Instituto Politécnico de Beja
Escola Superior de Tecnologia e Gestão
Licenciatura em Engenharia Informática
Programação Orientada por Objectos

Tabuleiro de Xadrez com Regras

Trabalho Prático de Avaliação

Beja, 24 de Maio de 2019

I'm only happy when it rains
I'm only happy when it's complicated
– Only happy when it rains – Garbage

https://www.youtube.com/watch?v=GpBFOJ3R0M4

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 17 valores. Os 18, 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 17 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 ficar com um programa que permita a dois jogadores jogar xadrez. O computador não vai saber jogar Xadrez. Vai apenas saber as principais regras e deve utilizá-las para indicar as jogadas possíveis para cada peça e para registar as jogadas feitas.

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

1. O jogo decorre num tabuleiro de Xadrez o qual pode ser representado por uma grelha de botões; veja a página `https://pt.wikipedia.org/wiki/Xadrez`;
2. Pode utilizar texto em cada botão para indicar qual a peça que está nessa posição. Cada peça deve utilizar os seguintes nomes:
   - p - peão (note que o p é minúsculo)
   - B - Bispo
   - C - Cavalo
   - T - Torre
   - R - Rei
   - D - Rainha (Dama)
3. A cor de fundo do botão pode ser utilizada para a cor dessa posição do tabuleiro.
4. Para indicar a cor da peça em texto, utilize o prefixo "B" para uma peça branca e "P" para uma peça preta. Por exemplo:

   P D significa rainha preta;

   B p significa peão preto;

   P B significa bispo preto;

   B B significa bispo branco.
5. Como referência para notação utilize a página `https://pt.wikipedia.org/w/index.php?title=Nota%C3%A7%C3%A3o_alg%C3%A9brica_de_xadrez&oldid=55090248`;
6. Como referência para as regras, nomeadamente para a movimentação das peças, utilize a página `https://pt.wikipedia.org/wiki/Xadrez`.

Seguem-se os requisitos para o trabalho. Antes de escrever o 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.

Req. 1 – Código de Teste (6,0 valores) Para as peças (bispo, torre, rainha e rei) escreva três métodos de teste (num total de 12 métodos de teste):

1. Quando selecciona uma posição no tabuleiro em que está uma peça, o método possibleMoves deve devolver uma lista de posições para onde a peça se pode mover sem capturar uma peça adversária.
2. Quando selecciona uma posição no tabuleiro em que está uma peça, o método possibleTakes deve devolver uma lista de posições para onde a peça se pode mover e capturar uma peça adversária.
3. Quando a peça se move para uma posição livre (o utilizador indicou a posição inicial e final), o método de teste verifica que de facto a peça ficou na posição final e já não está na posição inicial.

Note que em cada posição do tabuleiro apenas pode estar uma ou zero peças.

Req. 2 – Classes para cada peça (1,0 valores) Para a realização do requisito 1 deve definir uma classe para cada um dos seguintes tipos de peça: bispo, torre, rainha e rei. Essas classes devem herdar de uma classe abstract `Piece` que deve conter os métodos abstractos `List<Position> possibleMoves()` e `List<Position> possibleTakes()`, bem como uma referência para o tabuleiro (*Board*). Assim, cada classe deve conter estes métodos e deve receber no construtor o objecto do tipo tabuleiro (`Board`).

Req. 3 – Gravação dos movimentos em ficheiro (2,0 valores) Todas as jogadas devem ficar registadas em ficheiro. A notação utilizada deve ser a exemplificada em pt.wikipedia.org no Exemplo 1 (1):

> "
> (1) escritas em duas colunas, como um par branco/preto, precedido pelo número da jogada e um ponto:
> 1. e4 e5
> 2. Cf3 Cc6
> 3. Bb5 a6
> "

Cada classe que herda de `Piece` deve ter um método `String movementText(Position begin, Position end)` que recebe duas posições e devolve uma String para escrever em ficheiro.

Req. 4 – Desenho do Tabuleiro (1,0 valores) O programa deve desenhar um tabuleiro de Xadrez com um botão para cada posição, as peças nas posições iniciais e *labels* à esquerda e direita e por cima e por baixo do tabuleiro.. Veja por exemplo o tabuleiro na página em `https://pt.wikipedia.org/wiki/Xadrez`. O resultado da leitura do requisito 3 deve ser vísível neste tabuleiro.

4 Requisitos Não Essenciais

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

Req. 5 – Jogo (3,0 valores) Deve ser possível utilizar o tabuleiro na interface gráfica para fazer jogadas para posições livres e para capturar peças. Para tal deve clicar na posição de origem e na posição de destino. Quando clica na posição de origem, todas as posições livres para onde a peça se pode mover devem ficar assinaladas a verde e todas as posições para onde a peça se pode mover capturando uma peça adversária devem ficar a amarelo. Com o segundo clique devem ser atualizadas as posições das peças.

Req. 6 – Leitura das posições em ficheiro (2,0 valores) Deve ser possível ler de um ficheiro que utiliza a notação exemplificada na wikipedia no Exemplo 1 (1), uma situação de jogo, ou seja quais as peças que estão em cada posição. Depois deve ter um método de teste que lê e testa se as peças estão de facto nas posições especificadas em ficheiro.

