- `hit` method with three inputs, namely: `scale`, `random_lb`, `random_ub` (in that order), which returns the damage inflicted to the opponent from executing a move based on the following formula:

$\lceil$ `r` $\times$ `power` $\times$ `scale` $\rceil$ where `r` is a random number (generated by `random.uniform` of type `float`) between `random_lb`, `random_ub`, and $\lceil x \rceil$ rounds up number x to the nearest integer number.

- `get_hit` method with one input, namely: `damage` denoting the damage that is received from the opponent, which sets the remaining health `health` of the character based on the formula: $\max($ `health` $-$ `damage` $, 0)$
  If the character dies (i.e., when `health` is equal to 0), `get_hit` method must also invoke the character's death cry `death_cry`.

▶ Expand

Please click on the **'Mark'** button after solving Question 3.

Congratulations - you have successfully implemented the general character class `GeneralClass`!

We are now ready to implement the special characters of the game. The game has the following character classes: i) Knight, ii) Archer, iii) Mage and iv) Super Mage.

Please do not forget to think about inheritance in the design of your classes.

Question 4 (Knight Character Class) ⚔

Knight has average values for `power` and `health`. A knight is capable of performing the following moves:

- `regular_attack` - the damage is calculated as $\lceil$ `r` $\times$ `power` $\rceil$ where `r` is a random number (generated by `random.uniform` of type `float`) between 0.9 and 1.1, and $\lceil x \rceil$ rounds up number x to the nearest integer number.
- `spear_attack` - the damage is calculated as $\lceil$ `r` $\times$ `power` $\times$ 1.1 $\rceil$ where `r` is a random number (generated by `random.uniform` of type `float`) between 0.7 and 1.4, and $\lceil x \rceil$ rounds up number x to the nearest integer number.

which are used to override the `hit` method. The overridden `hit` method will input a move from `move_list` and output the damage computed by `GeneralClass` class' `hit` method using the values specified for `scale`, `power`, `random_lb` and `random_ub`.

A knight is initialised with the following values of its attributes:

- `name` is set to `'Knight'`.
- `health` is randomly set to an integer between 10 and 15.
- `power` is randomly set to an integer between 2 and 4.
- `move_list` is set to `['regular_attack', 'spear_attack']`.
- `death_cry` is set to `'Arrrrgh!'`.

▶ Expand

Please click on the **'Mark'** button after solving Question 4.

Question 5 (Archer Character Class)

Archer has a high value for `power`, but a low value for `health`. An archer is capable of performing the following moves:

- `regular_attack` - the damage is calculated as $\lceil$ `r` $\times$ `power` $\times$ 1.2 $\rceil$ where `r` is a random number (generated by `random.uniform` of type `float`) between 0.9 and 1.1, and $\lceil x \rceil$ rounds up number x to the nearest integer number.
- `bow_attack` - the damage is calculated as $\lceil$ `r` $\times$ `power` $\times$ 1.5 $\rceil$ where `r` is a random number (generated by `random.uniform` of type `float`) between 0.7 and 1.3, and $\lceil x \rceil$ rounds up number x to the nearest integer number.

which are used to override the `hit` method. The overridden `hit` method will input a move from `move_list` and output the damage computed by `GeneralClass` class' `hit` method using the values specified for `scale`, `power`, `random_lb` and `random_ub`.

An archer is initialised with the following values of its attributes:

- `name` is set to `'Archer'`.
- `health` is randomly set to an integer between 7 and 13.
- `power` is randomly set to an integer between 3 and 5.
- `move_list` is set to `['regular_attack', 'bow_attack']`.
- `death_cry` is set to `'Eeeeek!'`.

▶ Expand

Please click on the **'Mark'** button after solving Question 5.

Additional Question 1 (Mage Character Class) - Do this if you want more practise

Mage has a high value for `health`, but a low value for `attack`. A mage is capable of performing the following moves:

- `regular_attack` - the damage is calculated as $\lceil$ `r` $\times$ `power` $\rceil$ where `r` is a random number (generated by `random.uniform` of type `float`) between 0.6 and 1.0, and $\lceil x \rceil$ rounds up number x to the nearest integer number.
- `spell_attack` - the damage is calculated as $\lceil$ `r` $\times$ `power` $\times$ 0.9 $\rceil$ where `r` is a random number (generated by `random.uniform` of type `float`) between 0.8 and 0.9, and $\lceil x \rceil$ rounds

up number x to the nearest integer number.

which are used to override the `hit` method. The overridden `hit` method will input a move from `move_list` and output the damage computed by `GeneralClass` class' `hit` method using the values specified for `scale`, `power`, `random_lb` and `random_ub`.

A mage is initialised with the following values of its attributes:

- `name` is set to `'Mage'`.
- `health` is randomly set to an integer between 19 and 23.
- `power` is randomly set to an integer between 1 and 3.
- `move_list` is set to `['regular_attack', 'spell_attack']`.
- `death_cry` is set to `'Noooo!'`.

▶ Expand

Please click on the **'Mark'** button after solving Question 6.

Additional Question 2 (Super Mage Character Class) □□ - Do this if you want more practise

Super Mage has high values for both `power` and `health`. A super mage is capable of performing the following moves:

- `regular_attack` - the attack damage is calculated as $\lceil$ `r` $\times$ `power` $\rceil$ where `r` is a random number (generated by `random.uniform` of type `float`) between 0.6 and 1.0, and $\lceil x \rceil$ rounds up number x to the nearest integer number.
- `spell_attack` - the attack damage is calculated as $\lceil$ `r` $\times$ `power` $\times$ 0.9 $\rceil$ where `r` is a random (generated by `random.uniform` of type `float`) between 0.8 and 0.9, and $\lceil x \rceil$ rounds up number x to the nearest integer number.
- `super_spell_attack` - the attack damage is calculated as $\lceil$ `r` $\times$ `power` $\rceil$ where `r` is a random number (generated by `random.uniform` of type `float`) between 1.2 and 1.7, and $\lceil x \rceil$ rounds up number x to the nearest integer number.

which are used to override the `hit` method. The overridden `hit` method will input a move from `move_list` and output the damage computed by `GeneralClass` class' `hit` method using the values specified for `scale`, `power`, `random_lb` and `random_ub`.

A super mage is initialised with the following values of its attributes:

- `name` is set to `'Super Mage'`.
- `health` is randomly set to an integer between 22 and 25.
- `power` is randomly set to an integer between 3 and 5.
- `move_list` is set to `['regular_attack', 'spell_attack', 'super_spell_attack']`.

- `death_cry` is set to `'-.-'`.

▶ Expand

Please click on the **'Mark'** button after solving Question 7.

Congratulations - you have successfully implemented all four special character classes!

Given you have successfully implemented all the special character classes, you are actually *really* close to creating your first game!

W8.3 Post-Class

Your second task is to implement the gameplay - so let us focus on the 1-Player version of the game next. Please write your program that allows the user to play the game into the file named `game.py`. Here is what the pseudo code for the game looks like:

Please write your program that allows the user to play the game into the file named `game.py`. Here is what the pseudo code for the game looks like:

```
Prompt the user to select a character, and instantiate the selected character.
Randomly instantiate the opponent's character.
Until one of the characters die, perform the following in a loop.
    Display move_list for the selected character.
    Prompt the user to select a move from move_list and execute that move (i.e., using the methods
    The opponent selects a random move from move_list and executes that move (i.e., using the metho
#Enjoy :)
```

▶ Expand

Feedback

Question 1

Feedback

What worked best in this lesson?

No response

Question 2

Feedback

What needs improvement most?

No response