Projecto Guiado – 2ª Iteração
Estruturação em Classes

2.1 [Separação em camadas] Restruturar o programa por forma à interacção com o utilizador e a lógica do jogo serem tratadas em classes separadas. Isto é importante para facilitar posteriormente a existência em paralelo de múltiplas formas de usar/exercitar a lógica do jogo: através da interface alfanumérica (já desenvolvida na aula anterior), através de uma interface gráfica (a criar numa próxima aula), através de testes unitários automáticos (idem). Organizar o programa em *packages* distintos, como por exemplo: maze.logic e maze.cli (*command line interface*). No arranque do programa, deve ser possível começar o jogo com o exemplo pré-definido (da aula anterior) ou com um labirinto gerado automaticamente.

2.2 [Estruturação em classes] Reorganizar o *package* com a lógica do jogo (e outros *packages* se necessário) com pelo menos as seguintes classes (com todos os campos privados ou protegidos):

- **a)** Uma classe para representar o estado do jogo, agregando o labirinto em si e os elementos presentes no labirinto, disponibilizando as operações de comando do jogo pelo utilizador e consulta do estado do jogo;
- **b)** Uma classe para representar o labirinto em si (terreno);
- **c)** Classes para representar os vários tipos de elementos que podem estar presentes no labirinto (espada, dragão, herói), com o respectivo estado, e uma super classe com as propriedades comuns (posição, etc.), tirando o mais possível partido de herança e polimorfismo (devendo o comportamento dos vários elementos ser distribuído o mais possível pelas classes respetivas);
- **d)** Uma classe para o gerador de labirintos (ver o padrão BUILDER).[1]

Sugestão: fazer um rascunho de diagrama de classes e discutir com o docente eventuais alternativas.

2.3 Criar uma nova estratégia em que o dragão pode adormecer por algum tempo de forma aleatória, tendo um visual diferente quando está a dormir. Quando o dragão está a dormir, o herói pode matar o dragão se estiver armado, mas não pode ser morto pelo dragão. No interface alfanumérico, representar por letra minúscula quando está a dormir. No arranque do programa ou no início do jogo, o utilizador deve poder escolher a estratégia pretendida (dragão parado, dragão com movimentação aleatória, dragão com movimentação aleatória intercalada com dormir).

2.4 Criar a possibilidade de existência de mais do que um dragão.

2.5 [Para casa] Acrescente um novo tipo de arma: dardos. Podem existir diversos dardos espalhados pelo terreno do jogo, os quais são apanhados pelo herói quando passa nas quadrículas respetivas. O herói pode lançar dardos em qualquer das 4 direções. Se existir um dragão na linha de mira do herói no sentido do lançamento, o dragão é morto. Criar também a possibilidade de o dragão cuspir fogo matando o herói, quando este está a uma distância máxima de 3 quadrículas (este pode ser um comportamento compulsivo do dragão sempre que está acordado e tem o herói na linha de mira). Se o herói tiver um escudo protetor, não é afetado. O escudo protetor é recolhido pelo herói quando o encontra.

[1] Também de pode criar um interface MazeBuilder com duas classes que o implementam: uma para o gerador aleatório e outra para o labirinto por defeito.