

Documentação - RPG de Texto Python (POO)

Visão Geral

Este projeto é um RPG de texto simples, desenvolvido em Python, onde o jogador escolhe entre três classes de heróis (Mago, Guerreiro ou Arqueiro) e enfrenta batalhas automáticas contra um inimigo. O objetivo é demonstrar os principais conceitos da Programação Orientada a Objetos (POO) de forma prática e divertida.

Como Jogar

1. Execute o arquivo `main.py`
 2. Digite o nome do seu personagem
 3. Escolha uma das classes disponíveis:
 - Mago
 - Guerreiro
 - Arqueiro
 4. O sistema cria um inimigo automaticamente
 5. A batalha acontece em turnos, com ataques alternados até que um dos personagens morra
 6. O resultado da batalha é exibido no terminal.
-

Instalação e Execução

```
git clone https://github.com/Eltondrknss/rpg-texto-python.git
cd rpg-texto-python
python main.py
```

Não é necessário instalar nenhuma biblioteca externa.

Estrutura dos Arquivos

```
rpg-texto-python/  
├── main.py           # Arquivo principal, controla o fluxo do jogo  
├── personagem.py     # Classe base Personagem  
├── classes_herois.py # Subclasses: Mago, Guerreiro, Arqueiro  
├── readme.md         # Instruções básicas e conceitos  
└── .gitignore        # Arquivos ignorados pelo Git
```

- `main.py` : Inicia o jogo, recebe entrada do usuário e gerencia o loop da batalha.
- `personagem.py` : Define a classe base `Personagem` com atributos e métodos comuns.
- `classes_herois.py` : Implementa as subclasses `Mago`, `Guerreiro` e `Arqueiro`, cada uma com lógica de ataque personalizada.

Detalhamento das Classes

Personagem

Classe base para todos os personagens do jogo.

Atributos protegidos:

- `_nome` : Nome do personagem
- `_vida` : Vida atual
- `_vida_maxima` : Vida máxima
- `_ataque` : Valor de ataque
- `_defesa` : Valor de defesa

Principais métodos:

- `receber_dano(quantidade_dano)` : Aplica dano ao personagem, considerando defesa.
- `curar(quantidade_cura)` : Recupera vida, sem ultrapassar o máximo.
- `atacar(outro)` : Método abstrato, sobrescrito nas subclasses.
- `status()` : Exibe os atributos do personagem.
- `esta_vivo()` : Retorna `True` se o personagem está vivo.

- `morreu()` : Retorna `True` se a vida chegou a zero.

Heróis (herdados de Personagem):

Mago

- Ataque: Bola de fogo (dano dobrado).
- Mensagens temáticas para ataques e erros.

Guerreiro

- Ataque: Espadada (defesa do inimigo reduzida pela metade).
- Mensagens temáticas para ataques e bloqueios.

Arqueiro

- Ataque: Flechada com chance de crítico (10% de chance de dano dobrado).
- Mensagens temáticas para ataques, críticos e erros.

Fluxo do Jogo

1. Usuário escolhe nome e classe.
2. Instância do jogador é criada usando a classe escolhida.
3. Instância do inimigo é criada como Guerreiro.
4. Loop de batalha:
 - Jogador ataca inimigo.
 - Se inimigo sobreviver, ataca o jogador.
 - Mensagens são exibidas a cada ação.
5. Quando um personagem morre, o resultado é exibido.

Exemplo de Saída

BEM VINDO AO SIMULADOR DE BATALHAS RPG DE TEXTO DO ELTON

Digite o nome do seu jogador: Elton

Escolha sua classe :

- 1 - 🧙 Mago
- 2 - ⚔️ Guerreiro
- 3 - 🏹 Arqueiro

Digite o número da classe escolhida: 1

🧙 Elton lançou uma bola de fogo e causou 60 de dano, deixando Jubileu com 40 de vida

⚔️ Jubileu meteu a espadada no Elton e causou 20 de dano, deixando Elton com 80 de vida

...

FIM DA BATALHA

Elton venceu a luta! Parabéns, Elton.

Pilares de POO Aplicados

- **Encapsulamento:**

Atributos protegidos (`_nome` , `_vida` , `_ataque` , `_defesa`) e acesso controlado via `@property` . Métodos como `receber_dano` e `curar` garantem regras do jogo (vida não negativa, cura limitada).

- **Herança:**

Subclasses `Mago` , `Guerreiro` e `Arqueiro` herdam de `Personagem` , reutilizando atributos e métodos comuns.

- **Polimorfismo:**

O método `atacar` é sobrescrito em cada subclasse, permitindo comportamentos diferentes para cada tipo de personagem.

- **Abstração:**

A classe base define métodos e atributos comuns, enquanto as subclasses implementam detalhes específicos.