

Documentação - RPG de Texto Python (SOLID)

Este projeto é um RPG de texto desenvolvido em Python, com foco em boas práticas de programação orientada a objetos e princípios SOLID. O jogador pode escolher entre diferentes classes de herói e enfrentar batalhas automáticas contra inimigos gerados pelo sistema. O jogo é totalmente executado no terminal, com mensagens temáticas e feedbacks dinâmicos.

Fluxo do Jogo

1. O jogador inicia o programa e digita seu nome.
 2. Escolhe uma classe de personagem.
 3. O sistema gera um inimigo automaticamente.
 4. Os personagens lutam em turnos, alternando ataques.
 5. Cada ataque calcula o dano com base nos atributos de ataque e defesa.
 6. Mensagens temáticas são exibidas a cada ação.
 7. O jogo termina quando um dos personagens morre, exibindo o resultado final.
-

Instalação e Execução

```
git clone https://github.com/Eltondrknss/rpg_texto_python_SOLID.git
cd rpg_texto_python_SOLID
python main.py
```

Não é necessário instalar nenhuma biblioteca externa.

Estrutura do Projeto

```
RPG_SOLID/
|
|— main.py          # Ponto de entrada do jogo
|— models/
|   |— personagem.py  # Classe base Personagem
|   |— guerreiro.py   # Classe Guerreiro
|   |— mago.py        # Classe Mago
|   |— arqueiro.py    # Classe Arqueiro
|   |— ...           # Outras classes e utilitários
|— README.md        # Guia rápido do projeto
|— documentacao.md   # Documentação detalhada
|— .gitignore       # Arquivos ignorados pelo Git
```

- **main.py**: Gerencia o fluxo principal do jogo, entrada do usuário e loop de batalha.
- **models/**: Contém todas as classes de personagens e subclasses.
- **README.md**: Guia rápido para execução e entendimento do projeto.
- **documentacao.md**: Documentação detalhada sobre arquitetura e funcionamento.

Funcionalidades




- Escolha de nome e classe do personagem (Mago, Guerreiro ou Arqueiro).
 - Geração automática de inimigos.
 - Sistema de batalha por turnos, com cálculo de dano e defesa.
 - Mensagens temáticas para cada ação.
 - Encapsulamento dos atributos dos personagens.
 - Fácil expansão para novas classes e mecânicas.
 - Código modular e organizado.
-

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 espada no Elton e causou 20 de dano, deixando Elton com 80 de vida

...

FIM DA BATALHA

Elton venceu a luta! Parabéns, Elton.

Detalhamento das Classes

Personagem

Classe base para todos os personagens do jogo.

Responsável por atributos comuns e métodos utilitários.

Atributos:

- `nome` : Nome do personagem.
- `vida` : Vida atual.
- `vida_maxima` : Vida máxima.
- `ataque` : Valor de ataque.
- `defesa` : Valor de defesa.

Métodos:

- `receber_dano(quantidade_dano)` : Reduz vida do personagem, impedindo valores negativos.
- `curar(quantidade_cura)` : Recupera vida até o máximo permitido.
- `esta_vivo()` : Retorna se o personagem está vivo.
- `atacar(outro)` : Método abstrato, implementado nas subclasses.
- `status()` : Exibe status atual do personagem.
- `morreu()` : Retorna se o personagem morreu.

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.

Princípios SOLID aplicados

- **Single Responsibility:**

Cada classe tem uma responsabilidade única (ex: Personagem gerencia atributos e métodos comuns, subclasses implementam ataques específicos).

- **Open/Closed:**

O sistema permite adicionar novas classes de personagem sem modificar as existentes.

- **Liskov Substitution:**

Subclasses podem ser usadas no lugar da classe base sem alterar o funcionamento.