Técnicas de Desenvolvimento de Software

Licenciatura em Engenharia Informática e de Computadores

Semestre de Inverno de 2021/1022

Trabalho prático - fase 1

A avaliação da componente prática da disciplina será realizada com base na aplicação *DesktopChess*, a ser desenvolvida ao longo do semestre. A aplicação viabiliza que dois jogadores, cada um no seu computador, joguem partidas de xadrez. O desenvolvimento da aplicação será realizado em duas fases. O presente documento descreve os requisitos da primeira fase.

Na primeira fase (primeiro trabalho) pretende-se desenvolver uma aplicação em modo consola, onde as jogadas são descritas textualmente e o tabuleiro é também apresentado textualmente, com o aspecto semelhante ao apresentado na figura, em que as peças pretas são representadas por letras minúsculas e as brancas por letras maiúsculas.

```
g1:Black> play h6
    a b c d e f g h
   -----------------
 8 | r n b Q   b n r |
 7 | p p     p p p   |
 6 |         k     p |
 5 |   p             |
 4 |           P     |
 3 |                 |
 2 |   P P     P   P |
 1 | q N B Q   R K   |
   -----------------
g1:White>
```

Na segunda fase (segundo trabalho) será desenvolvida uma aplicação com interface gráfica composta por uma janela que apresenta o tabuleiro e as peças ainda em jogo, e onde as jogadas são realizadas com clicks do rato nas quadrículas do tabuleiro. Os detalhes dessa interface com o utilizador serão descritos no enunciado da segunda fase. Importa sublinhar que se pretende maximizar a reutilização do código implementado na primeira fase, concretamente, o código relativo ao modelo do jogo, ou seja, o estado do tabuleiro, as jogadas já realizadas, e a lógica das regras do jogo, ou seja, tudo aquilo que não seja relativo à interface com o utilizador.

A aplicação em anexo (`Trab1.jar`) é uma demo do pretendido na primeira fase. Esta aplicação usa ficheiros de texto localizados no diretório de trabalho para armazenar o estado corrente dos jogos, em vez de uma base de dados remota, conforme requerido na implementação a ser produzida pelos alunos. Para usar esta demo devem executar duas instâncias usando o mesmo diretório de execução e o mesmo computador.
Para executar a demo, numa janela de comandos use **`kotlin Trab1.jar`** ou **`java -jar Trab1.jar`**

A aplicação da primeira fase suporta os seguintes comandos:

open <gameId>	Abre o jogo com o nome <gameId> para jogar com as peças brancas. Caso o jogo ainda não exista é criado um com esse nome.
join <gameId>	Abre o jogo já existente com o nome <gameId> para jogar com as peças pretas. Reporta erro se o jogo ainda não existe.
play <move>	Realiza a jogada <move> caso seja possível, ou seja, se for a vez do jogador em causa e se a peça indicada pode fazer o movimento descrito.
refresh	Atualiza o estado do jogo depois do outro jogador fazer uma jogada.
moves	Apresenta todas as jogadas realizadas desde o início do jogo.
exit	Termina a aplicação.

Cada comando é lido depois de apresentado o *prompt* no formato: `<gameId>` **:** `<turn>` **>** em que `<gameId>` é o nome do jogo corrente e `<turn>` é a cor das peças a jogar. O tabuleiro do jogo, com o aspecto da figura, é

apresentado novamente sempre que for bem sucedida a execução dos comandos `open`, `join`, `play` e `refresh`.

No comando `play`, `<move>` tem o formato `<piece> <from> <capture> <to> <promotion>` onde:

<piece>	Letra maiúscula (opcional) que identifica a peça (**K**, **Q**, **B**, **N**, **R** ou **P**), sendo P por omissão
<from>	Quadrícula origem; Letra minúscula (**a**..**h**) da coluna (opcional) e dígito (**1**..**8**) da linha (opcional); A coluna e/ou a linha podem ser omitidos caso não exista ambiguidade.
<capture>	Letra **x** em caso de captura; Pode ser omitido caso não exista ambiguidade.
<to>	Quadrícula destino; Letra minúscula (**a**..**h**) da coluna e dígito (**1**..**8**) da linha;
<promotion>	Em caso de promoção o símbolo **=** e letra maiúscula da peça escolhida na promoção do peão;

Para viabilizar que cada jogador faça as jogadas no seu computador, a informação armazenada do estado do jogo é armazenada numa base de dados do tipo Document DB e acessível a partir das máquinas onde o jogo esteja a ser executado. Na base da dados, cada jogada é uma String com o mesmo formato do `<move>` **mas onde não são omitidas as partes opcionais**.

Para distribuir a carga imposta pela realização do trabalho, a primeira fase está dividida em etapas. Na primeira etapa prende-se implementar o tipo **`Board`** que representa o tabuleiro do jogo com as respetivas peças. Este tipo está definido de forma a satisfazer os seguintes testes:

```
@Test
fun `Initial position Board`() {
   val sut = Board()
   assertEquals(
        "rnbqkbnr"+
        "pppppppp"+
        "        ".repeat(4) +
        "PPPPPPPP"+
        "RNBQKBNR", sut.toString() )
}
@Test
fun `MakeMove in Board`() {
   val sut = Board().makeMove("Pe2e4").makeMove("Pe7e5").makeMove("Nb1c3")
   assertEquals(
        "rnbqkbnr"+
        "pppp ppp"+
        "        "+
        "    p   "+
        "    P   "+
        "  N     "+
        "PPPP PPP"+
        "R BQKBNR", sut.toString() )
}
```

As etapas seguintes serão identificadas em breve. =)

Bom trabalho!
Paulo Pereira e Pedro Pereira

ISEL, 19 de Outubro de 2021