Instituto Politécnico de Beja
Escola Superior de Tecnologia e Gestão

Licenciatura em Engenharia Informática
Programação Orientada por Objectos

Boulder Dash

Trabalho Prático de Avaliação

3 de maio de 2021

So come out of your cave walking on your hands
And see the world hanging upside down
– The Cave – Mumford & Sons

https://youtu.be/fNy8llTLvuA

1 Introdução

Este enunciado é algo extenso. Tal significa que deve ser lido mais do que uma vez e com muita atenção. O que é pedido está classificado em três tipos de **REQUISITOS**:

1. **ESSENCIAIS**
2. **NÃO ESSENCIAIS**
3. **QUE IMPLICAM PENALIZAÇÕES.**

Os essenciais permitem obter 10 valores, mas apenas desde que não estejam presentes motivos para penalizações, ou seja, se cumprir totalmente todos os requisitos que podem implicar penalizações. Para não ter penalizações basta ter cuidado a ler o enunciado e respeitar o que é requerido. Assim, poderá garantir que o programa entregue não irá oferecer motivos para a aplicação de penalizações, as quais podem fazer baixar a nota para níveis certamente indesejados. Os requisitos não essenciais permitem obter até 18 valores. Os 19 e 20 valores ficam apenas para quem fez tudo o que é pedido de forma absolutamente impecável e inventou algo mais para fazer, os chamados **EXTRAS**. Esses extras têm de ter uma dificuldade superior ao que é pedido embora devam continuar na linha do que é pedido e utilizando apenas os conteúdos que já foram dados nas aulas. Não vale a pena fazer extras sem ter TUDO o que é pedido feito. Também não vale a pena fazer os requisitos não essenciais sem ter feito os essenciais. Em resumo:

- Só pode ter mais do 10 valores se tiver cumprido totalmente os requisitos essenciais e não tiver penalizações.

- Só pode ter mais do 18 valores se tiver cumprido totalmente os requisitos essenciais e também os não essenciais e não tiver penalizações.

Para mais informações pode e deve ler as regras de avaliação no guia de funcionamento da unidade curricular.

2 O Jogo

Neste trabalho prático pretende-se criar um programa para jogar uma variante de um jogo clássico de *Arcade* de 1984, o *Boulder Dash* (`https://en.wikipedia.org/wiki/Boulder_Dash`). De acordo com a Wikipedia, este é um jogo criado em 1983 pelos programadores canadianos Peter Liepa and Chris Gray. Hiroyuki Imabayashi, e publicado em 1984 pela First Star Software, uma *software house* sediada em Chappaqua no estado de Nova Iorque.

A versão que nos vai servir de guia está disponível em `https://www.onlinegames.com/boulderdash/`. Pode também aproveitar a informação em `https://www.c64-wiki.com/wiki/Boulder_Dash` para melhorar a sua versão do jogo. Mas note que não tem de implementar todas as funcionalidades presentes ou referidas nesses links. Deve fazer o que é requerido neste enunciado.

Note que a interface do programa TicTacToe, já feito nas aulas, tem uma grelha de botões (em lugar de labels) e a forma de comunicação entre o model e a interface pode ser feita de forma idêntica.

O jogo que se pretende implementar dever ter as seguintes características:

