UNIVERSIDADE DE TRÁS-OS-MONTES E ALTO DOURO

JOGO DO GALO

Usando Java RMI

CURSO
Redes de Computadores

CADEIRA
Redes de Computadores

SÍNTESE
Breve descrição (parágrafo) com a descrição de alto nível do jogo/projeto

AUTORES
Eduardo Queirós nº22653
João Campeã nº20114

VERSÃO
Versão RC01

DATA
25 de junho de 2020

Índice

Resumo do Jogo

História e conceito do jogo

O jogo do galo ou Tic Tac Toe, é um jogo e um passatempo popular.

É um jogo de regras extremamente simples, que não traz grandes dificuldades para seus jogadores e é facilmente aprendido.

A origem pode ter começado no antigo Egito, onde foram encontrados tabuleiros esculpidos na rocha, que teriam mais de 3.500 anos.

O jogo poderá terá nascido em Portugal, na cidade de Almada no ano 545, no entanto, só foi popularizado no ano 1500, pelo descobridor Pedro Álvares Cabral, que adorava jogar este jogo durante as suas viagens.

Álvares Cabral terá decidido que este jogo seria o primeiro a ser ensinado ao povo indígena no Brasil.

Analisando o número de possibilidades de forma simplista, existem 362.880 maneiras de se dispor a cruz e o círculo no tabuleiro, sem considerar jogadas vencedoras.

Quando consideramos as combinações vencedoras, existem 255.168 jogos possíveis

Conceito do jogo

O conceito do jogo não se baseia na sua jogabilidade como jogo de entretimento, mas sim no estudo realizado numa tecnologia que suporta a plataforma Java RMI facilitando o desenvolvimento de aplicações distribuídas.

Tema principal

O seu tema principal assenta num único jogo de tabuleiro disputado entre apenas dois jogadores, cada um com o seu único “user name”, tentando no menor tempo possível ganhar segundo as regras do jogo mais abaixo mencionadas.

Pode resultar vitória ou empate.

Características principais

Este jogo tem como característica principal a sua jogabilidade exclusiva num ambiente on-line, podendo haver mais que um jogo em simultâneo.

Dessa forma, permitir-se-á que haja um cliente remoto (possivelmente corre em máquina diferente que a máquina que serve o jogo) que interage com outro jogador e que invoca as funções do servidor via Java RMI (Remote Method Invocation).

Potencial audiência

Este nosso jogo tem como destino principal o estudante ou académico e não o tradicional jogador comercial, dirigido para conhecedores que queiram evoluir e/ou iniciantes que queiram aprender Java RMI, utilizando-o como base para o seu estudo.

Aspetos visuais

Como o jogo não se baseia no gameplay, é usado uma interface básica e simples, CLI (command-line interface), para interagir entre o utilizador e o programa, necessita apenas, de emitir comandos sob a forma de sucessivas linhas de texto.

Esta interface exige menos código, menos recursos da máquina e é mais rápido que qualquer outra interface gráfica.

Não é imune, mas evita muitos bug’s (erros), principalmente a trabalhar em java RMI.

Breve sumário do gameplay

Os jogadores inscrevem-se no jogo, inserindo o seu “user name” único, ficando a aguardar jogo ou entrar diretamente num jogo já iniciado.

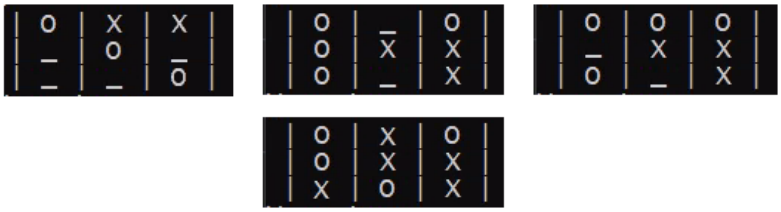
Começam a jogada de uma forma clássica, um jogador coloca o “X”, o outro “O”, sendo impresso o estado do tabuleiro em cada alternância da jogada.

