Pontifícia Universidade Católica do Rio Grande do Sul – PUCRS
Curso de Ciência da Computação
Programação Distribuída – Turma 128 – Prof. Roland Teodorowitsch – 2018/2

TRABALHO DA ÁREA 1: *KING'S VALLEY* DISTRIBUÍDO EM JAVA RMI

Objetivo

Implementar uma aplicação distribuída em Java RMI (*Remote Method Invocation*) que permita que usuários remotos disputem o jogo *King's Valley* (YAMAMOTO, 2018).

Regras do Jogo

King's Valley é um jogo de estratégia para dois jogadores, inventado em 2006 por Mitsuo Yamamoto (2018).

King's Valley é disputado em um tabuleiro quadrado de 5 casas por 5 casas, cuja posição central é chamada de "*King's Valley*". Cada jogador tem uma peça de rei e 4 peças de soldados. As peças de cada jogador são colocadas alinhadas, com o rei no centro, em uma das extremidades do tabuleiro, conforme mostra a Figura 1. O objetivo do jogo é ser o primeiro jogador a colocar a sua peça de rei no espaço central do tabuleiro.

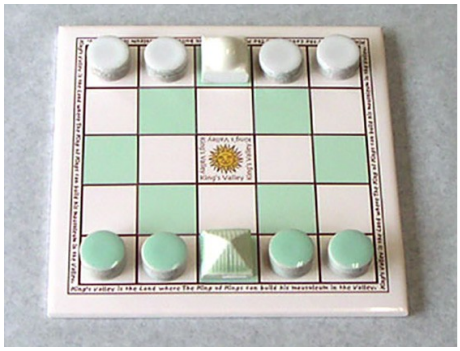

Figura 1 – Posição inicial do Jogo *King's Valley* (YAMAMOTO, 2018)

O jogo consiste em jogadores alternando turnos. Em seu turno um jogador deve mover uma de suas peças (ou o rei ou um de seus soldados) para uma casa vazia do tabuleiro. Todas as peças se movem da mesma forma: define-se o sentido do deslocamento (horizontal, vertical ou diagonal) e deslocam a peça neste sentido até uma casa vazia e enquanto não chegar até o limite do tabuleiro ou até encontrar outra peça, ponto em que o movimento deve se encerrar. O jogador que iniciar a partida deverá obrigatoriamente mover um de seus soldados (apenas neste movimento inicial). Por exemplo, se o jogador que vai iniciar a partida desejar movimentar uma de suas peças no sentido vertical, então ele deverá obrigatoriamente deslocá-la por 3 casas, pois na quarta casa encontrará uma peça do adversário. Parar um movimento sem avançar por todas as casas possíveis no sentido escolhido **não** é possível. Também **não** é possível pular sobre peças.

A única peça que um jogador pode parar na casa central do tabuleiro é o seu rei (o que o torna o vencedor do jogo). Neste movimento final, o rei também não poderá parar se houver casas disponíveis no sentido do seu movimento. Note que soldados podem passar sobre a casa central, mas **não** podem parar sobre ela.

As jogadas são obrigatórias, ou seja, um jogador não pode deixar de mover uma de suas peças em seu turno.

Se um rei ficar encurralado em uma casa, sem poder de mover, isto também determina a derrota do respectivo jogador.

Funcionamento da Aplicação

O servidor deverá funcionar de modo que:

- sejam suportadas 500 partidas simultâneas de *King's Valley* entre 2 jogadores devidamente registrados (ou identificados) no servidor;
- quando um jogador se registra, ele deverá esperar que outro jogador também se registre (quando o próximo jogador se registrar, será formada uma dupla que disputará a próxima partida);
- o primeiro jogador a se registrar inicia jogando;
- responda a invocações remotas de métodos realizadas pelos clientes (por exemplo, conforme a descrição de operações descrita a seguir);
- haja limites de tempo para determinados eventos: 2 minutos (120 segundos) pelo registro do segundo jogador; 60 segundos pelas jogadas de cada jogador; e 60 segundos para "destruir" a partida depois de definido o vencedor.

O cliente será responsável: pela interface com o usuário (que poderá ser tanto em modo texto quanto em modo gráfico); e por executar as invocações remotas de métodos disponíveis no servidor, de modo que os usuários possam jogar partidas consistentes.

Operações

As seguintes operações remotas deverão ser implementadas pelo servidor:

1) **registraJogador**
   Recebe: *string* com o nome do usuário/jogador.
   Retorna: id (valor inteiro) do usuário (que corresponde a um número de identificação único para este usuário durante uma partida), -1 se este usuário já está cadastrado ou -2 se o número máximo de jogadores tiver sido atingido.

2) **encerraPartida**
   Recebe: id do usuário (obtido através da chamada **registraJogador**).
   Retorna: -1 (erro), 0 (ok).