1. O jogo decorre numa grelha em que cada uma das posições é de um dos seguinte tipos:
   (a) Túnel ocupado (`OccupiedTunnel`) – os pedregulhos que caem não podem passar por essas posições; o túnel ocupado passa a livre quando o Rockford passa por lá;
   (b) Túnel livre (`FreeTunnel`) – os pedregulhos que caem podem passar por essas posições;
   (c) Portão (`Gate`) – uma posição que só surge (a piscar) quando o nível termina e por onde o Rockford pode sair;
   (d) Muro (`Wall`) – uma posição intransponível.
2. O jogo pode ter vários objectos imóveis, bem como objectos e personagens móveis: os objectos móveis são os diamantes e os pedregulhos; os objectos imóveis são bónus de pontuação; as personagens móveis são o Rockford (controlado pelo jogador) e os seus inimigos.
3. O tabuleiro de jogo – de um dado nível – é lido de um ficheiro de texto em que uma letra diferente é utilizada para cada posição; no início, o ficheiro de texto indica a quantidade de linhas e de colunas do tabuleiro, depois o tabuleiro com as letras e finalmente a posição de cada objectivo especial; o formato deve suportar comentários iniciados por %. Por exemplo:

```
4 13
WWWWWWWWWWWWW
OOOOOOOWOOOOW
```

```
WWWLLOOOWWWWW
WWWWWWOWWWWWW
J 1 2 % posição do jogador
D 1 5 1 7 % dois diamantes nas posições (1,5) e (1,7)
E 2 3 % um inimigo que se move aleatoriamente dentro das posições livres (chão livre)
I 2 4 % posição com inimigo imóvel
B 3 7 % um bónus de pontuação
```

Neste exemplo a letra `W` representa o muro (nada pode deslocar-se para essa posição), a letra `O` uma posição de túnel ocupada, a letra `L` uma posição de túnel livre e a letra `P` um pedregulho (que pode cair se a posição do túnel, imediatamente abaixo, ficar livre).

4. O Rockford tem de apanhar todos os diamantes no nível em que está para que surja o portão por onde terá de sair para concluir esse nível;
5. O Rockford apenas pode deslocar-se para locais com chão livre ou ocupado ou para locais com diamantes, ou ainda para locais com bónus; quando o Rockford vai para uma posição com um diamante, o diamante desaparece do tabuleiro ficando o Rockford no lugar onde estava o diamante; o diamante é contabilizado na pontuação;
6. os diamantes e os pedregulhos apenas de deslocam de cima para baixo e em posições de túnel livre;
7. O jogo termina quando todos os níveis forem completados.
8. O jogo progride passo a passo; em cada passo o Rockford move-se para uma posição adjacente (estas são apenas na horizontal ou vertical, nunca na diagonal); também em cada passo os pedregulhos e os inimigos podem mover-se;
9. O movimento dos inimigos e dos pedregulhos deve ser independente do movimento do Rockford: os pedregulhos só se podem mover de cima para baixo enquanto há posições de túnel livre por baixo (até chegarem à base ou a uma posição de chão ocupado ou a uma parede).

Seguem-se os requisitos para o trabalho. Antes de escrever código, leia-os atentamente mais do que uma vez. Antes de entregar o trabalho leia-os, pelo menos, mais uma vez.

3 Requisitos Essenciais

O incumprimento de qualquer destes requisitos implica uma classificação negativa no trabalho e a não contabilização dos requisitos não essenciais. Nos requisitos seguintes, o movimento do Rockford pode ser programado indicando as coordenadas da posição para onde pretende deslocar-se, por exemplo, para ir para a linha 2 e coluna 3 seria `rockford.gotoPosition(2, 3)`. Note que cada uma nos pontos nos requisitos seguintes pode implicar várias linhas de código.

Req. E1 - Classes a definir (0,5 valores) A implementação do jogo deve respeitar os seguintes pontos:

- O tabuleiro deve ser definido por um objeto de uma classe `Board` que contém um *array* de *array*s OU uma lista de listas (`java.util.List` de `java.util.List`) de objetos de um tipo `AbstractPosition`.

- O tipo `AbstractPosition` deve ser definido por uma classe abstracta;
- Cada objeto do tipo `AbstractPosition` deve também ser de um dos seguintes tipos, todos eles classes: `OccupiedTunnel`, `FreeTunnel`, `Gate` ou `Wall`.
- Os diamantes, e outros objectos móveis e imóveis, devem ser objetos da respetiva classe que sabem em que linha e coluna estão no tabuleiro;
- O Rockford deve ser um objeto da classe `Rockford` que sabe em que linha e coluna está no tabuleiro; este objeto deve ser um *singleton* pelo que a classe deve implementar essa *pattern*; e.g. http://www.javaworld.com/article/2073352/core-java/simply-singleton.html