Esse processo continua até que um jogador ganhe ou o tabuleiro fique cheio (indicando que ocorreu um empate).

Para especificar a jogada é usado uma marca no formato de dois números inteiros, que especificam a linha e a coluna, posição essa no tabuleiro.

Exemplo de várias jogadas possíveis de finalizar o jogo:

Jogador 'O', ganha o jogo com linha diagonal, vertical, horizontal ou simplesmente empata a partida, respetivamente



Desafios e ações (gameplay)

Resumo do progresso no jogo

No gameplay, a parte do cliente irá ter um papel mais preponderante que a parte do servidor, este apenas fica ligado e disponível para ser acedido on-line.

Seguido da disponibilidade do servidor, o player regista-se com o seu “user name”.

Depois de estar registado na plataforma online e caso pretenda jogar, faz uma procura por um jogo já iniciado.

Se existir uma partida disponível, entra na partida, caso contrário, cria uma partida nova, aguardando outro player que pretenda jogar, fazendo também previamente o seu registo.

O jogo acaba com 2 resultados possíveis, vitoria ou empate.

Nessa partida única vai desenvolver o tipo de jogada alternada, em que um jogador coloca o “X”, o outro “O”, indicando cada um, na sua vez, a linha e a coluna onde vai colocar o seu ícone, acabando quem fizer primeiro o 3 em linha.

É definido um método, que será chamado após cada movimento para atualizar a flag (estado), verificando o estado do tabuleiro de jogo.

O jogo é programado como uma máquina de estado. Dependendo do estado atual e da jogada do jogador, o jogo entra no próximo estado.

Missão e desafios e objetivos

A nossa missão não é o jogo em si, mas sim o trabalho em Java RMI, desenvolvendo um jogo do galo “distribuído”, possibilitando um jogador disputar um jogo com um adversário, no mesmo dispositivo ou não, como “cliente e servidor”.

Esta linguagem permite ao programador invocar métodos de objetos remotos, que estão alojados em máquinas virtuais Java distintas (JVM), duma forma muito semelhante às invocações a objetos locais.

Esta linguagem é organizada em métodos, que acedem a variáveis globais comuns. (Os programas são organizados em classes.)

A nossa comunicação entre **interface** e a **implementação** é assegurada pelo RMI utilizando o protocolo TCP/IP, sendo a camada de transporte responsável pela validação da porta, no nosso caso 1099 (podia ser outra porta).

Basicamente o nosso trabalho começou por:

a) Definimos a nossa **interface** remota (*ServerRMI*), onde o objeto remoto é uma instância de uma classe que implementa uma interface remota, declarando um conjunto de métodos remotos.
Cada método remoto declara no inicio "java.rmi.RemoteException" entre outros.
b) **Implementamos** o nosso servidor (*ServerGame*) que é uma classe "servidor", neste contexto, é a classe que possui um método que cria uma instância da implementação do objeto remoto, exporta o objeto remoto e vincula essa instância a um nome num registo Java RMI.
A classe que contém esse método pode ser a própria classe de implementação ou outra classe.
c) **Implementamos** o cliente em que programa cliente (App) obtém um plano do registo no host do servidor, procura esse plano do objeto remoto por nome no registo e chama o método no objeto remoto usando esse plano.
d) Compilamos os arquivos de origem (pode ser feito através da linha de comando," java -jar server.jar"
e) - Iniciamos o registo, servidor e cliente Java RMI (quando o servidor estiver pronto, o cliente pode ser iniciado).

Basicamente e resumindo, começamos por implementar um servidor numa plataforma on-line, disponibilizando esse local/server, instanciando as classes subjacentes no código para executar as várias tarefas pretendidas para o funcionamento do nosso jogo anteriormente descrito.
Quando um cliente fizer login, e houver um player para jogar, terá de procurar no servidor pelo nome que lhe foi atribuído, que neste caso é "Game"
O objetivo do jogo do galo é o primeiro jogador a construir uma linha com 3 peças iguais.

