Exercício Aula 21

Prof: Henrique Augusto Maltauro

Desenvolvendo Algoritmos

Exercício

Vamos imaginar um videogame.

Criar três classes: Personagem, NPC e Jogador.

Em todas as classes, o método construtor deve receber **ALGUNS** dos atributos como parâmetros.

Exercício

- Classe Personagem

Deve possuir os seguintes atributos:

- ID: protected long
- Nome: protected string
- PontosDeVida: protected int
- Ataque: protected int
- Defesa: protected int
- DirecaoAtual: protected string = “norte”

Exercício

- Classe Personagem

Deve possuir os seguintes métodos (Parte 1):

- Personagem: public (Método construtor)
- GetID: public long
- GetNome: public string
- GetPontosDeVida: public int

Exercício

- Classe Personagem

Deve possuir os seguintes métodos (Parte 2):

- Atacar: public virtual int
- Defender(int ataqueInimigo): public virtual bool
- ReceberDano(int dano): protected void
- CurarPontosDeVida(int pontosDeCura): public void
- Andar(int passos): public void
- Andar(int passos, string direcao): public void

Exercício

- Classe Personagem

Se a vida do personagem for menor ou igual a zero, cada método deve bloquear as ações do personagem.

O método Atacar deve retornar os pontos de ataque do personagem.

O método Defender irá receber o ataque do inimigo, e deverá exibir no console se a defesa foi bem sucedida ou não, verificando se a defesa é maior que o ataque, e deverá retornar um booleano dizendo se a defesa foi bem sucedida.

Exercício

- Classe Personagem

O método ReceberDano deve ser executado dentro do método Defender, **somente quando** a defesa não for bem sucedida, e deve diminuir dos PontosDeVida a diferença dos valores de ataque e defesa. Caso a vida chegue a zero, deverá exibir na tela que o personagem morreu.

O método CurarPontosDeVida irá receber os pontos de cura que deverão ser restaurados na vida do personagem.

Exercício

- Classe Personagem

O método Andar deverá exibir no console a direção que o personagem andou a quantidade de passos recebida.

Ex: “O personagem Harry Potter andou 3 passos para o sul.”

No caso do método Andar receber uma direção, o método deverá exibir no console que o personagem mudou de direção e andou a quantidade de passos recebida.

Ex: “O personagem Harry Potter virou para o leste e andou 3 passos”

Exercício

- Classe NPC

Deve ter uma herança da classe Personagem.

Porém não terá atributos extras.

Exercício

- Classe NPC

Deve possuir os seguintes métodos:

- NPC: public (Método construtor)
- Atacar: public override int
- Defender(int ataqueInimigo): public override bool

Exercício

- Classe NPC

Se a vida do npc for menor ou igual a zero, cada método deve bloquear as ações do npc.

O método Atacar deve retornar os pontos de ataque do npc, porém, deverá utilizar um modificador de ataque aleatório entre 1 e 100.

Exercício

- Classe NPC

O método Defender irá receber o ataque do inimigo, e deverá exibir no console se a defesa foi bem sucedida ou não, verificando se a defesa é maior que o ataque, porém, deverá utilizar um modificador de defesa aleatório entre 1 e 100, e deverá retornar um booleano dizendo se a defesa foi bem sucedida.

Exercício

- Classe NPC

O método ReceberDano deve ser executado dentro do método Defender, **somente quando** a defesa não for bem sucedida, e deve diminuir dos PontosDeVida a diferença dos valores de ataque e defesa. Caso a vida chegue a zero, deverá exibir na tela que o personagem morreu.

Exercício

- Classe Jogador

Deve ter uma herança da classe Personagem.

Além disso, deve possuir os seguintes atributos:

- ExperienciaAtual: private int
- Nivel: private int

Exercício

- Classe Jogador

Deve possuir os seguintes métodos:

- Jogador: public (Método construtor)
- GetNivel: public int
- GanharExperiencia: public void
- Atacar: public override int
- Defender(int ataqueInimigo): public override bool

Exercício

- Classe Jogador

O método construtor não deve receber o nível como parâmetro, ele deve definir como 1 de forma fixa.

Se a vida do jogador for menor ou igual a zero, cada método deve bloquear as ações do jogador.

O método Atacar deve retornar os pontos de ataque do jogador, porém, deverá utilizar um modificador de ataque aleatório entre 25 e 100.

Exercício

- Classe Jogador

O método Defender irá receber o ataque do inimigo, e deverá exibir no console se a defesa foi bem sucedida ou não, verificando se a defesa é maior que o ataque, porém, deverá utilizar um modificador de defesa aleatório entre 25 e 100, e deverá retornar um booleano dizendo se a defesa foi bem sucedida.

Exercício

- Classe Jogador

O método ReceberDano deve ser executado dentro do método Defender, **somente quando** a defesa não for bem sucedida, e deve diminuir dos PontosDeVida a diferença dos valores de ataque e defesa. Caso a vida chegue a zero, deverá exibir na tela que o personagem morreu.

Exercício

- Classe Jogador

O método GanharExperiencia deve ser executado em dois momentos, dentro do método Defender, **somente quando** a defesa for bem sucedida, e quando o ataque for bem sucedido, ou seja, quando o método Defender do inimigo retornar false. Para cada execução, deve aumentar a experiência em 20. Sempre que a experiência chegar em 100, deve aumentar um nível do jogador, zerar a experiência e mostrar no console que o jogador subiu de nível.

Exercício

- Modificador de ataque (1 a 100)

```
// Modificador 1 a 100
Random random = new Random();
double modificador = Convert.ToDouble(random.Next(1, 100)) / 100;
int ataque = Convert.ToInt32(Math.Truncate(Ataque * modificador));
int defesa = Convert.ToInt32(Math.Truncate(Defesa * modificador));
```

Exercício

- Modificador de ataque (25 a 100)

```
// Modificador 25 a 100
Random random = new Random();
double modificador = Convert.ToDouble(random.Next(25, 100)) / 100;
int ataque = Convert.ToInt32(Math.Truncate(Ataque * modificador));
int defesa = Convert.ToInt32(Math.Truncate(Defesa * modificador));
```