Req. E2 - Testes do model (4,0 valores) Deve ter código de teste para os seguintes cenários; cada um dos métodos deve ter o nome `testN` em que `N` é letra do cenário na seguinte lista (testeA, testeB):

1. Movimento para posição livre;
2. Tentativa de movimento para posição ocupada por parede;
3. Tentativa de movimento do Rockford para posição ocupada por um pedregulho;
4. Movimento do Rockford para uma posição com um diamante; este deve desaparecer e pontuação deve aumentar;
5. Movimento do Rockford para uma posição que faz um diamante cair, atrás de si, até à última posição livre (túnel livre);
6. Movimento para o portão e que termina o jogo com sucesso.

Req. E3 - Interface com o utilizador (1,5 valores) Escreva o código necessário para criar o tabuleiro constituído por uma grelha de objetos da classe `javafx.scene.control.Button` ou de uma classe que herde desta. Cada botão deve ter uma imagem diferente correspondente ao objeto que representa (Rockford, túnel livre, túnel ocupado ou parede).

Req. E4 - Leitura do tabuleiro (nível) de ficheiro (2,0 valores) O tabuleiro apresentado deve ser lido de um ficheiro de texto com o formato indicado na número 3 na Secção 1.

Req. E5 - Jogo sem objectos móveis (1,5 valores) Escreva o código necessário para que seja possível jogar utilizando a interface gráfica definida no requisito anterior. O jogo deve detectar o final, quando todos diamantes foram apanhados e o Rockford chegou ao portão e deve também permitir a desistência do jogo voltando ao tabuleiro inicial. O jogo não necessita ter pedregulhos, nem inimigos móveis nem movimento dos diamantes.

Req. E6 - Comandos com o teclado (0,5 valores) Deve ser possível comandar o Rockford com o teclado. As teclas devem estar indicadas numa caixa de diálogo antes do primeiro jogo.

4 Requisitos Não Essenciais

Estes requisitos só são contabilizados se os requisitos essenciais estiverem totalmente correctos.

Req. NE1 - Objetos móveis (2,0 valores) Considere objectos móveis: diamantes (com movimento), pedregulhos que caem, e inimigos móveis. Note que os objetos que caem, só o podem fazer através de posições livres do túnel.

Req. NE2 - Vidas e apresentação da pontuação (0,5 valores) A pontuação é definida em função da quantidade de diamantes recolhidos e da quantidade dos movimentos efectuados. A função é a seguinte:

$$points = NDiamonds * 100 - 5 * NMovements$$

O nome do jogador é perguntado no início e deve ter um máximo de 8 caracteres. No final de cada jogo, deve surgir num painel ao lado do tabuleiro de jogo a pontuação conseguida, o nome do nível atingido e a lista das 5 maiores pontuações até à data. Se a pontuação conseguida fizer parte dessa lista, deve surgir assinalada com *** logo após os pontos. Considere 5 vidas para o Rockford.

Req. NE3 - Leitura e escrita da pontuação para ficheiro (1,0 valores) No final de cada jogo, a pontuação obtida deve ser acrescentada ao ficheiro de pontuações. Este é um ficheiro de texto com o nome `scoresAAAAMMDD.txt` em que AAAA é o ano, MM é o mês e DD o dia do mês, por exemplo, scores20210609.txt para 9 de Junho de 2021. A pontuação tem a informação indicada na alínea anterior: o nome de cada jogador, o nível e os pontos obtidos. A pontuação apresentada no final tem de ter em conta o ficheiro com a pontuação mais recente, se existente.

Req. NE4 - Registo de movimentos em janela (1,0 valores) Adicione a possibilidade de registar os movimentos do Rockford numa segunda janela que funciona como uma segunda *view*. Para tal as linhas são denominadas por letras maiúsculas e as colunas por números. As jogadas são registadas num dos seguintes dois formatos:

```
rockford A3 -> B3
rockford B3 -> B4
...
```

Req. NE5 - Níveis (0,5 valores) Adicione níveis ao jogo. Tal implica ler um mapa diferente para cada nível sempre que termina o nível anterior.

