INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA PIAUÍ	**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO PIAUÍ** **Curso: ADS** **Disciplina: Programação Orientada a Objetos** **Professor: Ely**

Trabalho Final

Crie um jogo de batalha em duplas em TypeScript ou Java com a possibilidade de batalhas, personagens e ações.

1. Objetivo Geral

Criar um sistema de batalha com personagens, herança, regras distintas por classe, registro de ações e uma interface de texto interativa.

2. Requisitos Técnicos

O sistema deve ser desenvolvido em TypeScript ou Java utilizando:

- Classes;
- Herança e polimorfismo;
- Encapsulamento;
- Validações;
- Arrays para coleta de ações;
- Interface de texto para interação com o usuário (via console);
- Persistência de personagens;
- Execução programável em um arquivo principal.

3. Especificação das Classes (2,0)

3.1 Classe Base: Personagem

Todo personagem deve possuir:

- id: number;
- nome: string;
- vida: number entre 0 e 100;
- ataque: number;

- historico: Acao[].

Métodos obrigatórios:

- atacar(alvo: Personagem): Acao;
  - deve funcionar juntamente com o método receber dano, onde o valor do ataque do personagem chama o método alvo.receberDano();
- receberDano(valor: number): void;
  - reduz o valor do atributo vida do personagem. Caso o atributo venha a ficar negativo, deve-se substituir pelo valor0,
  - atualiza o atributo vivo do personagem caso atinja o valor de vida menor ou igual a zero;
- estaVivo: boolean
  - retorna verdadeiro caso o valor do atributo vida for maior que zero;
- registrarAcao(acao: Acao): void;
  - registra no array de ações do personagem um objeto do tipo Acao (definido mais abaixo)

3.2 Subclasses Obrigatórias

Crie três subclasses com comportamentos especiais e herdam da classe personagem.

3.2.1 Guerreiro

- Possui um atributo defesa. Ao ser atacado, o valor do ataque do inimigo é subtraído do valor defesa;
- Causa +30% a mais de dano no ataque quando a vida estiver abaixo de 30%;
- Caso um guerreiro seja atacado por alguém com o valor de ataque menor, o ataque não surtirá efeito.

3.2.2 Mago

- Seu ataque ignora defesa dos guerreiros;
- Causa dano com valor dobrado nos arqueiros;
- Cada ataque consome 10 pontos de vida do próprio mago. Isso também deve ser registrado como uma ação

3.2.3 Arqueiro

- Possui um atributo chamado ataqueMultiplo;
- Ao atacar, possui chance 50% (definida randomicamente) de realizar um ataque com ataqueMultiplo. Ex: ataque = 10, ataqueMultiplo = 3. Faça um “sorteio” e se for um ataque múltiplo, o valor total do ataque = 30.

3.3 Classe Acao

Toda ação deve ter os atributos:

- id: number
- origem: Personagem;
- alvo: Personagem;
- descrição: string; (padrão livre)
- valorDano: number;
- dataHora: Date.

.

3.4 Classe Batalha

A classe deve permitir organizar uma batalha entre vários personagens:

- personagens: Personagem[];
- acoes: Acao[].

Métodos obrigatórios:

- adicionarPersonagem(p: Personagem): void;
  - deve criar e cadastrar um personagem que com nome único;
- turno(atacanteId: number, defensorId: number): Acao[];
  - consulta o atacante e devensor e realiza um ataque
  - também registra a ação do ataque no array de ações da classe;
- consultarPersonagem(nome: string): Personagem
  - retorna um personagem pesquisando-o pelo nome;
- listarPersonagens(): Personagem[];
- listarAcoes(): Acao[];
- verificarVencedor(): Personagem.

Toda ação deve ser registrada no histórico do personagem e no log geral de ações e a batalha deve terminar quando restar apenas um personagem vivo. Caso necessário, adicione mais métodos para ter um melhor controle.

4. Validações Obrigatórias (2,0)

As seguintes regras mínimas devem ser seguidas com o uso de exceções:

- Personagem morto não pode atacar e nem ser atacado;
- Personagem não pode atacar a si mesmo;
- Nomes de personagens não podem se repetir;
- Valores numéricos devem ser válidos;
- Personagem não encontrado também lança exceção.

5. Simulação Interativa (2,0 pontos)

- Implemente uma interface de terminal interativa que permita executar os métodos da Batalha;
- Seja criativo e, sempre que possível, liste opções para o usuário escolher previamente, minimizando a necessidade de memorizar nomes ou códigos e outros identificadores;
- Crie mecanismos para ordenar ações corretamente e mostrar a linha do tempo da batalha;
- Dê o feedback a cada interação, exibindo o que foi solicitado com sucesso ou que houve erro.

6. Persistência de Dados (2,0 pontos)

- Salve dados dos personagens de uma batalha em arquivo ao fechar a aplicação;
- Recupere os dados dos personagens ao abrir a aplicação;
- Sugestão: usar JSON ou CSV.

6. Extras (2,0 pontos)

Adicione funcionalidades extras como:

- Novo personagem: criar um tipo com dinâmicas próprias;
- Modo simulação: a batalha ocorre de forma iterativa, mas com sorteios de quem vai atacar e quem vai defender;
- Estado da batalha: controlar início, andamento e fim. Mesmo que fiquem mais personagens vivos;

- Listar vivos e mortos: consultar personagens por status;
- Estatísticas do personagem: dano causado, recebido, abates;
- Resumo da batalha: gerar texto com vencedor, ações e rodadas;
- Filtrar ações: por personagem, tipo, período ou apenas ataques;
- Replay da batalha: reproduzir ações em ordem no terminal;
- Persistir log completo: salvar personagens + ações + vencedor no arquivo;
- Carregar batalhas passadas: reabrir e visualizar batalhas anteriores;
- Atalhos no menu: listar opções prontas sem digitar nomes e valores manualmente;
- Validação visual: exibir mensagens claras de erro (morto, nome repetido etc.).