3) **temPartida**
   Recebe: id do usuário (obtido através da chamada **registraJogador**).
   Retorna: -2 (tempo de espera esgotado), -1 (erro), 0 (ainda não há partida), 1 (sim, há partida e o jogador inicia jogando) ou 2 (sim, há partida e o jogador é o segundo a jogar).

4) **obtemOponente**

Recebe: id do usuário (obtido através da chamada **registraJogador**).
Retorna: *string* vazio para erro ou *string* com o nome do oponente.

5) **ehMinhaVez**

Recebe: id do usuário (obtido através da chamada **registraJogador**).
Retorna: -2 (erro: ainda não há 2 jogadores registrados na partida), -1 (erro), 0 (não), 1 (sim), 2 (é o vencedor), 3 (é o perdedor), 4 (houve empate), 5 (vencedor por WO), 6 (perdedor por WO).

6) **obtemTabuleiro**

Recebe: id do usuário (obtido através da chamada **registraJogador**).
Retorna: *string* vazio em caso de erro ou *string* representando o tabuleiro de jogo.
O tabuleiro pode, por exemplo, ser representado por 25 caracteres indicando respectivamente o estado de cada casa do tabuleiro: '`.`' (casa não ocupada), '`C`' (rei da cor clara, pertencente ao primeiro jogador), '`c`' (soldado da cor clara, pertencente ao primeiro jogador), '`E`' (rei da cor escura, pertencente ao segundo jogador), '`e`' (soldado da cor escura, pertencente ao segundo jogador). Recomenda-se representar o tabuleiro como se estivesse disposto "horizontalmente", com as peças do jogador que inicia na coluna da esquerda e as peças do segundo jogador na coluna da direita. Assim, para a posição inicial do seguinte tabuleiro (onde S indica soldado, R indica rei, c indica claro e e indica escuro):

	0	1	2	3	4
0	S_c				S_e
1	S_c				S_e
2	R_c				R_e
3	S_c				S_e
4	S_c				S_e

a cadeia de caracteres retornada por **obtemTabuleiro** poderia ser:

`c...ec...eC...Ec...ec...e`.

8) **movePeca**

Recebe: id do usuário (obtido através da chamada **registraJogador**), número da linha do tabuleiro onde se encontra a peça que se deseja mover (de 0 até 4, inclusive), número da coluna do tabuleiro onde se encontra a peça que se deseja mover (de 0 até 4, inclusive), sentido do deslocamento (0 a 7, inclusive). Para o sentido do deslocamento deve-se usar a seguinte convenção para ambos os jogadores: 0 = para direita; 1 = diagonal direita-inferior; 2 = para baixo; 3 = diagonal esquerda-inferior; 4 = esquerda; 5 = diagonal esquerda-superior; 6 = para cima; 7 = diagonal direita-superior.
Retorna: 2 (partida encerrada, o que ocorrerá caso o jogador demore muito para enviar a sua jogada e ocorra o *time-out* de 60 segundos para envio de jogadas), 1 (tudo certo), 0 (movimento inválido, por exemplo, em um sentido e deslocamento que resulta em uma posição ocupada ou fora do tabuleiro), -1 (jogador não encontrado), -2 (partida não iniciada: ainda não há dois jogadores registrados na partida), -3 (parâmetros de posição e orientação inválidos), -4 (não é a vez do jogador).

Avaliação

Trabalhos com trechos copiados integralmente ou parcialmente serão avaliados com a nota mínima (ZERO). Os demais trabalhos serão avaliados numa escala de 0 (ZERO) até 10 (DEZ), levando em consideração as características descritas neste documento. Serão utilizados os seguintes pesos nesta avaliação:

- 40%: a aplicação executou corretamente sem erros, apresentando o comportamento esperado, conforme as regras do jogo.
- 20%: todas as particularidades (tais como número de partidas e usuários, etc.) definidas neste documento foram implementadas.
- 20%: o aluno usou padrões de programação adequados (programação estruturada, nomes de variáveis significativos, comentários, etc.).
- 10%: usou bloqueios para proteger variáveis compartilhadas contra inconsistências referentes a acesso concorrente;
- 10%: foi implementado um mecanismo de temporização que funciona corretamente?

Entrega

O trabalho deve ser desenvolvido individualmente.

A data de entrega do trabalho é **4 de outubro de 2018**.

Cada aluno deverá entregar todos os arquivos com extensão ".java" necessários para compilar e executar o projeto. E também deverá apresentar a execução de sua aplicação para o professor.

Em caso de cópia de trabalhos serão avaliados com a nota mínima (zero).

REFERÊNCIAS

YAMAMOTO, Mitsuo. **King's Valley**. [s.l]: Logy Games, 17 abr. 2018. Disponível em: <http://www.logygames.com/english/kingsvalley.html>. Acesso em: 25 ago. 2018.