Projeto Guiado - 1ª Iteração

Introdução

As aulas práticas da primeira metade do semestre (5 aulas) serão dedicadas à realização iterativa de um "projeto guiado", em grupos de 2 alunos da mesma turma (escolha livre), com enunciado idêntico para todos os grupos. Em cada semana é apresentado o enunciado de uma iteração adicional do projeto. Este projeto deve ser concluído até 4/4 (23h55), com a entrega do código fonte documentado, tendo um peso de 30% na classificação final (sendo a avaliação formada com base no acompanhamento semanal e na avaliação das entregas).

Tema: Jogo do labirinto

O nosso **herói** acaba de acordar ao som de um longo rugido. Não se lembra ao certo do que aconteceu, mas encontrando-se desarmado, o seu primeiro instinto é procurar um meio de se defender... e sair de onde quer que esteja. Sem saber, o nosso herói encontra-se perdido numa labiríntica masmorra, na companhia de um **dragão** de ideias fixas, determinado a fazer dele uma deliciosa refeição. Algures no **labirinto**, existe uma **saída** e uma **espada**. A saída só se abre quando o dragão morre. Sempre que os dois se encontram em quadrados adjacentes o dragão ataca primeiro, e caso o nosso herói não possua a espada, morre.

Objetivos para a 1ª iteração

1.1 Representar em memória e imprimir o seguinte labirinto (só com caracteres, sem cores).

X	X	X	X	X	X	X	X	X	X
X	H								X
X		X	X		X		X		X
X	D	X	X		X		X		X
X		X	X		X		X		X
X							X		S
X		X	X		X		X		X
X		X	X		X		X		X
X	E	X	X						X
X	X	X	X	X	X	X	X	X	X

Legenda: X – parede; H – herói; S – saída; D – dragão; E – espado; em branco – espaço livre.

1.2 Receber comandos do utilizador para movimentação do herói (H) de uma posição de cada vez (cima, baixo, esquerda, direita). Se existir uma parede na direção pretendida, o herói mantém a sua posição. Se o herói chegar à posição da espada, apanha a espada, passando a ser representado por "A" (herói armado) (a espada deixa de ser representada). Se o herói chegar a uma posição adjacente ao dragão e estiver desarmado, o jogo termina com a morte do herói; se estiver armado, o dragão morre e desaparece. A cada comando, o programa atualiza e mostra a nova situação do labirinto. O jogo termina quando o herói chega armado à saída depois de ter morto o dragão, ou quando o herói é morto pelo dragão. O herói só pode deslocar-se para a posição de saída depois de ter morto o dragão.

1.3 Organizar o programa por forma à interação com o utilizador (pela linha de comando) e a lógica do jogo serem tratadas em *packages* separados (cada um com uma ou mais classes), como por exemplo `maze.cli` (*command line interface*) e `maze.logic`. Isto é importante para facilitar posteriormente a existência em paralelo de múltiplas formas de usar/exercitar a lógica do jogo: através da interface alfanumérica, através de testes unitários automáticos (a criar numa próxima aula) ou através de uma interface gráfica (a criar numa próxima aula). O *package* com a(s) classe(s) responsável(eis) pela interação com o utilizador (através de `System.in` e `System.out`) deve conter pelo menos uma classe com um método `main` (ponto de entrada no

programa) e eventuais métodos auxiliares. Recomenda-se colocar o ciclo do jogo na camada de interação com o utilizador; esse ciclo consiste em repetidamente pedir um comando ao utilizador, invocar a ação correspondente na camada de lógica do jogo e imprimir o novo estado do labirinto, até o herói morrer ou chegar à saída. O *package* com a lógica do jogo deve conter pelo menos as seguintes classes (com todos os campos privados ou protegidos, não estáticos):

- **a)** Uma classe para representar o estado do jogo, agregando o labirinto em si (terreno do jogo) e os elementos presentes no labirinto (em campos privados), e disponibilizando uma API (conjunto de construtores e métodos públicos) a usar pela camada de interação com o utilizador para: instanciar um novo jogo (eventualmente passando como argumento uma matriz com o estado inicial pretendido); dar ordem de movimentação do herói numa determinada direção; saber se o jogo terminou e de que forma; consultar o estado do jogo por forma a conseguir imprimir o labirinto (uma possibilidade é usar o método `toString`);
- **b)** Uma classe para representar o labirinto em si (terreno do jogo);
- **c)** Classes para representar os vários tipos de elementos que podem estar presentes no labirinto (espada, dragão, herói), com o respetivo estado, e uma superclasse com as propriedades comuns (posição, etc.), tirando o mais possível partido de herança e polimorfismo (podendo o comportamento dos vários elementos ser distribuído pelas classes respetivas).

1.4 [Para casa] Após cada comando de movimentação do herói referido em 1.2, o programa deve mover aleatoriamente o dragão de 1 posição (cima, baixo, esquerda, direita, manter). Se o dragão estiver na mesma posição da espada, deve ser impressa a letra "F"; quando o dragão se afasta, voltam a aparecer as letras "E" e "D". Se o dragão se movimentar para uma posição adjacente ao herói, morre o herói ou o dragão, conforme o herói está desarmado ou armado.

1.5 [Para casa] Criar uma nova estratégia em que o dragão pode adormecer por algum tempo de forma aleatória, tendo um visual diferente quando está a dormir. Após cada comando de movimentação do herói referido em 1.2, o programa deve dar oportunidade ao dragão de se manter como está, acordar (se estava a dormir), adormecer (se estava acordado), ou mover-se aleatoriamente de 1 posição (se estava acordado). 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 o estado do dragão 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).