PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
ESCOLA POLITÉCNICA
BACHARELADO EM ENGENHARIA DE SOFTWARE
DISCIPLINA DE LABORATÓRIO DE REDES
PROF. CRISTINA MOREIRA NUNES

Jogo da Velha

Trabalho 1 da Disciplina

Rafael Barni Ritter
rafael.barni@acad.pucrs.br

Gabriel Ferreira Kurtz
gabriel.kurtz@acad.pucrs.br

Porto Alegre, 24 de Setembro de 2018

1. Introdução

Este projeto foi realizado para o primeiro trabalho da disciplina de Laboratório de Redes. Foi desenvolvido um software de Jogo da Velha na linguagem Java, com interface de texto, no modelo Cliente/Servidor, onde é possível que dois clientes diferentes realizem uma partida remotamente conectando ao servidor.

2. Conexão

Para a conexão, utilizamos o Socket do próprio Java, mais especificamente as classes OutputStream e InputStream. Tanto o cliente quanto o servidor, ao conectarem o Socket padrão (classe Socket), fazem o upgrade para um OutputStream que permite o envio de dados e um InputStream que recebe os dados, sendo que para transformar o InputStream

em Strings formatadas foi utilizado um BufferedReader posteriormente. Foi utilizado o protocolo TCP.

3. Model

As classes do Model da arquitetura foram bastante simples. Optamos por uma classe Jogador, que representa um dos clientes, com dois atributos, nome e elemento. Estes campos não são configuráveis: o primeiro jogador a entrar na partida (aquele que cria a sala) é sempre o jogador Xis e o segundo jogador (que entra na sala) é o jogador Bolinha. O campo nome exibe esta informação no formato String, enquanto o campo elemento exibe no formato Elemento.

A segunda classe do model, Elemento, portanto, é um enum que permite exibir apenas três informações:

XIS("X"),

BOLINHA("O"),

VAZIO("_");

E é utilizada tanto para demonstrar qual usuário está utilizando cada sinal como para preencher os campos do espaço de jogo com quem marcou cada campo ou então se o campo ainda está vazio.

4. Cliente

O cliente instancia um BufferedReader para receber o input do usuário na interface, e inicia solicitando a porta da conexão. É possível alterar também o endereço da conexão facilmente modificando a constante "*private static final String HOST = "127.0.0.1";*" (a primeira constante do código).

Possuindo estes dados, é chamada a função *menu()*, onde é solicitada a conexão do socket neste endereço e feito o upgrade para InputStream e OutputStream. A interface pergunta ao usuário então se ele deseja criar uma nova sala ou entrar em uma sala existente.

Caso deseje criar uma nova sala, o usuário deve entrar com o nome da sala no input. Este nome é enviado ao servidor para verificar se está disponível e, caso esteja, recebe a resposta que a sala foi criada. Caso contrário, deve tentar novamente.

Com a sala criada, é necessário esperar que outro jogador conecte através de um novo cliente. Este jogador irá selecionar a opção de entrar em uma sala existente e informar o mesmo nome, e assim o jogo inicia para ambos.

Uma string representando o tabuleiro do Jogo da Velha será impressa para ambos os jogadores, sendo que o jogador Xis vai jogar primeiro enquanto ao jogador Bolinha é solicitado que aguarde a jogada do oponente. Durante esta espera, o Socket fica aguardando a informação do servidor que é a vez daquele jogador, e quando o cliente recebe a mensagem informando a última jogada, troca para a vez deste e imprime novamente o tabuleiro com a última jogada do oponente.

Cada uma das jogadas gera uma mensagem a ser enviada ao servidor informando a jogada a ser feita. Ambos usuários vão jogando alternadamente até que um deles vença ou o jogo termine.

Ao final da partida, é perguntado se desejam jogar novamente. Caso deseje, repete-se todo o processo de criar a sala ou entrar nela e jogar uma partida. Caso contrário, o cliente encerra.