Req. NE6 - Tempo (0,5 valores) Adicione no topo de janela um cronómetro com minutos e segundos para o tempo em cada nível O cronómetro inicia a contagem no início de cada nível e pára no fim de cada nível ou quando o jogador perde.

Req. NE7 - Leitura de percursos animada ou final (1,5 valores) Adicione a possibilidade do programa escrever os percursos feitos pelo Rockford em ficheiro. Depois o programa deve oferecer a possibilidade de *replay* automático do jogo, no final de cada nível, utilizando a tecla R. Também deve ser possível fazer *replay* de um nível gravado em ficheiro através de uma opção no arranque do programa.

Req. NE8 - Diagrama de classes (1,0 valores) Adicione um diagrama de classes para o código escrito, mas sem incluir o código de teste. O diagrama deve incluir as classes e interfaces escritas para este trabalho. Para essas classes e interfaces basta apresentar o nome (não deve indicar nomes de atributos nem de métodos). O diagrama deve apresentar relações entre essas classes: associação simples, agregação partilhada e composta, herança, realização e utilização. Devem também ser indicadas relações de utilização entre as classes e interfaces escritas e as packages de bibliotecas utilizadas (JavaFX).

5 Requisitos que podem implicar penalizações ou bonificações

O incumprimento de um ou mais dos seguintes requisitos implica a atribuição da penalização especificada e a automática impossibilidade de obter uma nota superior a 18.

Req. PB1 - Utilização de polimorfismo (-3,0 a 3,0) valores) A utilização de polimorfismo quando o Rockford tenta ir para uma posição adjacente deve estar presente. Mais especificamente, cada objecto (pedregulho, túnel vazio, túnel cheio, muro) devem decidir o que sucede quando o Rockford chega a essa posição. Também o movimento de cada objecto pode utilizar polimorfismo: cada objecto tem um método move(...) polimórfico.

Req. PB2 - Controlo de versões (-20,0 a 0.0 valores) antes do dia 15 de maio a resolução ainda incompleta deste trabalho terá de estar num repositório privado no github ao qual terá de ser dado acesso aos três professores da unidade currcular utilizando os seguintes usernames: O trabalho será penalizado se a data de 15 maio não for cumprida. Também haverá penalização se não existirem actualizações consideradas regulares para o repositório; por exemplo, a maior parte do programa não deve surgir no repositório muito perto da data de submissão para avaliação.

Req. PB3 - *View/controller* (*gui*) separada do *model* (-4,0 a -1.0 valores) A *view/controller* (*gui*) apenas deve tratar de comunicar ao *model* o que o jogador fez (por exemplo um clique num botão, uma opção de menu, etc.) e fazer na interface gráfica as alterações pedida pelo *model*. Cada view também pode utilizar *getters* do *model* para obter informação, mas **toda** a informação e lógica do jogo deve estar presente no *model*.

Req. PB4 - Packages (-1,0 valores) Todo o código deve estar definido num *package* com o nome `pt.ipbeja.estig.po2.boulderdash`. O código de interface deve estar num *package* com o nome `pt.ipbeja.estig.po2.boulderdash.gui`. O restante código deve estar num *package* com o nome `pt.ipbeja.estig.po2.boulderdash.model`.

Req. PB5 - Métodos com mais de 30 pontos e vírgulas (-4,0 a -2,0 valores) Nenhum método deve ter mais de trinta pontos e vírgulas (";"). Note que um ciclo `for` tem dois pontos e vírgulas. Deve preferir métodos pequenos. Deve optar por mais métodos pequenos em lugar de menos métodos grandes.

Req. PB6 - Utilização de testes de tipo (-4,0 valores) A utilização do operador `instanceof`, do método `getClass` (`https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html#getClass--`), ou algo idêntico que permita saber o tipo de dados de um objecto, será penalizado.

