Trabalho de Estrutura de Dados – Wumpus

Informações

O objetivo do trabalho é instigar e treinar os alunos sobre lógica e implementação de algoritmos que trabalham com estrutura de dados com grafos.

Regras

- **Deve ser feito em trios.**
- Valor: 40 pontos
- Data de entrega: combinado na aula com a turma.
  - A cada dia de atraso na entrega, a nota será reduzida em 5 (cinco) pontos.
- A entrega deve ser feita via Github. Cada equipe deve ter seu próprio repositório.
- A pontuação será feita da seguinte forma:
  - Funcionamento do programa.
  - Um **Readmd.md** explicando as regras implementadas e o uso da estrutura de dados (não é necessário descrever todos os métodos do código).
- Cuidado com o plágio!
- Observação: é obrigatório o uso da classe **`Caverna`**, abaixo (cada caverna(nó) possui uma referência para outras cavernas (mínimo 1 e máximo 4).

Descrição do problema – Hunt the Wumpus

1. O programa deve ser um jogo de Hunt the Wumpus. Existe um labirinto de cavernas conectadas entre si. Cada caverna pode conter um dos seguintes elementos:
   a) O player (apenas 1) → jogador humano (inicia em uma caverna livre aleatória).
   b) O monstro Wumpus (apenas 1) → se o jogador entrar na caverna do monstro, o jogador perde o jogo.
   c) Poço sem fundo (1 ou mais) → provoca dano (parcial ou total) no jogador, quando ele entra nesta caverna.
   d) Morcego (1 ou mais) → transporta o jogador para outra caverna aleatória.
   e) Flecha (de 1 a 3 cavernas com apenas 1 flecha cada caverna). Quando o player acha uma flecha, incrementa seu número. Quando o player lança ela (decrementa) ela só chega até a próxima caverna do lado escolhido.
2. Cada elemento deve ocupar apenas uma caverna do labirinto. O restante das cavernas ficam livres.
   a) Nas cavernas ao redor do Wumpus, o player consegue sentir o odor.
   b) Nas cavernas ao redor de um Morcego, o player consegue ouvir o bater das asas.
   c) Nas cavernas ao redor do Poço, o player consegue sentir a brisa.
3. Existe apenas 1 jogador humano. O objetivo é descobrir a caverna onde está o mostro e lançar a flecha nele.
   a) Veja os links no final do arquivo. Pesquise mais sobre o jogo.
4. O mapa é composto de N cavernas (mínimo 20 e máximo 30).
   a) Cada caverna pode ter de 1 a 4 conexões com outras cavernas, sendo uma para cada lado: LESTE, OESTE, NORTE e SUL.
   b) A estrutura da caverna possui (no mínimo) seguinte especificação.

```
public class Caverna {
```

```
        private Caverna leste; //null caso não houver conexão deste lado
        private Caverna oeste;
        private Caverna norte;
        private Caverna sul;
        private Inimigo inimigo; //null caso estiver livre
}
```

c) Usar um enum para mapear as direções. Exemplo:

```
public enum Direcao { LESTE, OESTE, NORTE, SUL; }
```

d) É permitido usar qualquer estrutura de dados do Java. Ex: List, ArrayList, Map, Hashmap, etc.

5. Todo o código que imprime algo na tela deve estar separado numa classe com todos os métodos estáticos de **Output**. Idem para a classe de **Input** (modo console).
6. Criar uma classe separada (**Input**) para ler do teclado das opções disponíveis no jogo:
   a) Ler as opções que o humano escolhe jogar.
7. O jogo deve possuir as seguintes informações (no mímimo):
   a) Uma coleção com todas as cavernas.
   b) Uma coleção com as cavernas já visitadas pelo player.
   c) A caverna onde o player está em um dado momento.
8. Ao iniciar, o programa monta o labirinto de cavernas com os inimigos em posições aleatórias. As cavernas devem ser geradas e conectadas aleatoriamente (usar uma *seed* fixa para poder reproduzir o ambiente, caso necessário).
9. A cada jogada deve-se imprimir o resultado da ação (as informações da caverna atual e as opções **válidas** de jogadas para o player (caso não o jogo não for encerrado).
10. Quando chegar ao final do jogo, o programa deve imprimir o motivo do fim:
    a) O player venceu → capturou o Wumpus
    b) O player perdeu → o Wumpus capturou o player.
    c) O player perdeu → acabou a vida do player
    d) O player perdeu → acabou as flechas do player e não há mais flexas no mapa.
11. Criar classes separadas para elemento: Caverna, Direcao, Inimigo (monstro, morcego e poço)(usar herança), Output, Menu e lógica do jogo.
12. Fazer as validações necessárias para o jogo não entrar em um estado inconsistente.
13. Anotar no arquivo README.md as decisões não especificadas nesta descrição (**não** é necessário explicar o código).

Extra: aumentar a dificuldade do jogo:

- Fazer o monstro mover uma caverna adjacente em direção ao player a cada rodada.
- Fazer algum morcego mover a cada rodada.
- Aumentar o número de morcegos ou poços.

Veja mais informações sobre o jogo nos links:
https://osric.com/wumpus
https://jayisgames.com/games/hunt-the-wumpus
http://markhuckvale.com/games/wumpus
https://archive.org/details/hunt-the-wumpus-1980
Existem algumas variações do jogo. Procure por outros links.