5.1. Servidor/Conexão

O servidor inicia também solicitando a entrada da porta e, em seguida, cria um ServerSocket e roda a função *tratamentoPrincipal().* Nesta função é criado um HashMap *salas* que mapeia um nome de sala ao Jogador Xis que a criou, através da função *ServidorTCP()*.

Posteriormente, é realizado um laço de escuta, que aguarda algum cliente entrar em contato. Ao receber contato do cliente, é criada uma conexão para ele em um Thread paralelo, e são realizados os upgrades do socket para OutputStream e InputStream (o input formatado por BufferedReader) da mesma forma que o cliente.

Como foi explicado anteriormente, o cliente possui duas opções neste estágio: criar uma sala ou entrar em uma sala existente. O servidor espera receber esta informação no

início da primeira mensagem recebida, e possui uma rotina de tratamento diferente para cada caso.

No caso de criar a sala, deve checar se o nome está disponível e, se estiver, adicionar a nova sala ao mapa *salas,* retornando ao cliente a mensagem de que a sala foi criada.

No caso de entrar em uma sala existente, verifica se a sala existe e não está cheia e então coloca o segundo jogador na sala. Ao conseguir unir dois jogadores na mesma sala, é criado um novo Thread partida e então solicita-se o início deste novo Thread, de modo que a partida será executada em paralelo com a chamada da função *run().*

5.2. Servidor/Partida

Ao executar a partida em paralelo, ocorrem novamente os upgrades dos sockets (são passados os sockets padrão como argumento), desta vez dois InputStreams e dois OutputStreams, pois é um para cada jogador (Xis e Bolinha). É instanciado também um objeto da classe Partida() que explicaremos posteriormente, que é a estrutura onde ficam gravados os dados de cada partida.

A cada usuário a quem cabe jogar, é enviada consecutivamente uma mensagem com o estado atual do tabuleiro e solicitado que selecione a jogada, sendo que as nove casas do tabuleiro são numeradas, e o número da casa que ele pretende marcar é recebido como resposta.

Se a jogada for válida, grava-se ela no objeto Partida e o turno passa para o jogador seguinte. O laço de jogo encerra quando houver algum vencedor pelas regras do Jogo da Velha ou então se ninguém vencer ao serem preenchidos os 9 campos (“deu velha”).

Ao serem emitidos os resultados, encerra a execução paralela da partida e prossegue o laço principal *tratamentoPrincipal(),* do momento em que é solicitado ao usuário se deseja continuar jogando.

6. Dados da Partida

Os dados da partida são gravados na classe Partida(), que possui como atributos:

- tabuleiro: Elemento[3][3] representando um tabuleiro de Jogo da Velha onde cada campo guarda um objeto da classe Elemento(), descrita anteriormente.

- jogadorXis e jogadorBolinha: Objetos da classe Jogador() representando ambos os jogadores.
- vez: Elemento que indica de quem é a vez de jogar.

Além de uma constante ENUMERACAO_DO_TABULEIRO que é a legenda que indica qual o número que corresponde a cada casinha do tabuleiro.

O método *jogar()* permite realizar uma jogada, recebendo uma String que determina a próxima jogada e fazendo as alterações correspondentes na estrutura de dados.

Após cada jogada, é necessário verificar se o jogo terminou ou não. Para isso, o método *venceu()* verifica se há algum vencedor e o método *velha()* verifica se deu velha.

O método *toString()* permite representar o estado atual do tabuleiro em formato String, para ser exibido junto com a legenda a cada jogador que possui a vez de jogar.

O servidor atua com esta estrutura de dados para gravar no tempo de execução as opções de cada jogador e, assim, executar uma partida.

7. Conclusão

Com estas informações, é possível compreender o funcionamento do nosso programa. Acreditamos que ele representa todas as regras do Jogo da Velha e que as exigências de conectividade são adequadas para a proposta. O código também está amplamente comentado, e é de fácil interpretação caso seja necessária uma compreensão mais específica de algum aspecto.