Regras

É sorteado aleatoriamente quem é o primeiro jogador a jogar.
Os dois jogadores colocam, alternadamente, as suas peças de forma a construir uma linha com 3 peças iguais em tabuleiros 3×3.
Ganha quem conseguir fazer primeiro o "3 em linha" e caso ninguém consiga e o quadro fique cheio, devolve resultado como empate.
A linha de peças iguais, pode ser construída na vertical, na horizontal ou na oblíqua.

Resumo das opções do jogo (menu principal de opções)

1 - Join in a party

Envia um pedido ao servidor para encontrar uma partida disponível, se não existir, cria uma partida e adiciona essa partida à lista de partidas.

2 - Show Players Online

Manda um pedido ao servidor para mostrar a lista dos players registados.

0 - Leave the game

Manda um pedido ao servidor para remover o player da lista de players online e de seguida termina a aplicação localmente.

Ações/implementações de jogo

Descrição das ações/implementações mais importantes

Implementação do nosso servidor (ServerRMI):

- Criar e registar o servidor "`Registry registry = LocateRegistry.createRegistry(Registry.REGISTRY_PORT)`"

- Criar e exportar um objeto "UnicastRemoteObjet" usando uma porta ex 1099. "GameInterface gameInterface = (GameInterface) UnicastRemoteObject.*exportObject*(serverGame, 0)"
- Fica em espera de um cliente

registar o player online (Conect):

- Cliente conecta com o servidor
- Servidor retorna o registo do player pelo método "`ConnectException" (this.registry = LocateRegistry.getRegistry"` de seguida pesquisa na web pelo nome do servidor(Game) que pretendemos `this.game=(common.GameInterface)registry.lookup("game")`.
- Caso não se consiga estabelecer conexão, devolve msg de erro ""`Problems connecting to the server`".

Visualizar lista de players online (Menu 2-List all players online):

- cliente faz um pedido ao servidor pelo método "`game.getPlayers()`", o getplayers retorna uma lista de dados por " `ArrayList<Player> getPlayers()`" e depois apresenta dos dados dessa lista "`System.out.println(p.toBeautifyString`"

Player entrar no jogo (Menu 1-Join a match):

