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

TRABALHO 2: *KING'S VALLEY* AUTOMATIZADO EM JAVA WEB SERVICES

Introdução

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.

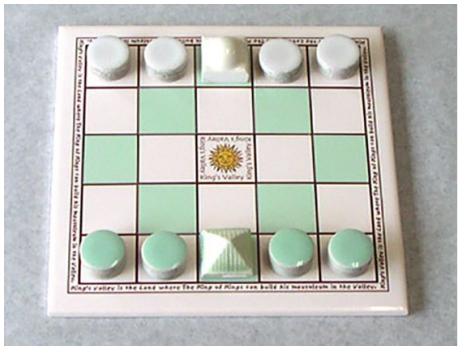

Figura 1 – Posição inicial do Jogo *King's Valley* (YAMAMOTO, 2018)

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.

Objetivos

Os objetivos deste trabalho são:

- exercitar a programação distribuída usando ***Web Services*** na linguagem **Java**;
- desenvolver uma aplicação formada por dois programas, um servidor (executado por um processo) e um cliente (eventualmente executado por vários processos) que interagem entre si para a solução de determinado problema;
- desenvolver uma aplicação servidora capaz de receber acessos concorrentes, gerenciando vários jogos simultaneamente, sem apresentar falhas de consistência.

Definição

Neste trabalho deverá ser implementada uma aplicação distribuída em **Java *Web Services*** (usando SOAP – *Simple Object Access Protocol*) que permita o gerenciamento de várias partidas simultâneas entre 2 jogadores do jogo *King's Valley* (conforme regras definidas na seção Introdução).

A aplicação deverá ser formada por dois programas: um servidor (executado por um processo) e um cliente (eventualmente executado por vários processos).

Quando o servidor for iniciado, ele deverá ser criado de forma que se possa disputar até 500 partidas simultâneas. Este servidor deverá funcionar de forma muito parecida com o servidor implementado como Trabalho 1 desta disciplina. No entanto, antes do registro normal dos jogadores para iniciar uma partida, o servidor deverá aceitar o "pré-registro" dos jogadores, onde informa-se o nome do primeiro jogador, o identificador que este primeiro jogador receberá quando ele fizer o registro, o nome do segundo jogador e o identificador que este segundo jogador receberá quando ele fizer o registro. Esta especificação de nomes e seus respectivos identificadores servirá apenas como ferramenta de teste automatizado do servidor. Se for feito o pré-registro envolvendo 2 jogadores, eles deverão ser associados na mesma partida, mesmo que entre o registro deles ocorra o registro de outros jogadores.

O programa cliente, por sua vez, terá uma estrutura diferente da estrutura do cliente tradicional para execução de uma aplicação interativa (portanto, diferente da estrutura do cliente sugerida para o Trabalho 1 desta disciplina). A nova estrutura do cliente lerá de um arquivo (com a extensão ".in") a especificação de um conjunto de chamadas remotas que devem ser executadas no servidor. E deverá salvar em outro arquivo (com a extensão ".out") o resultado da execução de cada uma destas chamadas.

Para iniciar uma partida de *King's Valley* no servidor remoto, é preciso fazer o registro de dois jogadores, cada um recebendo como resposta um identificador único (que será usado nas demais chamadas). O uso de determinado nome de jogador deve ser feito de forma única, sem permitir dois jogadores com o mesmo nome e reservando-se o direito de uso de determinado nome a quem registrar-se antes no servidor. Com o pré-registro, será possível prever qual o identificador que determinado jogador receberá no seu registro, e assim predefinir uma série de operações.

Especificação de Entradas e Saídas de um Cliente

O programa cliente deverá ser capaz de ler a especificação de um conjunto de chamadas remotas de um arquivo com a extensão ".in", e deverá ser capaz de escrever as respectivas respostas em um arquivo com a extensão ".out". Entradas e saídas serão sempre fornecidas em formato texto (codificação ASCII, sem acentos).

A especificação da entrada começa com o número total de operações remotas que o cliente deve enviar para o servidor. Na sequência aparece uma linha para cada operação remota. Cada uma destas linhas contém o código da operação seguido de um caractere de tabulação e dos parâmetros da operação remota separados por ":".

Devem ser usados os seguintes códigos para as operações:

- Operação 0[1] – **`preRegistro`** (usada para viabilizar o teste)
  Informa ao servidor o nome de um jogador (o primeiro da dupla), o identificador que o servidor deverá utilizar para este primeiro jogador, o nome de outro jogador (o segundo da dupla) e o respectivo identificador que o servidor deverá utilizar para este segundo jogador. Esta operação retorna sempre 0 e não haverá nenhuma inconsistência nas entradas referente às operações de pré-registro (ou seja, elas serão sempre consistentes).
- Operação 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.
- Operação 2 – **`encerraPartida`**
  Recebe: id do usuário (obtido através da chamada **registraJogador**).
  Retorna: -1 (erro), 0 (ok).
- Operação 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

1 Esta operação serve para viabilizar os testes, de forma que as demais operações (que necessitam especificar os identificadores dos jogadores) possam ser previamente especificadas com os identificadores que cada jogador receberá após o registro. Nas entradas de teste, garante-se apenas que os registros dos jogadores de uma dupla (especificados em determinado pré-registro) ocorrerão sempre na mesma ordem deste pré-registro. Após o registro, os dados do respectivo pré-registro não serão mais necessários e podem ser excluídos.

jogador inicia jogando) ou 2 (sim, há partida e o jogador é o segundo a jogar).

- Operação 4 – **`obtemOponente`**
  Recebe: id do usuário (obtido através da chamada **registraJogador**).
  Retorna: *string* vazio para erro ou *string* com o nome do oponente.
- Operação 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).
- Operação 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`.
- Operação 7 – **`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 (não é a vez do jogador), -4 (parâmetros de posição e orientação inválidos), -5 (não é uma casa com peça da cor do jogador).