7. Entrega

- Código no github;
- Apresentação ao professor por toda a equipe. Apenas membros presentes pontuarão;
- Vídeo no repositório de até 10 min demonstrando o funcionamento do trabalho. Edite o vídeo para torná-lo mais dinâmico. Sem o vídeo a nota será zero.

8. Exemplo de batalha

✅ 📌 **Definição dos Personagens**

■ **Thorgal – Guerreiro**

- Vida inicial: **100**
- Ataque: **25**
- Defesa: **15**
- Passiva:
  - Se a vida cair abaixo de 30%, recebe **+30% de dano**, arredondado para baixo.
  - Ataques com valor menor que seu ataque **não causam dano nele**.

■ **Lyra – Mago**

- Vida inicial: **100**

- Ataque: **30**
- Passiva:
  - Ataques **ignoram defesa de guerreiros**.
  - Causa dano **dobrado em arqueiros**.
  - Cada ataque feito custa **10 de vida** ao mago.

■ Elandor – Arqueiro

- Vida inicial: **100**
- Ataque: **15**
- Ataque múltiplo: **3**
- Passiva:
  - 50% de chance de ataque múltiplo (ataque × 3)

⚔️ 📜 **Extrato da Batalha (com status após cada ação)**

💙 status = vivo

❌ status = morto

Ação 1

Lyra lança magia em Thorgal, ignorando sua defesa.

➡️ Dano: **30**

👤 *Situação após a ação:*

- Thorgal: **70 vida** 💙
- Lyra: **100 vida** 💙
- Elandor: **100 vida** 💙

Ação 2

Lyra sofre o custo da magia.

➡️ Autodano: **10**

👤 *Situação após a ação:*

- Thorgal: **70 vida** 💙
- Lyra: **90 vida** 💙
- Elandor: **100 vida** 💙

Ação 3

Thorgal ataca Elandor com sua espada.

➡️ Dano: **25**

👤 *Situação após a ação:*

- Thorgal: **70 vida** 💙
- Lyra: **90 vida** 💙
- Elandor: **75 vida** 💙

Ação 4

Elandor dispara flecha em Lyra (ataque simples).

➡️ Dano: **15**

👤 *Situação após a ação:*

- Thorgal: **70 vida** 💙
- Lyra: **75 vida** 💙
- Elandor: **75 vida** 💙

Ação 5

Lyra lança outra magia em Thorgal.

➡️ Dano: **30**

👤 *Situação após a ação:*

- Thorgal: **40 vida** 💙
- Lyra: **75 vida** 💙
- Elandor: **75 vida** 💙

Ação 6

Lyra sofre o custo da magia novamente.

➡️ Autodano: **10**

👤 *Situação após a ação:*

- Thorgal: **40 vida** 💙
- Lyra: **65 vida** 💙
- Elandor: **75 vida** 💙

Ação 7

Thorgal atinge Elandor mais uma vez.

➡️ Dano: **25**

👤 *Situação após a ação:*

- Thorgal: **40 vida** 💙
- Lyra: **65 vida** 💙
- Elandor: **50 vida** 💙

Ação 8

Elandor dispara outra flecha em Lyra (ataque simples).

➡️ Dano: **15**

👤 *Situação após a ação:*

- Thorgal: **40 vida** 💙
- Lyra: **50 vida** 💙
- Elandor: **50 vida** 💙

Ação 9

Lyra lança magia poderosa em Thorgal.

➡️ Dano: **30**

⚠️ **Thorgal agora está abaixo de 30 de vida → bônus de +30% ativado**

👤 *Situação após a ação:*

- Thorgal: **10 vida** 💙 **(modo fúria ativo)**
- Lyra: **50 vida** 💙
- Elandor: **50 vida** 💙

Ação 10

Lyra sofre dano de conjuração novamente.

➡️ Autodano: **10**

👤 *Situação após a ação:*

- Thorgal: **10 vida** 💙
- Lyra: **40 vida** 💙
- Elandor: **50 vida** 💙

Ação 11 – Thorgal com bônus ativo

Thorgal desfere ataque poderoso em Elandor.

➡️ Cálculo do dano:

25 × 1.3 = 32.5 → arredondado para **32**

👤 *Situação após a ação:*

- Thorgal: **10 vida** 💙
- Lyra: **40 vida** 💙
- Elandor: **18 vida** 💙

Ação 12

Elandor ativa ataque múltiplo contra Lyra.

➡️ 15 × 3 = **45**

👤 *Situação após a ação:*

- Thorgal: **10 vida** 💙
- Lyra: **0 vida** ❌ **morta**
- Elandor: **18 vida** 💙

Ação 13 – Thorgal ainda no modo bônus

Thorgal parte para cima de Elandor para finalizar.

➡️ 25 × 1.3 = 32.5 → arredondado para **32**

👤 *Situação após a ação:*

- Thorgal: **10 vida** 💙
- Lyra: **0 vida** ❌ **morta**
- Elandor: **0 vida** ❌ **morto**

⏳ **Resultado Final**

✓ **Vencedor: Thorgal – Guerreiro**, sobrevivendo com **10 de vida**