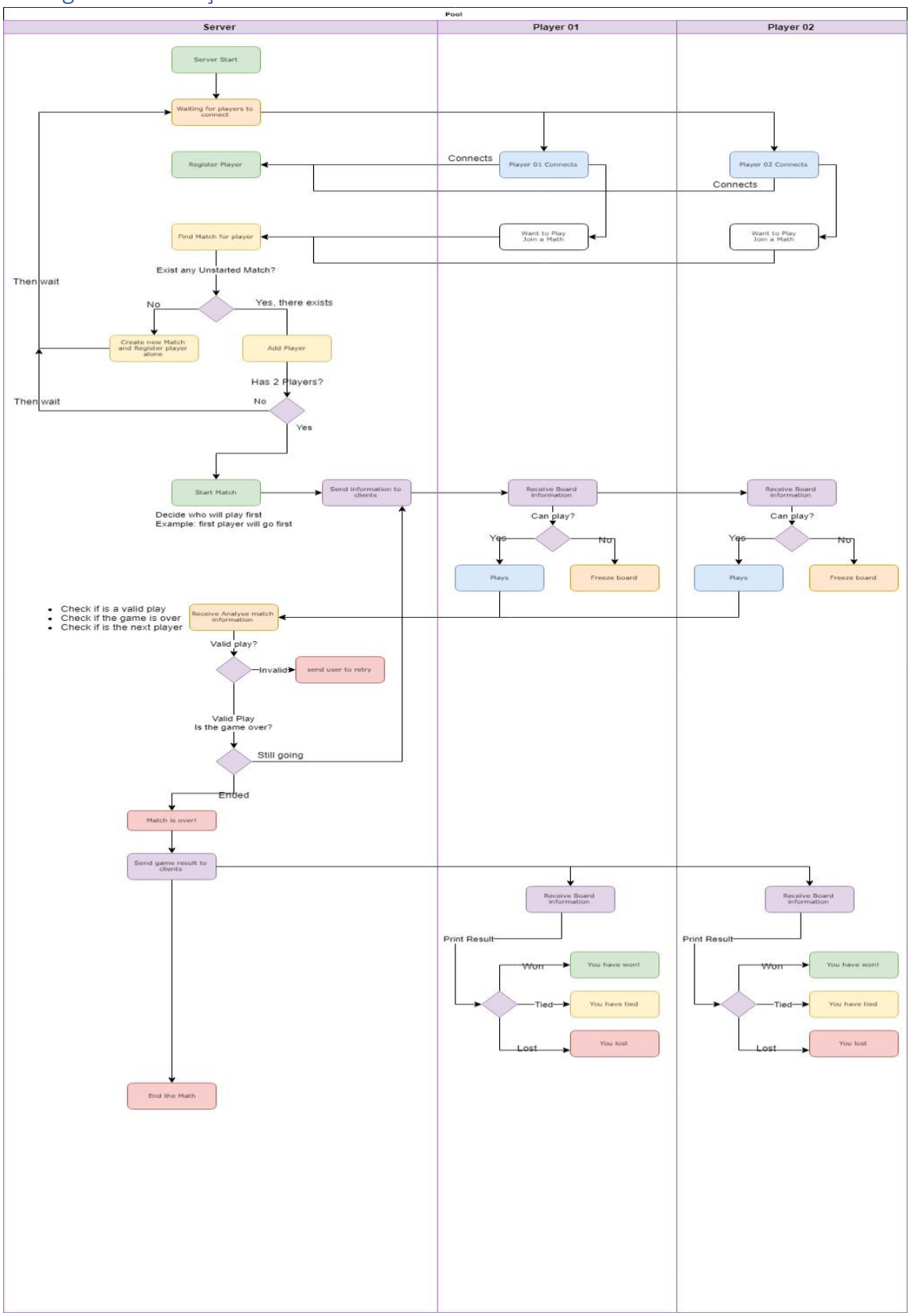
- O player quando faz o pedido para entrar no jogo "`game.join(player)`", faz a requisição de `player`.
- Se exitir um jogo já iniciado, adiciona o player no match e inicia o jogo. "`match.setAdversary(p); p.setMatch(match); match.setState(MatchState.RUN)`. "
- Caso contrário, cria novo jogo "`match = p.createMatch()`; regista o jogo na lista de jogos "`matches.add(match)`" e regista o "player alone", o servidor devolve a mensagem "`System.out.println("Wait for a match to start.")`", o player aguarda enquanto não começa o jogo, o que pode causar problemas porque pode ficar bloqueado por tempo indeterminado.

Jogar

- Entra no jogo o servidor envia mensagem: "`System.out.println("Start The Game. Good luck!")`"
- O servidor decide quem joga primeiro
- Envia essa informação aos clientes (players)
- Recebe a informação da Board e pede a player para jogar, caso contrário aguarda: "`if (game.isMyTurn()){if(game.iCanPlay()) performMove();else game.waitMyTurn();`"
- Recebe a informação do player, se é valida, se for valida imprime no tabuleiro, caso contrário envia informação a dizer que a jogada é invalida. "`if (game.isValid(row, col))game.move(row, col} else {System.out.println("This field is not free. Try again.");Util.pressToContinue();`"
- Valida se o jogo acabou: `if (game.haveWinner()) Util.showWinner(game.Winner().getUsername());else if (game.isLocked()) til.showTied();`"

- Se o jogo não acabou pede a jogada do player seguinte.
- Quando o jogo acabar imprime as opções Won ou Tied.
- Volta para o menu principal.

Fluxograma das acções



Gestão do projeto

Tecnologia usadas

Ferramentas e Softwares

IntelliJ IDEA da Jetbrains
GitHub Desktop
Visual Studio Code
Draw.io

Linguagens de programação e frameworks

Java Runtime Environment 14(JRE)
Java SE Development Kit 14 (JDK)

Recurso Humanos

Eduardo Fonseca
João Pereira

Com a orientação de Pedro Mestre (Docente da cadeira **Redes de Computadores**)