Pontifícia Universidade Católica do Rio Grande do Sul – PUCRS
Faculdade de Informática
Programação Distribuída – Turma 128 – Prof. Roland Teodorowitsch – 2017/1

TRABALHO DA ÁREA 2: JOGO DOWN UNDER AUTOMATIZADO EM JAVA WEB SERVICES

Introdução

Down Under exige de seus jogadores tenham boa capacidade de planejamento ao desenvolverem mentalmente uma estratégia para obterem, ao final do jogo, o maior número de pontos. E o grande segredo para propiciar este efeito é o tabuleiro mostrado na Figura 1.

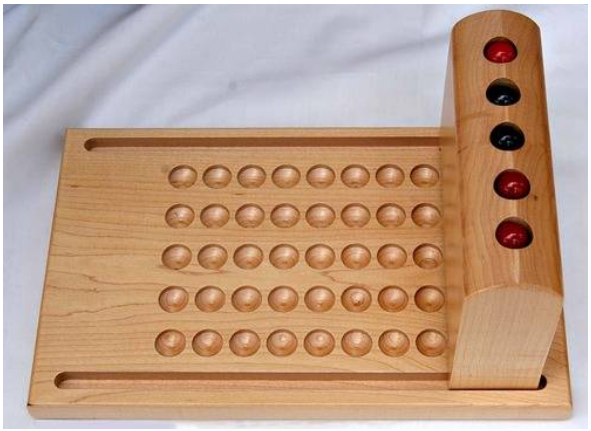

Figura 1 – Tabuleiro de Down Under (DOWN, [s.d.])

O jogo é disputado entre dois jogadores, que no início da partida recebem cada um 20 esferas de determinada cor: 20 esferas escuras para um jogador e 20 esferas claras para o outro jogador. O tabuleiro (Figura 1) é formado por: um espaço horizontal, onde há espaço para 40 esferas (5 x 8 casas), e por uma torre, onde há 5 orifícios alinhados e é possível colocar 8 esferas em cada orifício.

Define-se o jogador que iniciará jogando ou por sorteio ou usando algum outro critério (por exemplo, ordem de registro no servidor). A partir daí os jogadores alternam suas jogadas sempre colocando 2 de suas esferas em quaisquer dos 5 orifícios da torre, desde que ainda haja espaço, é claro. O objetivo é enfileirar o maior número de esferas em sequências horizontais, verticais ou diagonais.

É importante destacar que em relação às torres, cada jogador sabe apenas se um orifício está completamente preenchido ou não. Caso o orifício esteja completamente preenchido (com 8 esferas), é possível ver a cor da última esfera. Caso não esteja completamente preenchido, não é possível ver a cor da última esfera que foi inserida nele, nem saber quantas esferas ele contém.

Depois que o último jogador tiver colocado as suas 2 últimas esferas na torre, ela é movimentada, liberando as esferas, fileira por fileira, nas casas do espaço horizontal do tabuleiro, conforme mostra a Figura 2.

Figura 2 – Liberação das esferas da torre (DOWN, [s.d.])

Neste momento contam-se o número de sequências de 4 esferas e aquele que tiver feito o maior número de sequências vence o jogo. Nesta contagem: por exemplo, 5 esferas em sequência contam como 2 sequências; 6 esferas em sequência contam como 3 sequências; 7 esferas em sequência contam como 4 sequências; e 8 esferas em sequência contam como 5 sequências.

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*** (do tipo SOAP – *Simple Object Access Protocol*) que permita o gerenciamento de várias partidas simultâneas entre 2 jogadores do jogo Donw Under (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 50 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.