Req. PB7 - Regras de estilo e elegância do código (-4,0 a 1,0 valores) O código entregue deve respeitar as regras de estilo, nomeadamente **todas** as seguintes:

Utilização de asserts **Sempre que conveniente, os métodos devem utilizar asserts para testar os valores dos parâmetros ou valores de outras variáveis.**

Identificadores em inglês Os nomes de todas as variáveis, métodos e classes devem estar em inglês.

Nomes das variáveis, métodos, constantes e classe Utilização de letras minúsculas/maiúsculas e informação transmitida pelos nomes; por exemplo, `box` é um melhor nome para uma caixa do que `b` ou `xyz`.

Constantes Não deve utilizar constantes literais (por exemplo, 20, -300, 45.4) para representar valores que podem ser melhor representados por um nome. Nesses casos deve definir constantes utilizando a palavra reservada **`final`**.

Os espaçamentos Depois das vírgulas e antes e depois dos operadores.

Indentação coerente e para cada bloco, incluindo posicionamento e utilização coerente das chavetas.

Utilização de parâmetros Sempre que conveniente, os métodos devem utilizar parâmetros de modo a evitar duplicação de código.

Repetição de código Deve ser evitada a repetição de código.

Comentários Antes de cada método deve escrever um comentário javadoc (`/** */`) que explique o que o método faz, os respectivos parâmetros e valor devolvido (se existentes). Os comentários podem estar em português mas tente colocá-los em inglês. Os comentários têm de incluir *tag*s `@param` para cada parâmetro e `@return` quando necessário.

Utilização do `this` Utilização das referências antes do nome das operações. Por exemplo, **`this`**`.add(line)`.

Ocultação de informação Todos os atributos devem ser `private`.

Req. PB8 - Auto-avaliação (-2,0 a 0,0 valores) Num ficheiro "auto-aval.txt", deve indicar quais os requisitos que estão **totalmente** cumpridos (os únicos que contam como cumpridos) e a classificação resultante. No mesmo ficheiro **deve indicar quantas horas gastou a fazer este trabalho, incluindo o tempo em aulas e fora das aulas (trabalho autónomo)**.

Req. PB9 - Identificação (-1,0 valores) Todos os ficheiros com código (ficheiros *.java) têm de conter, em comentário no início, o nome e número de aluno do autor.

Req. PB10 - Quantidade de autores (-20,0 valores) O trabalho deve ser realizado individualmente ou em grupos de dois; quando for feito por dois alunos ambos os alunos terão de entregar a mesma resolução no prazo definido.

Req. PB11 - Nome do projecto (-1,0 a 0,0 valores) O nome do projecto no inteliJ tem de respeitar um dos seguintes formato conforme o trabalho seja feito por um ou dois autores.

```
Numero_PrimeiroUltimo_TP02_PO2_2020-2021
Por exemplo: 1232_VanessaAlbertina_TP02_PO2_2020-2021
Numero_PrimeiroUltimo_Numero_PrimeiroUltimo_TP02_PO2_2020-2021
Por exemplo:
12324_VanessaAlbertina_TP02_PO2_2020-2021
12324_VanessaAlbertina_12324_AnaLee_TP02_PO2_2020-2021
```

Note que `Primeiro` e `Ultimo` representam nomes de autores. `Numero` representa um número de aluno do autor. O trabalho entregue é o zip gerado pelo IntelliJ com o comando em File->Export->Project to Zip File... .

Req. PB12 - Originalidade (-20,0 a +3,0 valores) A originalidade das soluções encontradas para resolução dos requisitos essenciais e não essenciais, comparativamente com outros trabalhos entregues ou disponíveis na internet, pode ser valorizada num máximo de 3 valores e penalizada num máximo de 20,0 valores. Para efeitos de aplicação desta valorização ou penalização,

a originalidade é determinada pelas diferenças ou semelhanças entre o trabalho a ser avaliado e os restantes trabalhos entregues por outros alunos e disponíveis noutras fontes; se utilizar código encontrado na internet pode referir esse facto indicando a fonte e autor como forma e diminuir ou até evitar totalmente este penalização;

