Projeto Guiado – 4ª Iteração

Desenvolvimento da interface gráfica para o utilizador – 2ª parte

4.1. Em vez de mostrar o estado do jogo numa área de texto, mostrar esse estado num painel cujo conteúdo é desenhado pelo programa usando imagens sugestivas. Para esse efeito, é necessário criar uma classe derivada de `JPanel` e implementar o método `paintComponent`.
Sugestão: Ver slides sobre Swing e exemplo "GraphicsDemo.java".

4.2. Permitir a movimentação do herói através das setas do teclado. Para esse efeito, é necessário criar um `KeyListener` e implementar o método `KeyPressed`. Para apanhar os eventos do teclado pode ser necessário usar o método `requestFocus`. Pode ser mais conveniente usar janelas separadas para configurar as opções do jogo (tamanho do labirinto, número de dragões e tipo de dragões) e para jogar.
Sugestão: Ver exemplo "GraphicsDemo.java".

4.3 Criar a possibilidade de construir manualmente um labirinto, de forma interativa, usando o rato. Deve ser possível limpar/preencher quadrículas com cliques do rato. Deve também ser possível colocar objetos (espada, herói, dragões, etc.) no labirinto usando o rato.

4.4 [Opcional, não conta para avaliação!] Acrescentar a possibilidade de gravar a situação atual do jogo num ficheiro, bem como carregar um jogo previamente gravado num ficheiro.
Sugestão: ver os slides sobre *input/output*, nomeadamente a parte sobre serialização.

Finalização e entrega

4.5. Documentar API da camada de lógica do jogo com Javadoc
Documentar com comentários de documentação em Javadoc (/** …*/) as classes e métodos públicos da camada (package) de lógica de jogo que são utilizadas pelas camadas acima (interface gráfica, interface de linha de comando, testes unitários). Gerar a documentação da API em HTML usando a ferramenta Javadoc.
Referências:

- http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html
- http://www.devdaily.com/java/edu/pj/pj010014

4.6. Garantir que os testes têm a cobertura adequada
Para obter 100% de classificação no que se refere aos testes, os testes unitários em JUnit devem cobrir pelo menos 80% das linhas (ou instruções) da camada de lógica do jogo (usando ferramenta Eclemma) e pelo menos 60% dos mutantes gerados com a ferramenta Pitest.

Submissão e avaliação

Submeter através do moodle até ao fim do dia 4 de abril um ficheiro zip com a pasta completa do projeto, contendo pelo menos os ficheiros de código fonte e os ficheiros HTML gerados com Javadoc.

O trabalho será avaliado com base nos seguintes critérios e pesos:

1. Cumprimento das funcionalidades pedidas (50%)
    - Lógica base (10%)

- Geração aleatória de labirinto (7.5%)
- Funcionalidades avançadas dos dragões e do herói (7.5%)
- Interface por linha de comando (5%)
- Interface gráfica (jogar, configurar, construir) (20%)

2. Qualidade do código e dos testes (40%);
   - Organização modular em *packages* (separação das camadas em *packages* distintos com responsabilidades e dependências adequados) (10%);
   - Estrutura de classes tirando partido dos mecanismos da orientação por objetos (abstração, encapsulamento, herança, polimorfismo) (10%);
   - Qualidade do código dentro dos métodos (código duplicado, legibilidade, etc.) (5%);
   - Completude e qualidade dos testes (10%);
   - Completude e qualidade da documentação Javadoc (5%).
3. Acompanhamento ao longo das aulas (10%)