	INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO PIAUÍ **Curso: ADS** **Disciplina: Tópicos Especiais em** **Programação Professor: Ely**

1) Implemente o que se pede:

a) Crie uma interface chamada Iteravel<T>, ou seja, usando generics com os seguintes métodos com as respectivas semânticas:

i) proximo(): retorna o próximo elemento da coleção do tipo <T>;

ii) atual(): retorna o elemento atual do tipo <T>;

iii) isPrimeiro() e isUltimo(): retornam true se o Iteravel está no primeiro ou úlltimo elemento . Retornam false caso contrário;

iv) irParaPrimeiro(): retorna o Iteravel para primeira posição

b) Crie uma classe chamada ColecaoArray<T> que um array de <T> implemente a interface Iterável e tenha um método um método adiciona(<T>) que adiciona um novo elemento na coleção;

c) Instancie uma ColecaoArray para o tipo number e teste todos os métodos.

2) Implemente usando os recursos da linguagem type script o seguinte problema:

Um protótipo de um jogo possui alguns tipos de personagens. Um **Personagem** é uma entidade que representa o elemento mais genérico do jogo. Possui os atributos: **id** (um identificador inteiro), **nome** e um atributo **energia**, que varia de 0 a 100.00. Possui um método chamado **estaVivo** que retorna falso se a energia do personagem for igual a 0 e verdadeiro caso contrário. Também tem um método chamado **defenderAtaque**, no qual um valor é passado como parâmetro e tal valor é subtraído da sua energia. Esse método não deve deixar a energia ficar negativa. Caso o valor do ataque seja maior que a energia, ela deve ser zerada.

Já um **Soldado** tem as mesmas características e comportamentos de um personagem genérico e possui ainda uma **força de ataque**, que varia de 0 a 10. Além disso, um soldado possui um método chamado **atacar**, no qual:

- Um personagem “p” é passado como parâmetro;
- Dentro do método, o ataque é representado como a chamada do método defenderAtaque do “p” passado como parâmetro. O parâmetro do método defenderAtaque de “p” é a força de ataque do soldado;

Os soldados possuem uma armadura e assim, no seu método defenderAtaque, é

subtraída de sua energia apenas metade do valor passado como parâmetro. Ou seja: um ataque a um soldado reduz apenas metade do valor que um ataque a um personagem normal.

Por fim, existem os cavaleiros. Um **Cavaleiro** é um personagem que possui os mesmos comportamentos e características de um soldado, porém, seus métodos possuem comportamentos um pouco diferentes. O método atacar do cavaleiro tira do personagem passado como parâmetro duas vezes o valor de sua força. Ou seja: se o valor de ataque do cavaleiro for 5.00, o atacado perderá 10.00 pontos de energia ao se defender. Além disso, o método defenderAtaque de um cavaleiro possui um fator de resistência a mais, perdendo apenas 1/3 do valor passado como parâmetro.

As classes acima identificadas devem ficar em um módulo chamado **personagens.ts.** Já as seguintes classes devem pertencer ao módulo **jogo.ts**.

Além das classes que achar necessárias, crie também uma classe **Jogo** que possua um array desses elementos/personagens. Crie os métodos abaixo e observe que eles devem ser os mais genéricos possíveis e evitar duplicação de código. Além disso:

- O método **incluir**:
    - Não deve deixar um personagem com o mesmo id ser inserido;
- O método **consultar**: deve retornar o personagem caso exista um com o mesmo id no vetor e nulo caso contrário;
- O método **atacar** deve:
    - Receber dois ids (identificadores), pesquisar os personagens e chamar o método atacar do personagem do primeiro id passado como parâmetro;
    - Deve-se verificar se o personagem que está atacando é uma instancia de um soldado ou cavaleiro e emitir uma mensagem de erro caso contrário, não realizando assim o ataque;
    - Deve-se evitar também que o próprio personagem se ataque.
- O método **avaliar batalha** deve:
    - Imprimir todos os personagens do jogo mostrando: id – nome – energia – “está vivo?” (true ou false). Se preferir, implemente isso sobrescrevendo o método toString da classe mais genérica dos personagens.

Por fim, crie um script para testar o jogo importando os dois módulos. Instancie 4 personagens (1 personagem, 2 soldados e 1 cavaleiro), adicione-os à classe jogo e faça com que cada um ataque o outro. Após cada ataque, chame o método avaliar batalha e veja se sua implementação foi consistente com o especificado.