A sequência a seguir corresponde a um exemplo de entrada que mostra 2 pré-registros sendo feitos. A primeira partida será entre os jogadores “J1” (identificador 1) e “J2” (identificador 2). E a outra será entre os dois jogadores “J3” (identificador 3) e “J4” (identificador 4). Esta segunda partida, no entanto, será encerrada logo em seguida. A primeira partida é desenvolvida até o final, sendo vencida pelo primeiro jogador (“J1”). Ao longo desta partida são feitos alguns testes para verificar se as operações remotas estão gerando os resultados esperados.

Entrada com a especificação das operações		**Resultado esperado para cada operação**
84		
0	J1:1:J2:2	0
0	J3:3:J4:4	0
1	J1	1
3	1	0
5	1	-2
4	1	
1	J3	3
3	3	0
5	3	-2
4	3	
1	J2	2
3	1	1
5	1	1
4	1	J2
3	2	2
5	2	0
4	2	J1
1	J4	4
3	3	1
5	3	1
4	3	J4
3	4	2
5	4	0
4	4	J3
1	J1	-1
2	3	0
2	4	0
5	1	1
6	1	c...ec...eC...Ec...ec...e
7	1:1:0:1	1
6	1	c...e....eC...Ec...ec..ce
5	2	1
6	2	c...e....eC...Ec...ec..ce
7	2:0:3:0	-5
7	2:0:4:0	0
7	2:0:4:1	0
7	2:0:4:2	0
7	2:0:4:5	0
7	2:0:4:6	0
7	2:0:4:7	0
7	2:0:4:8	-4
7	2:0:4:3	1
7	2:3:1:0	-3
6	2	c........eC...Ece..ec..ce
5	1	1
6	1	c........eC...Ece..ec..ce
7	1:4:3:6	1
6	1	c..c.....eC...Ece..ec...e
5	2	1
6	2	c..c.....eC...Ece..ec...e
7	2:3:4:4	1
6	2	c..c.....eC...Ecee..c...e
5	1	1
6	1	c..c.....eC...Ecee..c...e
7	1:4:0:0	1
6	1	c..c.....eC...Ecee.....ce
5	2	1
6	2	c..c.....eC...Ecee.....ce
7	2:4:4:6	1
6	2	c..c.....eC...Ecee.e...c.
5	1	1
6	1	c..c.....eC...Ecee.e...c.
7	1:4:3:6	1
6	1	c..c....ceC...Ecee.e.....
5	2	1
6	2	c..c....ceC...Ecee.e.....
7	2:3:4:4	1
6	2	c..c....ceC...Eceee......
5	1	1
6	1	c..c....ceC...Eceee......
7	1:1:3:2	1
6	1	c..c.....eC..cEceee......
5	2	1
6	2	c..c.....eC..cEceee......

7	2:1:4:4	1
6	2	c..c.e....C..cEceee......
5	1	1
6	1	c..c.e....C..cEceee......
7	1:2:0:0	1
6	1	c..c.e......CcEceee......
5	1	2
5	2	3
2	1	0
2	2	0

Nesta entrada há 84 operações remotas especificadas. Na primeira coluna há o código da operação (conforme a definição de operações citada anteriormente) separado dos parâmetros da operação por um caractere de tabulação. Na coluna da direita é apresentada a saída (resultado gerado pela execução da operação) correspondente à operação da coluna da esquerda.

Quando há mais de um parâmetro para a operação, estes parâmetros estarão separados pelo caractere ":". Inicialmente usa-se a operação de número 0 para pré-registrar os jogadores de duas partidas. A primeira será formada pelos jogadores "J1" (com identificador 1) e "J2" (com identificador 2). Isto significa que eles que receberão respectivamente os identificadores 1 e 2 e **serão alocados na mesma partida** assim que fizerem o seu registro. A segunda operação de pré-registro é para os jogadores "J3" (identificador 3) e "J4" (identificador 4), que serão alocados na mesma partida. Na sequência, aparecem operações de registro (número 1) para "J1", "J3", "J4" e "J2" (nesta ordem), sendo que, mesmo que os registros dos jogadores das partidas estejam intercalados, eles serão corretamente alocados nas partidas, conforme o pré-registro. Nos arquivos com as definições de operações, apenas se garante que o primeiro jodador citado no pré-registro será registrado antes do que o segundo deste mesmo pré-registro. Na sequência, os jogadores de identificadores 3 e 4 executam a operação de número 2 (para abandonar o jogo, ou seja, encerrar sua partida). Seguem-se várias chamadas alternadas que executam uma partida entre os dois jogadores da primeira partida. O jogador "J1" (que inicia jogando) vence a partida contra o jogador "J2". Por fim, ambos os jogadores executam a chamada para encerramento da partida (operação 2).

Avaliação

O programa cliente será usado para submeter vários conjuntos de entradas e suas respectivas operações ao processo servidor. O requisito mínimo para entrega e apresentação corresponde a apresentar corretamente os resultados esperados para o exemplo de entrada apresentado nesta definição. Nos demais casos será avaliado tanto o número de entradas acertadas quanto o nível de erro apresentado.

Outras Especificações:

- O trabalho deverá ser realizado individualmente;
- Não incluir nenhum código-fonte copiado. Em caso de uso de código-fonte não desenvolvido pelos alunos, será atribuída a nota 0 (ZERO) ao trabalho.

Data de Entrega

• 21h15min do dia 22 de novembro de 2018. **Não há opção de entrega com atraso.**

Formato de Entrega

• Entregar os arquivos referentes ao código-fonte. Apresentar e descrever verbalmente o funcionamento da aplicação ao professor.

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.