Desafio: A Batalha Final em Text-RPG

Sua missão é criar um simulador de batalha entre um **Herói** e um **Monstro**. Para isso, você vai construir **uma única classe `Personagem`**, e depois instanciar **dois objetos diferentes**: um para o herói e outro para o monstro. Esses objetos irão lutar até que um deles seja derrotado.

Fase 1: Criação da Classe

Classe `Personagem`

- No `__init__`, cada personagem deve ter os seguintes atributos:
    - `nome` (string)
    - `vida` (número, ex.: 100)
    - `ataque` (número, ex.: 15)

Métodos

1. **atacar(self, alvo)**
    - Recebe outro objeto `Personagem` como alvo.
    - Imprime uma mensagem, por exemplo:
      `"Aragorn ataca o Dragão com sua espada!"`
    - Chama `alvo.receber_dano(self.ataque)`.
2. **receber_dano(self, dano)**
    - Subtrai o dano da vida do personagem.
    - Caso tenha um atributo de defesa, reduza o dano conforme necessário.
3. **esta_vivo(self)**

- Retorna `True` se a vida for maior que zero, `False` caso contrário.

Fase 2: Lógica da Batalha

Agora que a classe está pronta, podemos instanciar os objetos e escrever o loop de batalha.

Passo 1: Instanciando os Objetos

1. `heroi = Personagem("Aragorn", 100, 15)`
2. `monstro = Personagem("Dragão", 80, 10)`

Passo 2: Loop da Batalha

1. Use um `while` para controlar as rodadas.
2. Em cada rodada:
    - Imprima o status de vida: `"Vida do Herói: 80 | Vida do Dragão: 55"`
    - **Turno do Herói:** `heroi.atacar(monstro)`
        - Se o monstro morrer, imprima `"O Herói venceu!"` e termine o loop.
    - **Turno do Monstro:** `monstro.atacar(heroi)`
        - Se o herói morrer, imprima `"O Monstro venceu!"` e termine o loop.
3. Adicione uma pausa de 1 segundo usando `time.sleep(1)` para que a batalha não seja instantânea.

Fase 3: Tornando a Batalha Interativa

No turno do herói, o jogador poderá escolher suas ações:

3. Menu de ações:

```
Escolha sua ação:
(1) Atacar
(2) Fugir
(3) Usar Poção  # se implementado
(4) Defender    # se implementado
```

-
- Leia a escolha usando `input()` e execute a ação correspondente:
    - `"1"` → atacar
    - `"2"` → fugir e terminar a batalha
    - `"3"` → usar poção (verifique se há poções disponíveis)
    - `"4"` → defender (reduz dano do próximo ataque)

Objetivos Bônus

1. **Poção de Cura:**
    - Atributo `pocoes` no herói, começando com 1.
    - Método `usar_pocao(self)` aumenta a vida em 30 pontos e diminui o número de poções.
2. **Ataque Crítico:**
    - Use `random.random() < 0.2` para chance de causar **dobro de dano**.
    - Imprima `"Ataque Crítico!"` quando acontecer.
3. **Defesa:**
    - Se o herói escolher defender, o próximo ataque recebido terá **dano reduzido pela metade**.
    - Use um atributo `self.defendendo = True` e, após receber dano, volte para `False`.