Task - Pokemon

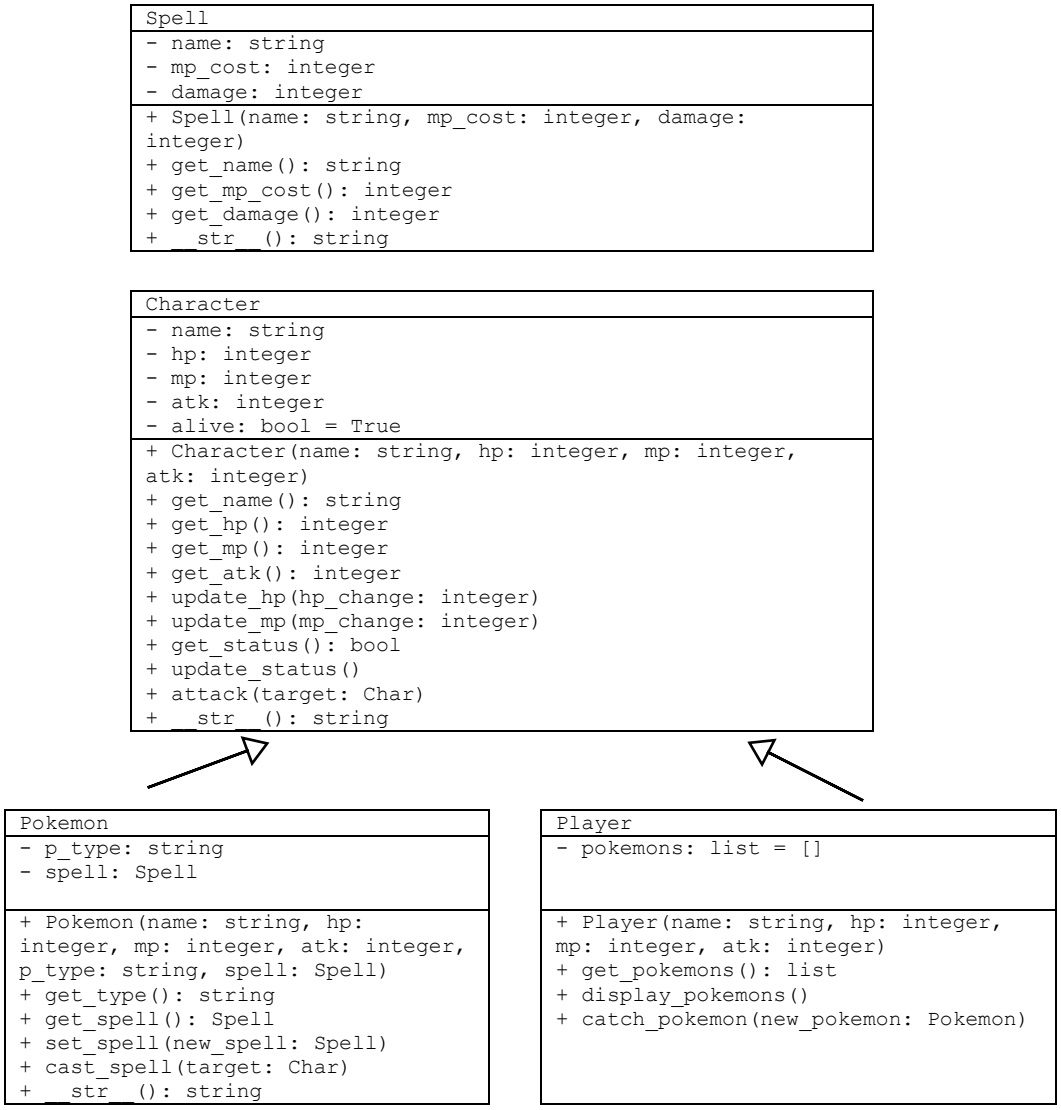
Task 1.1

You are tasked to create a simple text-based game which allows players to catch pokemons. You are tasked to work on an object-oriented solution to store the information of characters involved, which includes the `player` and the `pokemons`. For all `characters`, their `name`, `hp` (health point), `mp` (mana point), `atk` (attack value) are recorded.

There are mainly two kinds of characters involved:

- `Pokemon`: each pokemon belongs to a specific `p_type`, which can take values such as "`Fire`", "`Water`", "`Electric`" or "`Grass`".
- `Player`: player can catch a `list` of pokemons. However, they could only have up to one pokemon of each `type` to be stored in this list. If the player wishes to catch a new pokemon with the same type, he will need to release the old one first.

Below is an UML class diagram for your reference.



