

# Programação Distribuída - Remote Gaming Server

Autor: Ariel Rossetto Ril  
Matrícula: 15105050  
Email: [ariel.ril@edu.pucrs.br](mailto:ariel.ril@edu.pucrs.br)

## Introdução

Neste trabalho é apresentado uma implementação de um servidor remoto para um jogo onde os jogadores podem conectar no servidor (**register**), executar jogadas (**play**) e parar de jogar (**stop**). O servidor pode se comunicar com os jogadores para "iniciar" um jogador (**start**), verificar se um jogador está vivo (**liveness**) e bonificar um jogador após uma jogada (**bonus**).

O servidor desenvolvido neste trabalho utiliza o protocolo de comunicação **Java RMI** (*Remote Method Invocation*), o qual permite que um programa execute chamadas a métodos disponíveis em outros programas de forma remota. Para que exista uma comunicação nos dois sentidos (servidor ↔ jogador), foi desenvolvido um servidor para o jogo e um servidor para o jogador fazendo com que o servidor possa executar invocações remotas em métodos do jogador da mesma maneira que o jogador executa invocações remotas no jogo.

## Desenvolvimento

Para realizar a utilização de **Java RMI**, todo desenvolvimento foi realizado utilizando a linguagem de programação Java.

## Organização

Neste projeto foi desenvolvido 5 classes, 2 interfaces e um enum:

1. Interfaces
  1. IGame
  2. IPlayer
2. Classes
  1. Game → implementa IGame
  2. GameServer
  3. Player → implementa IPlayer
  4. PlayerServer
  5. Result
3. Enum
  1. ResultType

### Interface IGame

Esta interface disponibiliza os métodos para serem utilizados durante a comunicação entre jogador e jogo.

```
public interface IGame extends Remote {  
    public int register() throws RemoteException, ServerNotActiveException;  
    public int play(int playerId) throws RemoteException;  
    public int giveUp(int playerId) throws RemoteException;  
    public int stop(int playerId) throws RemoteException;  
    public void getResult(int playerId, int result, ResultType resultType) throws RemoteException;  
}
```

Todos os métodos da interface devem retornar a exceção `RemoteException`, pois esta é uma exceção que pode originar de uma comunicação remota utilizando **RMI**.

### Interface IPlayer

Esta interface disponibiliza os métodos para serem utilizados durante a comunicação entre o jogo e jogador.

```
public interface IPlayer extends Remote {  
    public void start() throws RemoteException;  
    public void bonus() throws RemoteException;  
    public void check() throws RemoteException;  
    public void getResult(int result, ResultType resultType) throws RemoteException;  
}
```

Todos os métodos da interface devem retornar a exceção `RemoteException`, pois esta é uma exceção que pode originar de uma comunicação remota utilizando **RMI**.

### Classe GameServer

Esta classe implementa a funcionalidade de servidor do jogo, o qual realiza a iniciação do jogo após os jogadores estarem registrados, realiza a solicitação de processamento de resultados dentro do jogo - solução de envio de resultado assíncrono para clientes - e solicitação para o jogo verificar se seus jogadores estão "vivos".

Nesta classe existe dois métodos principais:

- `public static void main(String[] args) throws RemoteException`
  - Inicializa a instância do servidor do jogo e o jogo
- `private static void run(Game game)`
  - Executa as funcionalidades do jogo, uma forma de separar a responsabilidade de executar as funcionalidades do jogo e "ser" o jogo

## Classe Game

Esta classe implementa a interface `IGame` para que os métodos definidos na interface possam ser invocados remotamente por algum jogador. Nesta classe foi implementado todas funcionalidade que um jogo possui:

- `public boolean isGameReady()`
  - verifica se já é possível iniciar o jogo
- `public boolean runningWithNoPlayers()`
  - verifica se ainda existe algum jogador vivo no jogo
- `public void start()`
  - inicia o jogo
- `public int register() throws RemoteException`
  - registra os jogadores
- `public void checkPlayers()`
  - verifica se os jogadores estão "vivos"
- `public int play(int playerId) throws RemoteException`
  - executa uma jogada de um jogador
- `public int giveUp(int playerId) throws RemoteException`
  - executa a desistência de um jogador
- `public int stop(int playerId) throws RemoteException`
  - executa a finalização do jogo para um jogador
- `public void processResults()`
  - processa a lista de resultados do jogo. Este método foi criado para que quando um jogador executa uma ação no jogo, o jogo possa enviar os resultados de maneira assíncrona para não "travar" o jogo
- `public void getResult(int playerId, int result, ResultType resultType) throws RemoteException`
  - recebe resultados de ações executadas nos jogadores

## Classe PlayerServer

Esta classe foi desenvolvida para realizar o controle das ações que um jogador irá executar durante a sua execução. Nesta classe existe 3 métodos principais:

- `public static void main(String[] args) throws RemoteException`
  - inicializa e realiza o registro do servidor para o jogador
- `private static void run(Player player)`
  - executa as funcionalidades de um jogador: registro, execução de jogada, envio de resultados e finalização do processo
- `private static void registration(Player player)`
  - executa o registro de um jogador no jogo. Este método foi separado porque é executado um processo de *rebind* do local onde o jogador esta definido. Ao inicializar um jogador ele é registrado em uma porta padrão no *registry* e após o registro do jogador no jogo é realizado a modificação do registro do jogador no *registry* para facilitar a inicialiação dos processos e a resolução de nomes do jogador dentro do jogo.