O programa cliente, por sua vez, terá uma estrutura diferente da estrutura do cliente tradicional para execução de um jogo Donw Under interativo (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 Donw Under 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 dos parâmetros 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` (relacionada ao jogo propriamente dito): recebe um *string* com o nome do usuário/jogador. Retorna o identificador (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` (relacionada ao jogo propriamente dito): recebe o identificador do usuário (obtido através da chamada `registraJogador`). Retorna: -1 (erro), 0 (ok).
- Operação 3 – `temPartida` (relacionada ao jogo propriamente dito): recebe o identificador 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 com as esferas claras, identificadas, por exemplo, com letras de "C") ou 2 (sim, há partida e o jogador é o segundo a jogar, com os as esferas escuras, identificadas, por exemplo, a letra "E")
- Operação 4 – `ehMinhaVez` (relacionada ao jogo propriamente dito): recebe o identificador 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

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.

(houve empate), 5 (vencedor por WO), 6 (perdedor por WO).

- Operação 5 – `obtemTabuleiro` (relacionada ao jogo propriamente dito): recebe o identificador do usuário (obtido através da chamada `registraJogador`). Retorna: *string* vazio em caso de erro ou *string* representando o tabuleiro de jogo. O tabuleiro deve ser representado por 10 caracteres da seguinte forma:
  "`-CE--^.^..`"
  Os 5 primeiros caracteres indicam o estado dos orifícios da torre: o primeiro orifício ainda pode receber esferas, o segundo e o terceiro estão completos (respectivamente com uma esfera clara e uma esfera escura no topo), e o quarto e o quinto orifício ainda podem receber esferas.
  Os últimos 5 caracteres indicam onde as duas últimas esferas foram soltas (o caractere '^' sinaliza o orifício onde as esferas foram soltas): primeiro e terceiro orifício.
  Depois que todas as esferas de ambos os jogadores tiverem sido jogadas, a torre é deslizada sobre as casas do espaço horizontal e elas são preenchidas revelando todas as esferas jogadas. Neste momento, um jogo em que a torre contém, por exemplo:

```
EECEE
CEECE
CEECC
CCCEE
CCEEE
EECEC
CECCE
CCECC
```

  deve apresentar o resultado com todas as esferas dispostas em uma única linha, com o número de pontos do jogador das esferas claras e com o número de esferas do jogador das esferas escuras separadas por vírgula. Para o exemplo acima, o seguinte *string* deve ser gerado:
  "EECEECEECECEECCCCCEECCEEEEECECCECCECCECC,3,2"
- Operação 6 – `soltaEsfera` (relacionada ao jogo propriamente dito): recebe o identificador do usuário (obtido através da chamada `registraJogador`) e número de identificação do orifício da torre onde se deseja soltar a esfera (de 0 até 4, inclusive). 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 orifício que já tem 8 esferas), -1 (erro: número inválido de orifício), -2 (partida não iniciada: ainda não há dois jogadores registrados na partida), -3 (não é a vez do jogador).
- Operação 7 – `obtemOponente` (relacionada ao jogo propriamente dito): recebe o identificador do usuário (obtido através da chamada `registraJogador`). Retorna um *string* vazio para erro ou um *string* com o nome do oponente.

A seguinte sequência corresponde a um exemplo de entrada que reproduz um jogo em que as esferas claras ("C") conseguem vencer as escuras ("E") por 3 pontos contra 2 pontos.Nesta sequência são especificadas 182 operações remotas.

Entrada com a especificação das operações	**Resultado esperado para cada operação**
182	
0 J1:10:J2:20	0
0 J3:30:J4:40	0
1 J1	10
3 10	0
4 10	-2
7 10	
1 J3	30
1 J4	40
1 J2	20

3	20	2
4	20	0
7	20	J1
3	10	1
4	10	1
7	10	J2
7	30	J4
7	40	J3
2	30	0
2	40	0
5	10	-----.....
4	10	1
6	10:0	1
5	10	-----^....
4	10	1
6	10:1	1
5	10	-----^^...
4	10	0
5	20	-----^^...
4	20	1
6	20:1	1
5	20	-----.^...
4	20	1
6	20:1	1
5	20	-----.^...
4	20	0
5	10	-----.^...
4	10	1
6	10:1	1
5	10	-----.^...
4	10	1
6	10:1	1
5	10	-----.^...
4	10	0
5	20	-----.^...
4	20	1
6	20:1	1
5	20	-----.^...
4	20	1
6	20:1	1
5	20	-----.^...
4	20	0
5	10	-----.^...
4	10	1
6	10:3	1
5	10	-----.^.^.
4	10	1
6	10:3	1
5	10	-----...^.
4	10	0
5	20	-----...^.
4	20	1
6	20:2	1
5	20	-----..^^.
4	20	1
6	20:3	1
5	20	-----..^^.
4	20	0
5	10	-----..^^.
4	10	1
6	10:0	1
5	10	-----^..^.
4	10	1
6	10:4	1
5	10	-----^...^
4	10	0
5	20	-----^...^
4	20	1
6	20:4	1
5	20	-----....^