Implement the classes based on the following descriptions:

Attributes/Methods	Description
Spell Class	
- name: string - mp_cost: integer - damage: integer	Private attributes of class `Spell`.
+ Spell(name: string, mp_cost: integer, damage: integer) + get_name(): string + get_mp_cost(): integer + get_damage(): integer	Constructor of `Spell` class and its getter methods.
+ __str__(): string	String method for the class, to display the necessary information in the following format: `name: Fireball, mp_cost: 12, damage: 25`
Character Class	
- name: string - hp: integer - mp: integer - atk: integer	Private attributes of class `Character`.
+ Character(name: string, hp: integer, mp: integer, atk: integer) + get_name(): string + get_hp(): integer + get_mp(): integer + get_atk(): integer	Constructor of `Character` class and its getter methods.
+ update_hp(hp_change: integer) + update_mp(mp_change: integer)	Methods to update the `hp` and `mp` values for the character. `hp_change` and `mp_change` can be positive or negative values; they should be added to the current `hp` and `mp` values.
+ get_status(): bool + update_status()	`get_status()` returns the current `alive` status of the character. `update_status()` will check the current `hp` value of the character and set `alive` to `False` if `hp` is less than or equals to `0`.

+ attack(target: Char)	Attack another target:
	- If the target is already dead, print a statement
	- If target alive, attack the target
	- update the target's hp
	- print meaningful output:
	
	`Ash attacks Misty.`
	`Misty is hit, hp changes from 100 to 90.`

+ __str__(): string	String method for the class, to display the necessary information in the following format: `name: Ash Ketchum, hp: 100, mp:100, atk: 20.`

Pokemon Class	

`- p_type: string`
`- spell: Spell`
`+ Pokemon(name: string, hp: integer, mp: integer, atk: integer, p_type: string, spell: Spell)`
`+ get_type(): string`
`+ get_spell(): Spell`
`+ set_spell(new_spell: Spell)`
	Private attribute, constructor, setter and getter method under `Pokemon` class.

`+ cast_spell(target: Char)`	Cast the spell to a target:
	- If the target is already dead, print a statement
	- If target is alive and pokemon has sufficient mana left, cast the spell to the target.
	- update the target's hp, and the caster's mp.
	- print meaningful output:
	
	`Pikachu casts Thunderbolt on Squirtle.`
	`Squirtle is hit, hp changes from 100 to 80.`

`+ __str__(): string`	**Polymorphed** string method to display additional information including the `p_type` and `spell` of the pokemon. E.g.
	
	`Name: Charmander, hp: 50, mp: 60, atk: 18.`
	`type: Fire.`
	`Spell details: name: Fireball, mp_cost: 12, damage: 25.`

Player Class	
`- pokemons: list = []`	Private attribute to store a list of `Pokemon` objects. Should be initialized as an empty list.
`+ Player(name: string, hp: integer, mp: integer, atk: integer)` `+ get_pokemons(): list`	Constructor and getter of `Player` class.

`+ display_pokemons()`	Method to print output to the python shell. If the `player` hasn't caught any pokemons, it should output:
	
	`You have not caught any pokemons yet.`
	
	Otherwise, print out a list of pokemons' information based on the following format:
	
	`Here are the pokemons in Ash Ketchum's team:`
	`Name: Charmander, hp: 50, mp: 60, atk: 18.`
	`type: Fire.`
	`Spell details: name: Fireball, mp_cost: 12, damage: 25.`
	`Name: Squirtle, hp: 55, mp: 60, atk: 15.`
	`type: Water.`
	`Spell details: name: Frostbolt, mp_cost: 8, damage: 16.`

`+ catch_pokemon(new_pokemon: Pokemon)`	Check through the current list of pokemons the player has caught.
	
	If the new pokemon is of the same type of an existing teammate pokemon, prompt the user with the following message:
	
	`Ash Ketchum, you already have:`
	`Name: Charmander, hp: 50, mp: 60, atk: 18.`
	`type: Fire.`
	`Spell details: name: Fireball, mp_cost: 12, damage: 25.`
	`Now you meet:`
	`Name: Ponyta, hp: 60, mp: 60, atk: 18.`
	`type: Fire.`
	`Spell details: name: Fireblast, mp_cost: 15, damage: 30.`
	`Would you like to replace the pokemon of the same type? [Y/N]`
	
	If player choose `Y`, replace the old pokemon with the new one. Then display the following message:
	
	`You have released Charmander, and caught Ponyta.`
	
	Otherwise, display the following message:
	
	`You choose not to catch Ponyta.`
	
	If the new pokemon belongs to a new type, add it to the pokemon list and print the following message:
	
	`You have caught Pikachu.`