Documentação do Jogo de Batalha Pokémon - Versão Final

---

1. Visão Geral do Jogo

Este jogo de batalha Pokémon é uma aplicação baseada em terminal, inspirada no universo Pokémon, onde os jogadores podem escolher um Pokémon e enfrentá-lo contra outro Pokémon adversário, que é selecionado aleatoriamente. O jogo se roda por turnos, e os jogadores são informados sobre as ações de ataque e o estado dos Pokémons após cada turno.

2. Principais Funcionalidades Desenvolvidas

1. **Seleção do Nome do Treinador**:
    - Antes de iniciar o jogo, o jogador deve digitar o nome do treinador através de uma interface de texto.
2. **Escolha do Pokémon**:
    - O jogador pode escolher seu Pokémon a partir de uma lista com cinco opções: Pikachu, Bulbasaur, Squirtle, Charmander e Eevee.
    - Cada Pokémon tem suas características únicas, como pontos de vida (HP), tipo, velocidade (SPD) e um conjunto de ataques (moves).
        i. Os moves são o que reduz o HP do pokémon inimigo ou aplicam algum demérito ou mérito neles mesmos
3. **Seleção do Adversário**:
    - Um Pokémon adversário é selecionado para a batalha. Assim, pode ser selecionado por um segundo jogador ou não
        i. Em uma versão anterior era utilizado uma versão randomizada do pokémon, escolhida por uma função Random
4. **Sistema de Batalha**:
    - A batalha é executada de forma simples por meio do terminal.
    - Cada Pokémon possui uma velocidade que define quem ataca primeiro no turno.
    - Cada ataque reduz o HP do adversário e o jogo segue até que um dos Pokémon perca todo o HP.

3. Arquitetura do Código

O jogo está organizado em diferentes módulos, cada um com funções distintas. Abaixo, há a descrição dos principais componentes do sistema.

- **`main.py`**: É o ponto de entrada do jogo. Ele lida com a interação do usuário, como seleção do nome do treinador e escolha do Pokémon, e inicia a batalha entre os Pokémon escolhidos.
- **`trainer_selection.py`**: Contém a função `selecionar_nome_treinador()` que permite ao jogador inserir o nome do treinador através do terminal.

- **`pokemon_selection.py`**: Contém a lógica de seleção do Pokémon, onde o jogador escolhe um Pokémon, e um oponente escolhe um pokémon para a batalha.
- **`batalha.py`**: Contém a lógica da batalha. Executa o combate entre os Pokémon e são aplicadas as regras de quem ataca primeiro, os ataques e as atualizações dos HPs.
- **Classes dos Pokémon**:
    - Cada Pokémon no jogo é uma instância de uma classe específica derivada da classe base `Pokemon`. Por exemplo, `Pikachu`, `Bulbasaur`, `Charmander`, etc., herdam suas características da classe `Pokemon`.

4. Programação Orientada a Objetos

Classes e Objetos

As classes são usadas extensivamente para modelar Pokémon e seus movimentos. Cada Pokémon é uma classe específica que herda da classe base Pokémon. Por exemplo:

```
# Arquivo: pikachu.py

from .base_pokemon import Pokemon
from moves.electric_moves import ThunderShock, ElectroBall
from moves.normal_moves import Tackle
from moves.steel_moves import IronTail

class Pikachu(Pokemon):
    def __init__(self):
        super().__init__("Pikachu", "Electric", 135, 55, 40, 50, 50,
90)
        self.moves = [
            ThunderShock(),
            ElectroBall(),
            Tackle(),
            IronTail()
        ]
```

Neste exemplo:

- **Classe `Pikachu`**: `Pikachu` é uma classe que herda da classe base Pokémon.
- **Objeto**: Cada instância de `Pikachu` representa um Pokémon Pikachu com atributos e movimentos específicos.

Herança

Herança é utilizada para modelar os diferentes Pokemon no jogo. A classe base `Pokemon` fornece as propriedades comuns a todos os Pokémon, como HP, tipo, ataques, etc. Cada Pokémon específico (`Pikachu`, `Charmander`, etc.) herda estas propriedades, mas pode ter seus próprios atributos e métodos adicionais.

Classe Base `Pokemon`:

```
# Arquivo: base_pokemon.py

class Pokemon:
    def __init__(self, nome, tipo, hp, atk, defense, sp_atk, sp_def,
spd):
        self.nome = nome
        self.tipo = tipo
        self.hp = hp
        self.atk = atk
        self.defense = defense
        self.sp_atk = sp_atk
        self.sp_def = sp_def
        self.spd = spd
        self.moves = []

    def atacar(self, alvo):
        if self.moves:
            move = random.choice(self.moves)
            print(f"{self.nome} usou {move.nome}!")
            move.executar(self, alvo)
```

Neste exemplo:

- A classe `Pokemon` define os atributos comuns como `hp`, `atk`, `defense`, e os métodos que serão herdados por todas as subclasses.
- `Pikachu`, `Bulbasaur`, etc., são subclasses que herdam essas propriedades e podem ter seus próprios ataques exclusivos.

Polimorfismo

O Polimorfismo é usado quando diferentes Pokémon podem usar o mesmo método `atacar()`, mas a implementação do movimento executado depende da instância do Pokémon. Cada

Pokémon tem seus próprios ataques disponíveis, e a função `atacar()` é chamada da mesma forma para todos os Pokémon, mas o ataque executado é diferente para cada um.

```
# Método atacar polimórfico
def atacar(self, alvo):
    if self.moves:
        move = random.choice(self.moves)
        print(f"{self.nome} usou {move.nome}!")
        move.executar(self, alvo)
```

Neste exemplo:

- **Polimorfismo** ocorre quando o método `atacar()` é utilizado em qualquer Pokémon (`Pikachu`, `Charmander`, etc.), mas cada Pokémon pode executar um movimento diferente, de acordo com a sua lista de ataques (`moves`).

Criado por Bruno Garcez e Yago Solner