## Classe Player

Esta classe implementa a interface `IPlayer` para que os métodos definidos na interface possam ser invocados pelo jogo de forma remota. Os principais métodos definidos nesta classe são:

- `public void play()`
  - este método executa uma jogada no jogo
- `public void start() throws RemoteException`
  - este método é utilizado pelo jogo para informar ao jogador que o jogo foi iniciado
- `public void bonus() throws RemoteException`
  - este método é utilizado pelo jogo para informar ao jogador do recebimento de um bonus em um de suas jogadas
- `public void check() throws RemoteException`
  - este método é utilizado pelo jogo para solicitar ao jogador que informe ao jogo se esta vivo
- `public void getResult(int result, ResultType resultType) throws RemoteException`
  - este método é responsável por receber o resultado de alguma ação realizada no jogo
- `public void processResults()`
  - este método é responsável por enviar os resultados gerados de uma ação executada pelo jogo no jogador
- `public void stopPlaying()`
  - este método executa a finalização de jogo para um jogador, solicitando ao jogo para que o jogador seja removido da lista de jogadores

## Classe Result

Esta classe foi desenvolvida apenas para servir como um envelope que contém o resultado de alguma ação executada no jogo ou no jogador.

```
public class Result {  
    public ResultType type;
```

```
public int value;
public int playerId;

public Result(int playerId, ResultType type, int value) {
    this.type = type;
    this.value = value;
    this.playerId = playerId;
}
}
```

## Demonstração

## Compilação

Para compilar (*build*) o código é necessário ir para a pasta `data/code`, a qual contém o código Java da implementação. Após estar dentro da pasta que contém o código é necessário executar o comando `make` para compilar o código Java e gerar os arquivos `.class`.

## Comandos

```
cd <project_root>/data/code
make # gera os arquivos .class
```

# Máquinas Virtuais

Para iniciar as máquinas virtuais para a demonstração é necessário ter a ferramenta **vagrant** (<https://www.vagrantup.com>) instalada além de possuir VirtualBox instalado. Após instalar as ferramentas necessárias é preciso estar na pasta **machine**, a qual contém o arquivo **Vagrantfile** com as configurações necessárias para iniciar 3 máquinas virtuais, para executar o comando **vagrant up**.

```
distprog-game-server.nosync/machine on master [!] via v2.2.15 took 4s
> vagrant up
Bringing machine 'host-1' up with 'virtualbox' provider...
Bringing machine 'host-2' up with 'virtualbox' provider...
Bringing machine 'host-3' up with 'virtualbox' provider...
==> host-1: Checking if box 'debian/buster64' version '10.20210228.1' is up to date...
==> host-1: A newer version of the box 'debian/buster64' for provider 'virtualbox' is
==> host-1: available! You currently have version '10.20210228.1'. The latest is version
==> host-1: '10.20210409.1'. Run `vagrant box update` to update.
```

## Simulação

Para executar a simulação é possível iniciar a quantidade que quiser de conexões com as máquinas virtuais, para este trabalho foi utilizado 3 máquinas virtuais e 6 conexões (duas para cada máquina virtual).



Na máquina 1 (**host-1**) foram executados o servidor do jogo (**GameServer**) e o jogador 1. Na máquina 2 (**host-2**) foram executados os jogadores 2 e 3. Na máquina 3 (**host**) foram executados os jogadores 4 e 5. Após conectar nas máquinas virtuais é preciso modificar o diretório para `~/code`, o qual é sincronizado com o diretório `data/code` da máquina local.

Após inicializar o servidor do jogo e inicializar todos os jogadores, o programa irá inicializar o jogo e todos os jogadores irão começar a executar as suas jogadas.

[illegible]

