Pontifícia Universidade Católica do Rio Grande do Sul – PUCRS
Faculdade de Informática
Programação Distribuída – Turma 128 – Prof. Roland Teodorowitsch – 2019/1

TRABALHO 2: SPECULATE AUTOMATIZADO EM JAVA WEB SERVICES

Introdução

Speculate é um jogo de dados tradicional baseado em apostas, muito popular em *pubs* e clubes fechados (JOGOS, [s.d.], p. 61).

O jogo consiste de 33 bolas, 1 dado e um tabuleiro (tal como o tabuleiro mostrado na Figura 1). Na parte central do tabuleiro há espaço armazenar bolas e há também casas numeradas de 1 até 5 que podem conter alguma das bolas e uma canaleta numerada com 6, que permite que bolas largadas nela caiam na parte central do tabuleiro (juntamente com as demais bolas).

Figura 1 – Tabuleiro do jogo Speculate
(http://gameanalyticz.blogspot.com.br/2008_10_01_archive.html)

Uma partida de Speculate pode ser disputada por dois ou mais jogadores, mas para simplificar o desenvolvimento será considerado que a partida ocorrerá sempre entre 2 jogadores. Inicialmente 3 bolas são colocadas nas casas 1, 3 e 5 do tabuleiro. As demais 30 bolas são divididas entre os participantes (15 bolas para cada jogador).

O primeiro jogador a se habilitar (registrar) para a partida, inicia jogando. E o objetivo dos jogadores será eliminar as bolas que cada um tem nas mãos. Antes de realizar suas jogadas, o jogador deve dizer quantas vezes quer lançar o dado. O número de lançamentos deve variar de 1 até o número de bolas que este jogador tem em suas mãos. Para cada lançamento de dado, o número obtido indica em que casa a bola deve ser colocada. Se a casa estiver livre, o jogador coloca uma de suas bolas ali. Se a casa estiver ocupada, o jogador deve pegar esta bola e colocá-la junto com as bolas que tem em suas mãos, deixando a respectiva casa livre. Se o jogador tiver obtido o número 6, ele coloca uma de suas bolas na respectiva cavidade e ela rola definitivamente para o centro do tabuleiro.

Depois de executar todos os seus lançamentos, o jogador passa a vez para o adversário, que procede exatamente da mesma forma (definindo o número de lançamentos e executando estes lançamentos).

O vencedor será o primeiro jogador a ficar sem nenhuma bola.

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 Speculate (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 Speculate 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: código de sucesso (0 indica sucesso e -1, erro).
  Observação: caso um dos jogadores chame **encerraPartida** antes de se determinar um vencedor para a partida, o outro jogador será vencedor por WO (ou seja, receberá o código 5 quando chamar **ehMinhaVez**).
- 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 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.

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.

- 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: jogador não encontrado), 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 – **obtemNumBolas**
  Recebe: id do usuário (obtido através da chamada **registraJogador**).
  Retorna: número de bolas que o jogador ainda tem em suas mãos, -2 (erro: ainda não há 2 jogadores registrados na partida), -1 (erro: jogador não encontrado).
- Operação 7 – **obtemNumBolasOponente**
  Recebe: id do usuário (obtido através da chamada **registraJogador**).
  Retorna: número de bolas que o oponente ainda tem em suas mãos, -2 (erro: ainda não há 2 jogadores registrados na partida), -1 (erro: jogador não encontrado).
- Operação 8 – **obtemTabuleiro**
  Recebe: id do usuário (obtido através da chamada **registraJogador**).
  Retorna: *string* vazio em caso de erro ou *string* com o tabuleiro de jogo.
  Observação: essa *string* é uma representação do tabuleiro que possui 6 caracteres, respectivamente correspondentes ao estado de cada uma das 6 casas do tabuleiro. Se o caractere corresponder a um "*", isto significa que a respectiva casa está ocupada por uma bola. Se a casa estiver desocupada, o caractere será o próprio valor do dado que deve ser tirado para colocar uma bola nesta casa. A casa 6, por exemplo, nunca conterá um "*". Como no início do jogo há bolas nas casas 1, 3 e 5, o *string* retornado no início do jogo deverá ser "*2*4*6".
- Operação 9 – **defineJogadas**
  Recebe: id do usuário (obtido através da chamada **registraJogador**), número de lançamentos que o jogador realizará.
  Retorna: 1 (tudo certo), ou -1 (erro), -2 (erro: ainda não há partida), -3 (não é a vez do jogador), -4 (é a vez do jogador, mas não para definir o número de lançamentos), -5 (o número de jogadas é inválido, por exemplo, maior do que o número de bolas que o jogador tem em mãos).
- Operação 10 – **jogaDado**
  Recebe: id do usuário (obtido através da chamada **registraJogador**).
  Retorna: número obtido no dado, ou -1 (erro), -2 (erro: ainda não há partida), -3 (não é a vez do jogador), -4 (é a vez do jogador, mas não para jogar dados).

