CS 172 – Lab 4: Inheritance

Monster Battle

In this lab we will create a battle game. You will complete two classes that derive from a `Character` class, `Monster` and `Hero` and use these to perform a battle!

Provided Code

You are provided with two files. Study each file before creating your own classes.

- **characters.py**: This file contains a fully implemented `Character` **abstract base class**, and starter code for two derived classes, `Monster` and `Hero`.
- **main.py**: The main program. This program allows a monster and hero to battle and determines the winner.

You will first work with the **characters.py** file and complete the `Monster` and `Hero` derived classes. **You are not allowed to change the code in** `Character` **class**.

Part 1 – Milestone #1: The `Monster` class

The `Monster` class is a class that inherits from the abstract `Character` class. The starter code states which methods must be implemented and how. But in general:

- `__init__`: This method takes in all the parameters needed to instantiate the base class, as well as one additional parameter, `motivation`. Using this information it instantiates the base class, and it creates a new attribute to store the monster's motivation.
- `__str__`: This method creates and returns a string representation of the `Monster` object. The format of this string is shown in the starter code and can be seen in the Example Execution Trace shown later. Note that `__str__` methods do not print anything. They just return a string.
- `getMotivation(self)`: This method returns the monster's motivation.
- `react(self)`: This method a string in the form: *Monster's name* laughs maniacally.

Once you have implemented the methods above, go to the bottom of the file. You will find some hardcoded tests for the `Monster` class. Run **characters.py** in Thonny and see if these tests pass. If so, this is a good point to submit your code! You will get partial lab credit for completing this part, even if you do not complete the rest of the lab.

Part 2 – Milestone #2: The `Hero` class

The `Hero` class is also a class that inherits from the abstract `Character` class. The starter code states which methods must be implemented and how. But in general:

- `__init__`: This method takes in all the parameters needed to instantiate the base class, as well as one additional parameter, `defenseName`. Using this information it instantiates the base class, creates a new attribute to store the defense name, and creates a `Boolean` attribute that stores the defense status, initializing it to `False`.
- `__str__`: This method creates and returns a string representation of the `Hero` object. The format of this string is shown in the starter code and can be seen in the Example Execution Trace shown later. Note that `__str__` methods do not print anything. They just return a string.
- `getDefenseName(self)`: This method returns the defense name.
- `isDefending(self)`: This method returns the defense status.
- `defend(self)`: This method sets the defense status to `True`.
- `takeDamage(self, amount)`: This method checks the defense status and if it's true, reduces the amount by 50%, then calls the parent's `takeDamage` method with this new damage amount.
- `react(self)`: This method a string in the form: *Hero's name* charges bravely.

Once you have implemented the methods above, go to the bottom of the file. You will find some hardcoded tests for the `Hero` class. They are “commented”. Once you finish part 2, uncomment them. Run **characters.py** in Thonny and see if these tests pass. If so, this is a good point to submit your code! You will get partial lab credit for completing this part, even if you do not complete the rest of the lab.

Part 3 – Milestone #3: The main script - Create the Tournament Play!

You should now be confident your two classes work. You will now begin implementing **main.py.**

Modify the provided file **main.py** to create a tournament between a monster and a hero.

You first must ask the user for the information needed to create a `Monster` and `Hero`. View the example trace below for the suggested format.

Once your characters are created, they battle! An outline of how the battle get run has been provided for you. There are some **###TODO###** sections throughout the document highlighting areas for you to complete yourself.

Once you are done working on the **main.py** file make your final submission for the lab.

Example Execution Trace

Here is an example of a single battle run between a `Dog` monster, and our hero, `Cat`:

```
Enter monster's name: Dog
Enter monster's description: Doggy and Dogy at the same time
Enter a number for monster's health: 10
Enter monster's weapon name: Bark
Enter monster's weapon damage (as a number): 7
Enter monster's motivation: Steal all the treats!
Enter hero's name: Cat
Enter the hero's description: Furry but Furious
Enter a number for the hero's health: 10
Enter hero's weapon name: Scratch
Enter hero's weapon damage (as a number): 5
Enter the hero's defense name: Catnap
Starting Battle Between
Dog: Doggy and Dogy at the same time
Cat: Furry but Furious
Cat goes first.
```

```
Pick one of these (a)ttack, (d)efend, or sto(p): a

Cat charges bravely.

Cat used Scratch on Dog

The attack did 5.0 damage.

Cat at 10

Dog at 5.0

Dog laughs maniacally.

Dog used Bark on Cat

The attack did 7.0 damage.

Dog at 5.0

Cat at 3.0

Pick one of these (a)ttack, (d)efend, or sto(p): d

Our hero is defending with Catnap !

Dog laughs maniacally.

Dog used Bark on Cat

The attack did 3.5 damage.

Dog at 5.0

Cat at -0.5


Battle is over. let's see who has won...

Dog is victorious!
```

Here is a second example of a single battle run between `The Brain` monster, and our hero, `Buttercup`:

```
Enter monster's name: The Brain

Enter monster's description: a highly intelligent mouse

Enter a number for monster's health: 15

Enter monster's weapon name: laser beam

Enter monster's weapon damage (as a number): 3

Enter monster's motivation: To take over the world!

Enter hero's name: Buttercup

Enter the hero's description: The Toughest Fighter
Enter a number for the hero's health: 15
Enter hero's weapon name: Fireball
Enter hero's weapon damage (as a number): 5
Enter the hero's defense name: Green Energy Orb
Starting Battle Between
The Brain: a highly intelligent mouse
Buttercup: The Toughest Fighter
Buttercup goes first.
Pick one of these (a)ttack, (d)efend, or sto(p): a
Buttercup charges bravely.
Buttercup used Fireball on The Brain
The attack did 5.0 damage.
Buttercup at 15
The Brain at 10.0
The Brain laughs maniacally.
The Brain used laser beam on Buttercup
The attack did 3.0 damage.
The Brain at 10.0
Buttercup at 12.0
Pick one of these (a)ttack, (d)efend, or sto(p): d
Our hero is defending with Green Energy Orb !
The Brain laughs maniacally.
The Brain used laser beam on Buttercup
The attack did 1.5 damage.
The Brain at 10.0
Buttercup at 10.5
Pick one of these (a)ttack, (d)efend, or sto(p): a
Buttercup charges bravely.
Buttercup used Fireball on The Brain
The attack did 5.0 damage.
Buttercup at 10.5
The Brain at 5.0
The Brain laughs maniacally.
The Brain used laser beam on Buttercup

The attack did 3.0 damage.

The Brain at 5.0

Buttercup at 7.5

Pick one of these (a)ttack, (d)efend, or sto(p): a

Buttercup charges bravely.

Buttercup used Fireball on The Brain

The attack did 5.0 damage.

Buttercup at 7.5

The Brain at 0.0


Battle is over. let's see who has won...

Buttercup is victorious!
```

Scoring

The score for the assignment is determined as follows:

- 20pts: `Hero` class is implemented correctly, as per lab specifications.
- 20pts: `Monster` class is implemented correctly, as per lab specifications.
- 20pts: `monster_battle(h1, m1)` function in main.py is implemented correctly.
- 10pts: main script in main.py is implemented correctly.
- 10pts: Attribute mangling is used in `Monster` and `Hero` classes. No redundant methods are coded in `Hero` and/or `Monster` classes.
- 10pts: Program is complete and runs without problems.
- 10pts: Compliance and style

Note: Please make sure you put write both lab partners' names and userIDs (in the form abc123) in the header comment of the file you submit for this assignment.