ATIVIDADE 02 - ENTREGA ATÉ 09/09/2024

Sistema de Gestão de Funcionários

Crie um sistema para gerenciar diferentes tipos de funcionários em uma empresa. A empresa possui funcionários em tempo integral, meio período e estagiários, e cada um desses tipos possui características específicas. Utilize **herança** para organizar essas classes.

Requisitos:

1. **Classe Base: Funcionário**
   - Atributos: nome, CPF, salário base.
   - Métodos: getters e setters para todos os atributos, um método `calcularSalario()` que retorne o salário base.
2. **Subclasses:**
   - **Funcionário em Tempo Integral:**
     - Atributos: bônus (percentual sobre o salário base).
     - Método específico: `calcularSalario()`, que retorna o salário base acrescido do bônus.
   - **Funcionário Meio Período:**
     - Atributos: horas trabalhadas, valor por hora.
     - Método específico: `calcularSalario()`, que retorna o valor calculado com base nas horas trabalhadas e no valor por hora.
   - **Estagiário:**
     - Atributos: instituição de ensino, bolsa auxílio.
     - Método específico: `calcularSalario()`, que retorna o valor da bolsa auxílio.

Atividade:

Implementar a classe base `Funcionario` e as subclasses `FuncionarioTempoIntegral`, `FuncionarioMeioPeriodo` e `Estagiario`.

Criar um programa principal que permita cadastrar e exibir as informações de diferentes funcionários. Use a classe Scanner.

O programa deve criar uma lista com vários funcionários e iterar sobre essa lista, chamando o método `calcularSalario()` de cada objeto para exibir o salário correspondente.

Executar os casos de teste fornecidos até que todos sejam aprovados.

Sistema de Gestão de Jogos RPG

Desenvolva um sistema para gerenciar personagens em um jogo de RPG (Role-Playing Game). O sistema deve permitir criar diferentes tipos de personagens (guerreiro, mago, arqueiro), cada um com suas características e habilidades específicas. Utilize **herança** para organizar essas classes e adicione interatividade para que os personagens possam lutar entre si.

Requisitos:

1. **Classe Base: Personagem**
    - Atributos: nome, nível, pontos de vida (HP), pontos de ataque (ATK), pontos de defesa (DEF).
    - Métodos:
        - `atacar(Personagem inimigo)`: Método para atacar outro personagem. O dano é calculado com base no ataque do personagem atacante e na defesa do personagem atacado.
        - `receberDano(int dano)`: Reduz os pontos de vida do personagem conforme o dano recebido.
        - `exibirStatus()`: Exibe os atributos do personagem, como nome, nível, HP, ATK, e DEF.
2. **Subclasses:**
    - **Guerreiro:**
        - Atributos adicionais: força extra, armadura.
        - Métodos específicos: `atacar(Personagem inimigo)` que considera a força extra.
    - **Mago:**
        - Atributos adicionais: mana, poder mágico.
        - Métodos específicos: `lancarMagia(Personagem inimigo)`, que consome mana para causar dano mágico.
    - **Arqueiro:**
        - Atributos adicionais: precisão, alcance.
        - Métodos específicos: `atirarFlecha(Personagem inimigo)`, que causa dano à distância.
3. **Interatividade:**
    - Permitir que os personagens lutem entre si, simulando batalhas onde cada um pode atacar ou usar habilidades especiais.
    - Implementar um sistema de turnos para que os personagens se alternem nos ataques até que um deles perca todos os pontos de vida.

Atividade:

Implementar a classe base `Personagem` e as subclasses `Guerreiro`, `Mago`, e `Arqueiro`.

Criar um programa principal onde seja possível criar instâncias de cada tipo de personagem.

Configurar uma batalha entre eles e exibir o resultado final da batalha.