DEATHLY HALLOWS PART 2++

Summary

*

A – User Guide

This simple program was written using Nagini (…since Nagini is a…python).

This is a simulator for the final show-down between Voldemort and Harry Potter. The war starts and the user can input two spells (first one for Harry, second one for Voldemort).

Each spell has a certain "effect" value. When a wizard casts a spell, his energy decreases by its effect value. When a wizard receives a spell, he can either be hurt by it (if his spell effect is less than that of the spell he received) and thus decreases his health or it can be useless for him.

A wizard dies if his health or energy ends.

You will be prompted for input until either Voldemort or HP dies.

B – Classes

- Wizard.PY

A wizard has health points, energy points, and shields (defaults: 100, 500, 3). Besides the setters and getters, a wizard can:

- **castSpell(spell_casted, spell_received)**
    - Subtracts spell's effect value from wizard's energy
    - Decreases health if spell_received's effect is larger than spell_casted's effect
    - Doesn't decrease health if spell_received's effect is smaller.
- **castUselessSpell(spell_casted)**
    - Used when the opponent has a working shield.

    - Only subtracts spell's effect value from wizard's energy
    - Doesn't change wizard's health

You can also check if a wizard:

- **hasShield()**
    - Checks if the wizard still has any remaining shields or if he had used them all.
- **isAlive()**
    - Checks whether wizard is still alive using his health and energy values.

- Spell.PY

Each new spell object created has a name, and "effect" value and a symbol indicating which of the two wizards can use it.

Only has setters and getters.

- OBSERVER.PY

The observer class only notifies main method when a wizard dies. It has two methods:

- **notifyProgram(Wizard1, Wizard2)**
    - checks if Wiz1 or Wiz2 has died and returns true if a death occurred.
- **whoWon(Wizard1, Wizard2)**
    - checks which wizard is still alive and returns their name.

C – Algorithms and Implementation

- Functions in Main: getSpellData()

The function **getSpellData()** returns a dictionary mapping a spell name (string) to a Spell Object. It opens the spells.txt file, reads it, puts spell details in a list and then using that list, we create Spell Objects. Then, we create a dictionary mapping each spell name to the Object it's related to (this is for easy object access using the name).

- MAIN METHOD

- We start by initializing two wizards: Voldemort and Harry, as well as initializing an observer.
- Then, a simple while loop iterates until a death happens (checked using the observer).
- Inside the while loop: We get a line input and then check its validity.

- Validity is checked by checking if the spells exist, and if each opponent is using their own spells not each other's spells.
- If all input is valid, the attack starts.
- An if statement checks if Harry has used a shield and if this use is valid (he had shields to use), thus, Voldemort uses the function "castsUselessSpell"
- Then we check if Voldemort has used a shield and his shield is functional and do the same.
- If no one has used a shield, or if someone has used a shield but didn't have any to use, then both Voldemort and Harry use the function "castsSpell"
- At the end of every iteration, stats of each opponent are printed.
- At the end of the while loop, the winner is printed using the observer's "whoWon" function.

D – Design Patterns: Observer

The use of the behavioral design pattern "Observer" allows some objects to notify other objects about changes in their state. In this program, the wizards notify the main method once any one of them die.