Req. 7 – Código de Teste (2,0 valores) Para as peças peão e cavalo escreva três métodos de teste (num total de 6 métodos de teste):

1. Quando selecciona uma posição no tabuleiro em que está uma peça, o método possibleMoves deve devolver uma lista de posições para onde a peça se pode mover sem capturar uma peça adversária.
2. Quando selecciona uma posição no tabuleiro em que está uma peça, o método possibleTakes deve devolver uma lista de posições para onde a peça se pode mover e capturar uma peça adversária.
3. Quando a peça se move para uma posição livre (o utilizador indicou a posição inicial e final), o método de teste verifica que de facto a peça ficou na posição final e já não está na posição inicial.

Para realização deste requisito deve definir uma classe para cada tipo de peça. Essa classes deve herdar de uma classe abstract `Piece` que deve conter os métodos abstractos `List<Position> possibleMoves()` e `List<Position> possibleTakes()`, bem como uma referência para o tabuleiro (*Board*). Assim, cada classe deve conter estes métodos e deve receber no construtor o objecto do tipo tabuleiro (`Board`).

5 Possíveis extras

- Em lugar de texto, utilizar imagens para todas as peças;
- Considerar as jogadas "pequeno roque" e "grande roque"e captura pelo peão "en passant" (`https://pt.wikipedia.org/wiki/En_passant`).
- Detectar o cheque ao rei; depois de uma peça de mover, se ficar a atacar o rei adversário deve surgir a palavra cheque, por exemplo na parte inferior da janela (por baixo do tabuleiro) de jogo;
- Considerar os movimentos proibidos do rei: o rei nunca se pode mover para uma posição onde possa ser capturado na jogada seguinte (onde esteja a ser atacado); com isto já pode detectar o cheque mate.
- Adicionar a possibilidade de o programa jogar sozinho utilizando as jogadas gravadas em ficheiro. Veja o exemplo do programa fifteen disponível na página moodle de PO2.

6 Requisitos que podem implicar penalizações

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 17.

Req. 1 - View/controller separada do Model e IMPLEMENTAÇÃO do padrão Observer-Observable (-1,0 a -4.0 valores) Cada view/controller apenas deve tratar de comunicar ao model o que o jogador fez (por exemplo um clique num botão) e fazer na interface as alterações pedida pelo *Model*. Cada view também pode utilizar *getters* do *Model* para obter informação. Toda a informação e lógica do jogo deve estar presente no *Model*. deve ser possível remover o código relativo à interface

Req. 2 - Packages (-1,0 valores) Todo o código deve estar definido num *package* com o nome `pt.ipbeja.estig.po2.chess`. O código de interface deve estar num *package* com o nome `pt.ipbeja.estig.po2.chess.gui`. O restante código deve estar num *package* com o nome `pt.ipbeja.estig.po2.chess.model`.

Req. 3 - Métodos com mais de 30 pontos e vírgulas (-2,0 a -4,0 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. 4 - Utilização do operador `instanceof` ou do método `getClass` (-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. No entanto, podem ser utilizados, sem penalização, nos métodos equals e compareTo.

Req. 5 - Regras de estilo e elegância do código (-1,0 a -4,0 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. 6 - Auto-avaliação (-2,0 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. 7 - 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. 8 - Quantidade de autores (-20,0 valores) O trabalho deve ser realizado por um ou por dois alunos. Ambos devem submeter a resolução.

Req. 9 - Nome do projecto (-1,0 valores) O nome do projecto no IntelliJ tem de respeitar o seguinte formato. Note que `Primeiro1` e `Ultimo1`, `Primeiro2` e `Ultimo2` representam os nomes dos autores. `Numero1` e `Numero2` representam os números de aluno de cada autor: Numero1Primeiro1Ultimo1-Numero2Primeiro2Ultimo2-TP02PO2-2018-2019, por exemplo 1232VanessaAlbertina-34564MariaAlbertina-TP02PO2-2018-2019

Req. 10 - 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.

Req. 11 - Entrega e apresentação do trabalho (até -20,0 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 na 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 9 e sempre por **duas** vias:

1. Na página da disciplina.
2. Por *email*, respeitando as seguintes regras: A entrega por *email* é feita para `joao.barros@ipbeja.pt`. O *email* a enviar deve conter em *attach* um único ficheiro no formato *zip*. O *subject* do *email* deve ser o nome do projecto do Requisito 9.

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

Req. 12 - Data limite de apresentação em época normal O trabalho deve ser entregue no moodle e também por email até às **23:55 de 27 de Junho de 2019**. Não deve assumir que o relógio do sistema está igual a qualquer outro pelo que deve entregar sempre pelo menos 15 min ANTES do tempo limite. A não entrega até à 01:00 de 28 de Junho de 2019 implica zero valores no trabalho.

Req. 13 - Data limite de apresentação em época de recurso O trabalho deve ser entregue no moodle e também por email até às **23:55 de 16 de Julho de 2019**. Não deve assumir que o relógio do sistema está igual a qualquer outro pelo que deve entregar sempre pelo menos 15 min ANTES do tempo limite. A não entrega até à 01:00 de 17 de Julho de 2019 implica zero valores no trabalho.

Finalmente, note que necessita de realizar mais do que o pedido para obter mais de 17 valores. A criatividade também pode justificar uma melhor classificação pelo que extras originais e sofisticados serão sempre uma boa aposta mas os sugeridos são interessantes porque completam

o jogo. 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. 11 da Secção 6) é outra forma de subir a pontuação e contribuir para obter mais do que 17 valores

7 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**. 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. Lembre-se que será avaliado em testes prático em frente a um computador. 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 algum tipo de ocorrência que considere anómala, a defesa do trabalho poderá ser anulada sendo repetida e contabilizada apenas esta segunda defesa.

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