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
524		
0	J1:1:J2:2	0
0	J3:3:J4:4	0
1	J1	1
3	1	0
5	1	-2
4	1	
9	1:10	-2
1	J3	3
3	3	0
5	3	-2
4	3	
1	J2	2
3	2	2
5	2	0
4	2	J1
9	2:10	-3
1	J1	-1
3	1	1
5	1	1
4	1	J2
1	J4	4
3	4	2
5	4	0
4	4	J3
3	3	1
5	3	1
4	3	J4
2	3	0
2	4	0
3	1	1
5	1	1
4	1	J2
8	1	*2*4*6
6	1	15
7	1	15
8	2	*2*4*6
6	2	15
7	2	15
9	2:15	-3
9	1:16	-5
10	2	-3
6	1	15
9	1:15	1
10	1	3
8	1	*234*6
6	1	16
10	1	3
8	1	*2*4*6
6	1	15
10	1	1
8	1	12*4*6
6	1	16
10	1	2
8	1	1**4*6
6	1	15
10	1	1
8	1	***4*6
6	1	14
10	1	1
8	1	1**4*6
6	1	15
10	1	4
8	1	1****6
6	1	14
10	1	5
8	1	1***56
6	1	15
10	1	2
8	1	12**56
6	1	16
10	1	2
8	1	1***56
6	1	15
10	1	4
8	1	1**456
6	1	16
10	1	5
8	1	1**4*6
6	1	15
10	1	6
8	1	1**4*6
6	1	14
10	1	5
8	1	1**456
6	1	15
10	1	4
8	1	1***56
6	1	14
6	2	15
9	2:15	1

Entrada com a especificação das operações		Resultado esperado para cada operação
10	2	5
8	2	1****6
6	2	14
10	2	1
8	2	*****6
6	2	13
10	2	4
8	2	***4*6
6	2	14
10	2	5
8	2	***456
6	2	15
10	2	2
8	2	*2*456
6	2	16
10	2	2
8	2	***456
6	2	15
10	2	1
8	2	1**456
6	2	16
10	2	2
8	2	12*456
6	2	17
10	2	6
8	2	12*456
6	2	16
10	2	5
8	2	12*4*6
6	2	15
10	2	2
8	2	1**4*6
6	2	14
10	2	6
8	2	1**4*6
6	2	13
10	2	6
8	2	1**4*6
6	2	12
10	2	3
8	2	1*34*6
6	2	13
10	2	6
8	2	1*34*6
6	2	12
6	1	14
9	1:14	1
10	1	3
8	1	1**4*6
6	1	13
10	1	6
8	1	1**4*6
6	1	12
10	1	1
8	1	***4*6
6	1	11
10	1	1
8	1	1**4*6
6	1	12
10	1	3
8	1	1*34*6
6	1	13
10	1	4
8	1	1*3**6
6	1	12
10	1	4
8	1	1*34*6
6	1	13
10	1	1
8	1	**34*6
6	1	12
10	1	4
8	1	**3**6
6	1	11
10	1	2
8	1	*23**6
6	1	12
10	1	2
8	1	**3**6
6	1	11
10	1	6
8	1	**3**6
6	1	10
10	1	3
8	1	*****6
6	1	9
10	1	5
8	1	****56
6	1	10
6	2	12
9	2:12	1
10	2	3

