Project 7: Pokemon

In this project we will be creating a basic implementation of Pokemon, and then create a game for them to battle. This project will have three steps. The Pokemon Classes, the User classes, and the game loop.

Overview

9 Pokemon

Each Pokemon comes from 1 of 3 types:

1. Grass
2. fire
3. water

Classes

You will be given a base Pokemon class which the grass, fire, and water classes will inherit from.

Class: User

- You will be given a bare-bones implementation of the base user class.

Class: Computer

- The computer class should inherit from the user class
- The computer class will need to overwrite certain methods of the user class.

Attacks

All Pokemon of the same type have the same set of attacks, one of the following:

1. hp (hit points)
2. pp(power points)
3. name.

Game Loop

Before the battle begins, Ask the user:

1. Name
2. Computer Name

Battle Pairing

1. Ask the user to pick 3 Pokemon for their hand
2. Randomly select the computer's Pokemon from the remaining Pokemon.
3. Ask the user which Pokemon they would like to use in battle with currently
4. Randomly select one for the Computer

5. Battle begins.

Each Turn

The user will be given the option to

1. `attack`
2. `heal`
3. `switch`.
4. Computer will be randomly selected, but should more frequently be `attack`.

Attack

Gives the user the options

1. attacks the Pokemon
2. pick one to attack with

Attack Name

An attack has a name, and corresponds to a list of the power of the attack and the accuracy.

An Attack should

1. be a random number in the range between the `attack power(or the pp if that is lower than the attack power) - 20`
2. the `attack power (or the pp if that is lower than the attack power)`
3. randomly fail according to the attack's accuracy.
4. decrease the enemy's hp, and print out a useful description.
5. The computer randomly selects an attack to use in the battle.

Heal

1. Gives 20 HP back to the current Pokemon,
2. Takes away the turn

Switch

1. Lists the stats (HP, AP, Name) of all Pokemon in the hand, and then
2. switches the current Pokemon
3. takes away the turn.
4. On the computer's turn if this option is randomly selected, choose a random Pokemon to switch with
5. If either the user's or the computer's current Pokemon is out of HP, they will be required to switch Pokemon
6. If there are no available Pokemon, that player has lost.

Continue looping until one player has lost

Behaviour

Game Play

- Ask the user to choose 3 Pokemon: At the beginning of the game the program will list out the pokemon to choose from. Options are the following, HP stands for the health points the pokemon has and AP says the max value of attack
- Grass Type:
    - Bulbasoar: 60HP, 40AP
    - Bellsprout: 40HP, 60AP
    - Oddish: 50HP, 50AP
- Fire Type:
    - Charmainder: 25HP, 70AP
    - Ninetails: 30HP, 50AP
    - Ponyta: 40HP, 60AP
- Water Type:
    - Squirtle: 80HP, 20AP
    - Psyduck: 70HP, 40AP
    - Polywag: 50HP, 50AP
- Ask the user to choose Pokemon to use in fight
- Generate Computer's Hand and randomly select a pokemon to start
- On each turn the player or computer can either `attack`, `heal`, or `switch`
- At the end of each the turn print out the HP of the current Pokemon, and check if either side has lost the game
- A player has lost if all their Pokemon are out of HP
- `switch`: Lists the stats (HP, AP, Name) of all Pokemon in the hand, and asks the user what Pokemon they would like to switch to. Cannot attack until next turn.
- `stats`: Lists all the stats of the Pokemon in the hand
- `heal`: adds 20 hp to the current Pokemon, cannot attack until next turn
- `attack`: lists all attacks of current Pokemon, prompts the user pick an attack and then inflicts that damage on the Computer's current pokemon. If this K.Os the enemy then the Computer must switch Pokemon on their next turn. Should print out the following:

```
>>> Ash is attacking. Bulbosaur used Leaf Storm and did 20 damage.
```

Computer

- Computer will select between the three options. 2/3 of the time a user should attack. 1/6 of the time they should heal and 1/6 of the time they should switch pokemon.
- `attack`: Will randomly select an attack and do damage to the User's current Pokemon.
- `heal`: WIll heal current Pokemon by 20 HP, but cannot attack that turn
- `switch`: switched current Pokemon, but cannot attack

Pokemon Attacks

All Pokemon of the same type have the same Attacks.

Grass Type

All grass type have the following attacks:

- Leaf Storm: 130 Power, 90% accurate
- Mega Drain: 50 Power, 100% accurate
- Razor Leaf: 55 Power, 95% accurate Grass Type is 1.5x stronger against Water Type.

Fire Type

All fire type have the following attacks

- Ember: 60 Power, 100% accurate
- Fire Punch: 85 Power, 80% accurate
- Flame Wheel: 70 Power, 90% accurate Fire Type is 1.5x stronger against Grass.

Water Type

All fire type have the following attacks:

- Bubble: 40 Power, 100% accurate
- Hydro Pump: 185 Power, 30% accurate
- Surf: 70 Power, 90% accurate
- Water Type is 1.5x stronger against Fire Types.

Implementation Details

- Use classes to build off the implemented base class for Pokemon. Keep track of HP, Max AP, and type.
  - Here is some starter_code
- Design classes for Fire, Grass, Water Types that inherit from the base Pokemon Class
- Each Pokemon should be an instance, use a master list to store all the Pokemon and create this the beginning of the game
- Use methods to set the Pokemon for each player and remove those pokemon from the master Pokemon list.
- Player's and Computer's Pokemon should be stored using a list
- Pokemon attacks should be stored using a dictionary from the attack name to a list of [attack's power, attack's accuracy]

Check Point 1

Implement the following:

- `get_attack_power` on Pokemon class
- `attack` on Pokemon class
- `take_damage` on Pokemon class
- `heal` on Pokemon class
- Grass, Fire, and Water type classes
- `set_type` on Fire, Water and Grass classes
- `set_attacks` on Fire, Water and Grass classes
- `get_attack_power` on Fire, Water, and Grass classes

Check Point 2

Implement the following:

- `list_pokemon` on User Class
- `switch` on User Class
- `heal` on User Class
- `is_end_game` on User Class
- `print_attacks` on USer Class
- `attack` on User Class

Computer class

- `play_turn` on Computer class
- `set_pokemon` on Computer Class
- `attack` on Computer Class
- `switch` on Computer Class

Check Point 3

Implement the following

- a game loop
- inputs that ask if user would like to attack, heal or switch
- call correct player function based on inputs
- check for the end of the game and end game if necessary

Grading

Scheme/Rubric

Functional Correctness(Behavior)	
Players can choose pokemon hand	15
Computer randomly picks hand	5
Player can choose pokemon to battle	5
Stats, Move, and All stats print properly	10
Attack command decreases HP properly	10
Sub total	45
Technical Correctness	
Correct use of classes	10
Correct use of inheritance	10
Correct use of instances	10
Correct use of variables and game loop	10
Correct use of printing/formatting	10
Sub total	50

Functional Correctness(Behavior)	
Total	95