# Assignment: Implementing Polymorphism in a Combat Simulation Game

## Objective:

The goal of this assignment is to implement polymorphism in a combat simulation game. You will create an abstract `Enemy` class that defines a common interface for different enemy types. Using polymorphism, you will invoke the `attack()` method for various enemy subclasses without knowing their specific types.  
  
To make the game more engaging, each enemy will have additional properties like health, attack power, and special abilities. You will also implement more interactions such as taking damage and checking if an enemy is defeated.

## Instructions:

### Part 1: Define the Abstract Base Class

1. Create an abstract class `Enemy` with the following attributes and methods:

* - \*\*Attributes\*\*:
* - `\_name` (string): Name of the enemy.
* - `\_health` (int): Health points of the enemy.
* - `\_attack\_power` (int): Damage the enemy can deal.
* - \*\*Methods\*\*:
* - `\_\_init\_\_(self, name: str, health: int, attack\_power: int)`: Constructor to initialize name, health, and attack power.
* - `attack(self)`: Abstract method that will be overridden by subclasses.
* - `take\_damage(self, damage: int)`: Reduces health by the damage amount.
* - `is\_defeated(self) -> bool`: Returns `True` if the enemy’s health is ≤ 0, otherwise `False`.
* - `\_\_str\_\_(self)`: Returns a string representation of the enemy, including health and attack power.

### Part 2: Implement Different Enemy Types

2. Create at least \*\*three\*\* subclasses of `Enemy`, each with a unique attack style and a special ability.

#### Subclass: Goblin

* - \*\*Attack:\*\* Deals normal attack power.
* - \*\*Special Ability:\*\* "Sneaky Strike" (10% chance to deal double damage).
* - \*\*Method:\*\* `special\_attack(self) -> int`: Returns extra damage if the ability activates.

#### Subclass: Dragon

* - \*\*Attack:\*\* Deals attack power + fire damage.
* - \*\*Special Ability:\*\* "Fire Breath" (random chance to burn the player, causing extra damage over time).
* - \*\*Method:\*\* `special\_attack(self) -> int`: Adds fire damage.

#### Subclass: Undead Knight

* - \*\*Attack:\*\* Deals attack power.
* - \*\*Special Ability:\*\* "Revive" (has a 20% chance to resurrect with half health when defeated).
* - \*\*Method:\*\* `revive(self)`: Restores health if the revive condition is met.

### Part 3: Implement a Battle Simulation

1. 3. \*\*Create a function `battle(enemy: Enemy)`\*\* that:

* - Takes an `Enemy` object and simulates an attack.
* - Calls the `attack()` method to deal damage.
* - Calls `take\_damage(damage)` to reduce health.
* - Checks `is\_defeated()` and prints appropriate messages.
* - If the enemy has a special ability, use it.

1. 4. \*\*Demonstrate Polymorphism\*\*:

* - Instantiate different enemy objects (`Goblin`, `Dragon`, `UndeadKnight`).
* - Call `battle(enemy)` for each enemy without needing to know its type.

### Bonus Challenges:

1. 1. \*\*Add a Boss Enemy\*\*

* - Create a `Boss` class that extends `Enemy` and has multiple phases.
* - When its health drops below a certain level, it changes attack patterns.

1. 2. \*\*Implement Player Interactions\*\*

* - Add a `Player` class that can fight back.
* - Allow the player to attack and defend against enemies.

## Expected Outcome Example:

A Goblin appears!   
Goblin attacks with 10 damage.  
Goblin used Sneaky Strike! Critical hit!   
The player takes 20 damage.

A Dragon appears!   
Dragon attacks with 30 fire damage.  
The player takes 30 damage and is burned!

An Undead Knight is defeated... but it resurrects with 50 health!