Entrada com a especificação das operações		Resultado esperado para cada operação
8	2	**3*56
6	2	13
10	2	5
8	2	**3**6
6	2	12
10	2	3
8	2	*****6
6	2	11
10	2	6
8	2	*****6
6	2	10
10	2	5
8	2	****56
6	2	11
10	2	3
8	2	**3*56
6	2	12
10	2	4
8	2	**3456
6	2	13
10	2	4
8	2	**3*56
6	2	12
10	2	4
8	2	**3456
6	2	13
10	2	2
8	2	*23456
6	2	14
10	2	4
8	2	*23*56
6	2	13
10	2	5
8	2	*23**6
6	2	12
6	1	10
9	1:10	1
10	1	4
8	1	*234*6
6	1	11
10	1	5
8	1	*23456
6	1	12
10	1	1
8	1	123456
6	1	13
10	1	1
8	1	*23456
6	1	12
10	1	3
8	1	*2*456
6	1	11
10	1	2
8	1	***456
6	1	10
10	1	2
8	1	*2*456
6	1	11
10	1	3
8	1	*23456
6	1	12
10	1	2
8	1	**3456
6	1	11
10	1	4
8	1	**3*56
6	1	10
6	2	12
9	2:12	1
10	2	2
8	2	*23*56
6	2	13
10	2	1
8	2	123*56
6	2	14
10	2	3
8	2	12**56
6	2	13
10	2	6
8	2	12**56
6	2	12
10	2	6
8	2	12**56
6	2	11
10	2	3
8	2	123*56
6	2	12
10	2	5
8	2	123**6
6	2	11
10	2	4
8	2	1234*6

Entrada com a especificação das operações		Resultado esperado para cada operação
6	2	12
10	2	5
8	2	123456
6	2	13
10	2	3
8	2	12*456
6	2	12
10	2	6
8	2	12*456
6	2	11
10	2	6
8	2	12*456
6	2	10
6	1	10
9	1:10	1
10	1	3
8	1	123456
6	1	11
10	1	2
8	1	1*3456
6	1	10
10	1	6
8	1	1*3456
6	1	9
10	1	6
8	1	1*3456
6	1	8
10	1	5
8	1	1*34*6
6	1	7
10	1	4
8	1	1*3**6
6	1	6
10	1	5
8	1	1*3*56
6	1	7
10	1	3
8	1	1***56
6	1	6
10	1	3
8	1	1*3*56
6	1	7
10	1	3
8	1	1***56
6	1	6
6	2	10
9	2:10	1
10	2	3
8	2	1*3*56
6	2	11
10	2	2
8	2	123*56
6	2	12
10	2	1
8	2	*23*56
6	2	11
10	2	5
8	2	*23**6
6	2	10
10	2	6
8	2	*23**6
6	2	9
10	2	5
8	2	*23*56
6	2	10
10	2	6
8	2	*23*56
6	2	9
10	2	6
8	2	*23*56
6	2	8
10	2	1
8	2	123*56
6	2	9
10	2	6
8	2	123*56
6	2	8
6	1	6
9	1:6	1
10	1	4
8	1	123456
6	1	7
10	1	5
8	1	1234*6
6	1	6
10	1	6
8	1	1234*6
6	1	5
10	1	5
8	1	123456
6	1	6
10	1	3