Req. PB13 - Entrega do trabalho e eventual apresentação (-20,0 a 0,0 valores) O trabalho entregue tem de ser uma directoria do projecto IntelliJ pronto a funcionar, sob a forma de um único ficheiro zip contendo toda a directoria mais o ficheiro de auto-avaliação. Antes de entregar, verifique que sabe pôr a funcionar o código no ficheiros zip entregue. Para tal parta desse ficheiro, descompacte-o, e leia-o no IntelliJ. Tal poderá ser requerido numa eventual apresentação individual do trabalho. Se não conseguir pôr a funcionar o conteúdo do ficheiro zip entregue (no moodle e por email) a classificação no trabalho poderá ser de zero valores. A entrega tem de ser feita num ficheiro no formato zip com o nome indicado no Requisito PB11 e sempre por **duas** vias:

1. Na página da disciplina.
2. Por mensagem directa para os trÊs docentes da unidade curricular no Teams.

Note que pode entregar mais do que uma vez, desde que dentro do prazo. A última entrega dentro do prazo é a única que conta.

Req. PB14 - Data limite de entrega em época normal O trabalho deve ser entregue no moodle e também por email até às **24:00 de 23 de junho de 2021.**. Não deve assumir que o relógio do sistema está igual a qualquer outro. Assim, a entrega depois da uma hora (da madrugada) do dia seguinte ao prazo de entrega implicará zero valores no trabalho.

Req. PB15 - Data limite de entrega em época de recurso O trabalho deve ser entregue no moodle e também por email até às **24:00 de 13 de julho de 2021.**. Não deve assumir que o relógio do sistema está igual a qualquer outro. Assim, a entrega depois da uma hora (da madrugada) do dia seguinte ao prazo de entrega implicará zero valores no trabalho.

Finalmente, note que necessita de realizar mais do que o pedido para obter mais de 18 valores. A criatividade também pode justificar uma melhor classificação pelo que extras originais e sofisticados serão uma boa aposta. Note que estas adições só contam para a classificação do trabalho se forem consideradas suficientemente significativas e se **todos** os requisitos estiverem completamente cumpridos e sem penalizações.

Lembre-se que a originalidade (Req. PB12 da Secção 5) é outra forma de subir a pontuação e contribuir para obter mais do que 18 valores

6 Nota importante

Todas as contribuições para o trabalho que não sejam da exclusiva responsabilidade dos autores têm de ser identificadas (com comentários no código e referências no relatório) e a sua utilização bem justificada e expressamente autorizada pelo professor responsável. Justificações insatisfatórias, ausência de autorização, ou ausência de referências para trabalhos ou colaborações externas utilizadas serão consideradas fraude sempre que o júri da unidade curricular considere os trabalhos demasiado semelhantes para terem sido criados

de forma independente e **tal terá como consequência a reprovação direta na unidade curricular de todos os alunos envolvidos sem possibilidade de realizar época de recurso ou especial**. Esta decisão de reprovação pode suceder até final do semestre logo que seja detectada a situação e independentemente de já ter sido ou não publicitada uma nota aos alunos. Assim, nenhum aluno deve dar cópia do seu código (ainda que em fase inicial) a outro. Por essa mesma razão não é boa ideia partilhar código com os colegas, quer directamente quer através do fórum. Naturalmente, cada aluno pode e deve trocar impressões e esclarecer dúvidas com todos os colegas, mas deve saber escrever todo o código sozinho. A classificação neste trabalho prático fica dependente de uma eventual apresentação individual do mesmo, tal como previsto no guia da unidade curricular e pode ser alterada em resultado do desempenho do aluno nessa mesma apresentação.

Caso o júri detecte, antes do final do semestres, algum tipo de ocorrência que considere anómala, a avaliação do trabalho poderá ser repetida e contabilizada apenas esta segunda avaliação.

Finalmente, antes de entregar leia com MUITA atenção todo o enunciado. A falha de parte do exigido num requisito implica o não cumprimento desse requisito e consequente penalização. A programação é também a atenção aos detalhes.

Bom trabalho!
João Paulo Barros