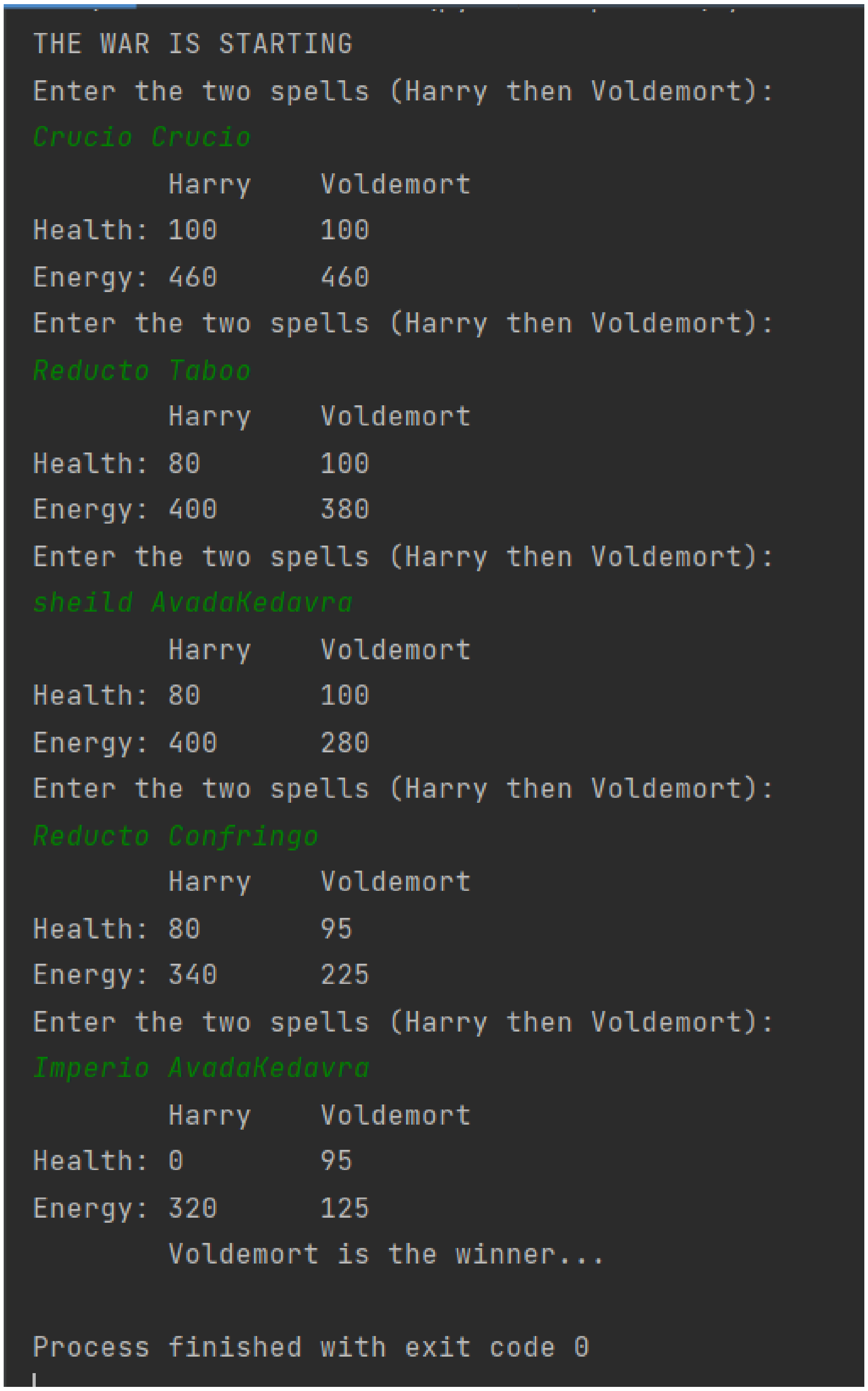
Since, this is a simple program, the use of Observer might seem a bit useless. However, if more details about the wizard were needed other than their death to change a certain state in the main, the Observer pattern will come in handy.

E – Use of OOP Principles: Encapsulation

Encapsulation is apparent in the use of private class attributes and setters and getters for all class attributes.

Inheritance, polymorphism, and abstraction weren't used explicitly in this program.

F – Sample Run



```
THE WAR IS STARTING
Enter the two spells (Harry then Voldemort):
Crucio Crucio
        Harry    Voldemort
Health: 100      100
Energy: 460      460
Enter the two spells (Harry then Voldemort):
Reducto Taboo
        Harry    Voldemort
Health: 80       100
Energy: 400      380
Enter the two spells (Harry then Voldemort):
sheild AvadaKedavra
        Harry    Voldemort
Health: 80       100
Energy: 400      280
Enter the two spells (Harry then Voldemort):
Reducto Confringo
        Harry    Voldemort
Health: 80       95
Energy: 340      225
Enter the two spells (Harry then Voldemort):
Imperio AvadaKedavra
        Harry    Voldemort
Health: 0        95
Energy: 320      125
        Voldemort is the winner...

Process finished with exit code 0
```