Entrada com a especificação das operações		Resultado esperado para cada operação
8	1	12*456
6	1	5
10	1	6
8	1	12*456
6	1	4
6	2	8
9	2:8	1
10	2	6
8	2	12*456
6	2	7
10	2	6
8	2	12*456
6	2	6
10	2	2
8	2	1**456
6	2	5
10	2	2
8	2	12*456
6	2	6
10	2	4
8	2	12**56
6	2	5
10	2	6
8	2	12**56
6	2	4
10	2	2
8	2	1***56
6	2	3
10	2	3
8	2	1*3*56
6	2	4
6	1	4
9	1:4	1
10	1	5
8	1	1*3**6
6	1	3
10	1	1
8	1	**3**6
6	1	2
10	1	4
8	1	**34*6
6	1	3
10	1	4
8	1	**3**6
6	1	2
6	2	4
9	2:4	1
10	2	1
8	2	1*3**6
6	2	5
10	2	3
8	2	1****6
6	2	4
10	2	4
8	2	1**4*6
6	2	5
10	2	6
8	2	1**4*6
6	2	4
6	1	2
9	1:2	1
10	1	1
8	1	***4*6
6	1	1
10	1	1
8	1	1**4*6
6	1	2
6	2	4
9	2:4	1
10	2	2
8	2	12*4*6
6	2	5
10	2	3
8	2	1234*6
6	2	6
10	2	2
8	2	1*34*6
6	2	5
10	2	1
8	2	**34*6
6	2	4
6	1	2
9	1:2	1
10	1	5
8	1	**3456
6	1	3
10	1	1
8	1	1*3456
6	1	4
6	2	4
9	2:4	1
10	2	6

Entrada com a especificação das operações		Resultado esperado para cada operação
8	2	1*3456
6	2	3
10	2	3
8	2	1**456
6	2	2
10	2	5
8	2	1**4*6
6	2	1
10	2	5
8	2	1**456
6	2	2
6	1	4
9	1:4	1
10	1	2
8	1	12*456
6	1	5
10	1	5
8	1	12*4*6
6	1	4
10	1	1
8	1	*2*4*6
6	1	3
10	1	3
8	1	*234*6
6	1	4
6	2	2
9	2:2	1
10	2	1
8	2	1234*6
6	2	3
10	2	3
8	2	12*4*6
6	2	2
6	1	4
9	1:4	1
10	1	1
8	1	*2*4*6
6	1	3
10	1	3
8	1	*234*6
6	1	4
10	1	6
8	1	*234*6
6	1	3
10	1	2
8	1	**34*6
6	1	2
6	2	2
9	2:2	1
10	2	3
8	2	***4*6
6	2	1
10	2	5
8	2	***456
6	2	2
6	1	2
9	1:2	1
10	1	6
8	1	***456
6	1	1
10	1	6
8	1	***456
6	1	0
5	1	2
2	1	0
2	2	0

Nesta entrada há 524 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

basicamente (entre outras chamadas de testes) operações de registro (número 1) para “J1”, “J3”, “J2” e “J4” (nesta ordem), sendo que, mesmo que os registros dos jogadores das partidas estejam intercalados, eles serão corretamente alocados nas partidas, conforme especificado nos pré-registros. 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).

Geração de Valores Aleatórios para os Dados

Ao longo de um jogo de Speculate, será necessário realilzar o lançamento dados. No entanto, como as partidas devem apresentar resultados previsíveis, para que a correção do código possa ser testada através dos resultados gerados, será necessário gerar estes valores seguindo o padrão descrito a seguir.

Desta forma, para viabilizar a realização dos testes, será preciso criar **para cada partida** um gerador de números aleatórios com semente definida a partir da soma dos identificadores dos dois jogadores envolvidos na partida (`id1` e `id2`):

```
// Inicializacao do gerador de numeros aleatorios
Random gerador = new Random(id1+id2);
```

Os valores obtidos a partir deste gerador serão usados exclusivamente na simulação dos lançamentos de dados. Deve-se usar, por exemplo:

```
int valorDado() {
  return gerador.nextInt(6)+1;
}
```

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 25 de junho de 2019. **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

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