Assim que todos jogadores finalizam suas jogadas e saem do jogo, o servidor do jogo finaliza o processo.

```

X1 machine vagrant ssh host-1 -- vagrant (ssh)
[1] Player (3) is playing
[2] player (3) is sleeping for [562]ms
[3] player (3) is playing
[4] player (3) is playing
[5] player (3) is playing
[6] player (3) is playing
[7] player (3) is playing
[8] Checking player (1) == [mml:/192.168.35.38:9999/player_server]
[9] Checking player (2) == [mml:/192.168.35.38:9999/player_server]
[10] Checking player (3) == [mml:/192.168.35.38:9999/player_server]
[11] Checking player (4) == [mml:/192.168.35.38:9999/player_server]
[12] Checking player (5) == [mml:/192.168.35.38:9999/player_server]
[13] player (3) is alive
[14] player (3) stopped playing
[15] player (3) is alive
[16] player (3) is alive
[17] player (3) is playing
[18] player (3) is playing
[19] player (4) stopped playing
[20] player (4) is alive
[21] player (4) stopped playing
[22] player (3) is playing
[23] player (3) is playing
[24] player (2) stopped playing
[25] player (2) is playing
[26] player (2) stopped playing
[27] player (2) is playing
[28] player (2) stopped playing
[29] player (2) is playing
[30] player (2) got a bonus
vagrant@host1:~$ cat /dev/null > /dev/null && exit 0
vagrant@host1:~/codes {}

X2 machine vagrant ssh host-2 -- vagrant (ssh)
[1] Player (2) is playing
[2] player (2) have (3) plays left
[3] player (2) is sleeping for [573]ms
[4] player (2) played
[5] player (2) have (6) plays left
[6] player (2) is sleeping for [452]ms
[7] player (2) played
[8] player (2) was checked for Liveness
[9] player (2) have (3) plays left
[10] player (2) is sleeping for [166]ms
[11] player (2) played
[12] player (2) have (4) plays left
[13] player (2) is sleeping for [302]ms
[14] player (2) played
[15] player (2) have (2) plays left
[16] player (2) is sleeping for [308]ms
[17] player (2) played
[18] player (2) was checked for Liveness
[19] player (2) have (2) plays left
[20] player (2) is sleeping for [729]ms
[21] player (2) played
[22] player (2) have (1) plays left
[23] player (2) is sleeping for [681]ms
[24] player (2) played
[25] player (2) have (8) plays left
[26] player (2) is sleeping for [183]ms
[27] player (2) played
[28] player (2) is stopping
[29] player (2) stopped. Bye!
[30] player (2) got a bonus
vagrant@host2:~/codes {}

X3 machine vagrant ssh host-3 -- vagrant (ssh)
[1] Player (4) have (7) plays left
[2] player (4) is sleeping for [878]ms
[3] player (4) played
[4] player (4) was checked for Liveness
[5] player (4) have (6) plays left
[6] player (4) is sleeping for [792]ms
[7] player (4) played
[8] player (4) have (5) plays left
[9] player (4) is sleeping for [57]ms
[10] player (4) played
[11] player (4) have (4) plays left
[12] player (4) is sleeping for [861]ms
[13] player (4) played
[14] player (4) was checked for Liveness
[15] player (4) have (3) plays left
[16] player (4) is sleeping for [931]ms
[17] player (4) played
[18] player (4) have (2) plays left
[19] player (4) is sleeping for [912]ms
[20] player (4) played
[21] player (4) have (1) plays left
[22] player (4) is sleeping for [106]ms
[23] player (4) played
[24] player (4) was checked for Liveness
[25] player (4) have (8) plays left
[26] player (4) is sleeping for [379]ms
[27] player (4) played
[28] player (4) is stopping
[29] player (4) stopped. Bye!
[30] player (4) got a bonus
vagrant@host3:~/codes {}

X4 machine vagrant ssh host-4 -- vagrant (ssh)
[1] Player (3) was checked for Liveness
[2] player (3) have (7) plays left
[3] player (3) is sleeping for [153]ms
[4] player (3) played
[5] player (3) have (4) plays left
[6] player (3) is sleeping for [106]ms
[7] player (3) played
[8] player (3) have (3) plays left
[9] player (3) is sleeping for [777]ms
[10] player (3) played
[11] player (3) was checked for Liveness
[12] player (3) have (2) plays left
[13] player (3) is sleeping for [688]ms
[14] player (3) played
[15] player (3) have (3) plays left
[16] player (3) is sleeping for [190]ms
[17] player (3) played
[18] player (3) have (2) plays left
[19] player (3) is sleeping for [361]ms
[20] player (3) was checked for Liveness
[21] player (3) have (1) plays left
[22] player (3) is sleeping for [405]ms
[23] player (3) played
[24] player (3) have (4) plays left
[25] player (3) is sleeping for [424]ms
[26] player (3) played
[27] player (3) is stopping
[28] player (3) stopped. Bye!
[29] player (3) got a bonus
vagrant@host4:~/codes {}

X5 machine vagrant ssh host-5 -- vagrant (ssh)
[1] Player (3) have (7) plays left
[2] player (3) is sleeping for [878]ms
[3] player (3) played
[4] player (3) was checked for Liveness
[5] player (3) have (6) plays left
[6] player (3) is sleeping for [792]ms
[7] player (3) played
[8] player (3) have (5) plays left
[9] player (3) is sleeping for [57]ms
[10] player (3) played
[11] player (3) have (4) plays left
[12] player (3) is sleeping for [861]ms
[13] player (3) played
[14] player (3) was checked for Liveness
[15] player (3) have (3) plays left
[16] player (3) is sleeping for [931]ms
[17] player (3) played
[18] player (3) have (2) plays left
[19] player (3) is sleeping for [106]ms
[20] player (3) played
[21] player (3) have (1) plays left
[22] player (3) is sleeping for [892]ms
[23] player (3) played
[24] player (3) was checked for Liveness
[25] player (3) have (8) plays left
[26] player (3) is sleeping for [213]ms
[27] player (3) played
[28] player (3) is stopping
[29] player (3) stopped. Bye!
[30] player (3) got a bonus
vagrant@host5:~/codes {}

```