4	20	1
6	20:1	1
5	20	-E---.^..^
4	20	0
5	10	-E---.^..^
4	10	1
6	10:2	1
5	10	-E---.^^..
4	10	1
6	10:2	1
5	10	-E---..^..
4	10	0
5	20	-E---..^..
4	20	1
6	20:3	1
5	20	-E---..^^.
4	20	1
6	20:3	1
5	20	-E---...^.
4	20	0
5	10	-E---...^.
4	10	1
6	10:3	1
5	10	-E---...^.
4	10	1
6	10:3	1
5	10	-E---...^.
4	10	0
5	20	-E---...^.
4	20	1
6	20:3	1
5	20	-E-E-...^.
4	20	1
6	20:0	1
5	20	-E-E-^..^.
4	20	0
5	10	-E-E-^..^.
4	10	1
6	10:0	1
5	10	-E-E-^....
4	10	1
6	10:4	1
5	10	-E-E-^...^
4	10	0
5	20	-E-E-^...^
4	20	1
6	20:2	1
5	20	-E-E-..^.^
4	20	1
6	20:4	1
5	20	-E-E-..^.^
4	20	0
5	10	-E-E-..^.^
4	10	1
6	10:0	1
5	10	-E-E-^...^
4	10	1
6	10:2	1
5	10	-E-E-^.^..
4	10	0
5	20	-E-E-^.^..
4	20	1
6	20:4	1
5	20	-E-E-..^.^
4	20	1
6	20:2	1
5	20	-E-E-..^.^
4	20	0
5	10	-E-E-..^.^
4	10	1

6	10:0	1
5	10	-E-E-^.^..
4	10	1
6	10:0	1
5	10	-E-E-^....
4	10	0
5	20	-E-E-^....
4	20	1
6	20:0	1
5	20	EE-E-^....
4	20	1
6	20:2	1
5	20	EE-E-^.^..
4	20	0
5	10	EE-E-^.^..
4	10	1
6	10:2	1
5	10	EECE-..^..
4	10	1
6	10:4	1
5	10	EECE-..^.^
4	10	0
5	20	EECE-..^.^
4	20	1
6	20:4	1
5	20	EECE-....^
4	20	1
6	20:4	1
5	20	EECEECEECECEECCCCCEECCEEEEECECCECCECCECC,3,2
4	20	3
4	10	2
2	10	0
2	20	0

Nesta entrada há 182 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 10) e "J2" (com identificador 20). Isto significa que eles que receberão respectivamente os identificadores 10 e 20 e **serão alocados na mesma partida** assim que fizerem o seu registro. A segunda operação de pré-registro é para os jogadores "J3" (identificador 30) e "J4" (identificador 40), 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. Na sequência, os jogadores de identificadores 30 e 40 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. O jogador "J1" (que joga com as esferas claras) vence a partida por 3 pontos contra 2 pontos do jogador "J2" (que joga com as esferas escuras). 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 27 de junho de 2017.

Formato de Entrega

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

Referências

DOWN Under: The Australian Pub Game. [s.l.]: BoardGameGeek, [s.d.]. Disponível em: <https://boardgamegeek.com/boardgame/34074/down-under-australian-pub-game>. Acesso em: 23 mar. 2017.

JOGOS de Todo Mundo. [Florianópolis]: SESC Santa Catarina, [s.d.]. 64 p.