ESTGOH
Escola Superior de Tecnologia e Gestão de Oliveira do Hospital

Instituto Politécnico de Coimbra

Licenciatura em Engenharia Informática

Programação Aplicada 2019/2020

Projeto nº 2

Programação Orientada a Objetos, Estruturas de Dados, Manipulação de Ficheiros, Comunicação em Rede

Elaborado em: 05/2020
Francisco Gabriel Fonseca Mesquita

Índice

Lista de Figuras

1 Introdução

Neste trabalho vou fazer uso dos conhecimentos aprendidos e praticados ao longo do primeiro semestre nas aulas de programação assim como os novos conceitos dados em programação aplicada tais como manipulação de datas e essencialmente comunicação em Redes através de Sockets.

Através da linguagem de programação JAVA tenho como objetivo desenvolver um aplicativo que cumpra com tudo que é pedido no enunciado do problema, É um aplicativo que tem como ideia principal a criação de um jogo Sudoku onde deve ser possível jogar sozinho contra o computador ou jogar contra outra pessoa em rede, numa ligação um para um através de Sockets. Estas são as principais funcionalidades da aplicação, mas há ainda outras como fazer listagens, ver replays, consultar o log entre algumas outras.

Para guardar os dados de forma permanente poderia ter sido usado base de dados mas neste caso optei por o uso de ficheiro, irei explicar um pouco mais esta escolho ao longo do relatório.

2 Implementação

2.1 Requisitos implementados:

Não consegui implementar os recursos R23, R25 E R33:

R23: Implementar a possibilidade de "dica" (i.e., o computador indica ao utilizador uma jogada provável – soluções com algoritmos que procurem jogadas realistas serão bonificadas).

R25: Sempre que um jogador ultrapasse múltiplos de 10 vitórias, deve surgir uma notificação imediata de congratulação.

R33: Deve ser possível ordenar jogadores por tempo mais rápido de jogo.

A justificação para a não realização do R23 foi a tentativa de implementar um algoritmo "inteligente" ao dar a dica, mas não obtive sucesso, para o R25 e R33 não dava para fazer da forma que implementei o programa.

Tudo o resto que não foi mencionado aqui foi realizado.

2.2 Classes implementadas e a lógica das mesmas:

Foram implementadas 17 classes, vou representar aqui as mesmas agrupadas pela função que têm no programa:

Classes que Caracterizam objetos a ser usados (classes modelo)

- **Jogador**
- **Jogada**
- **Tabuleiro**

Todas estas classes terão de ser geridas á parte de forma a terem alguma relevância no programa, é aí que entra o próximo tipo de classes.

Classes que gerem outras objetos de outras classes

- **Client**
- **Server**
- **RegistoJogo**
- **Log**
- **GereFicheiroObjeto**
- **GereFicheiroTexto**
- **GereJogadores**
- **GerirJogoMultiplayer**
- **GerirRegistoJogos**
- **ProtocolosMultiplayer**
- **TempoExecucao**

Todas estas classes são responsáveis para gerir objetos, são nestas classes que são implementados todos os **métodos necessários para fazer a ligação aos Sockets, mostrar informações sobre o tempo decorrido na aplicação, gerir o ficheiro log, assim como classes que gerem diretamente ou indiretamente as classes modelo.**

Entre classes que representa um objeto e classes que gerem esses objetos é preciso existir classes que comunicam com o utilizador, sendo esta esta classe a responsável por isso:

- **Aplicacao**

Classe "Aplicacao" funciona com "ponte" entre a classe de um objeto e a classe gere destinada a gerir esse objeto, é a única classe que comunica diretamente com o jogador.

Classes uteis que não se encaixam exatamente em nenhum dos outros grupos:

- **ObjectOutputStreamExtended**

Esta classe é necessária para **reescrever o método ObjectOutputStream(),** sendo isto necessário para não haver conflito com o cabeçalho do ficheiro de texto ao escrever um objeto ou mesmo na sua leitura posteriormente.

Classe Main

A classe Main é a classe responsável para o programa "arrancar", é aqui que começa a execução do programa.

A classe Main foi projetada para ser uma organizadora de métodos, ou seja, todos os métodos aplicados no método main (a maioria) não apresentam interação com o utilizador (essa função é da classe Aplicação).

Ainda na Main tenho vários métodos que não se encaixavam propriamente em nenhuma das outras classes ou não havia necessidade de ter uma classe apenas para aquele método como é o caso dos métodos para escrever e ler objetos no ficheiro, ou até mesmo os métodos de simulação de jogadas e sequência de jogadas.

Conexão em rede através de Sockets

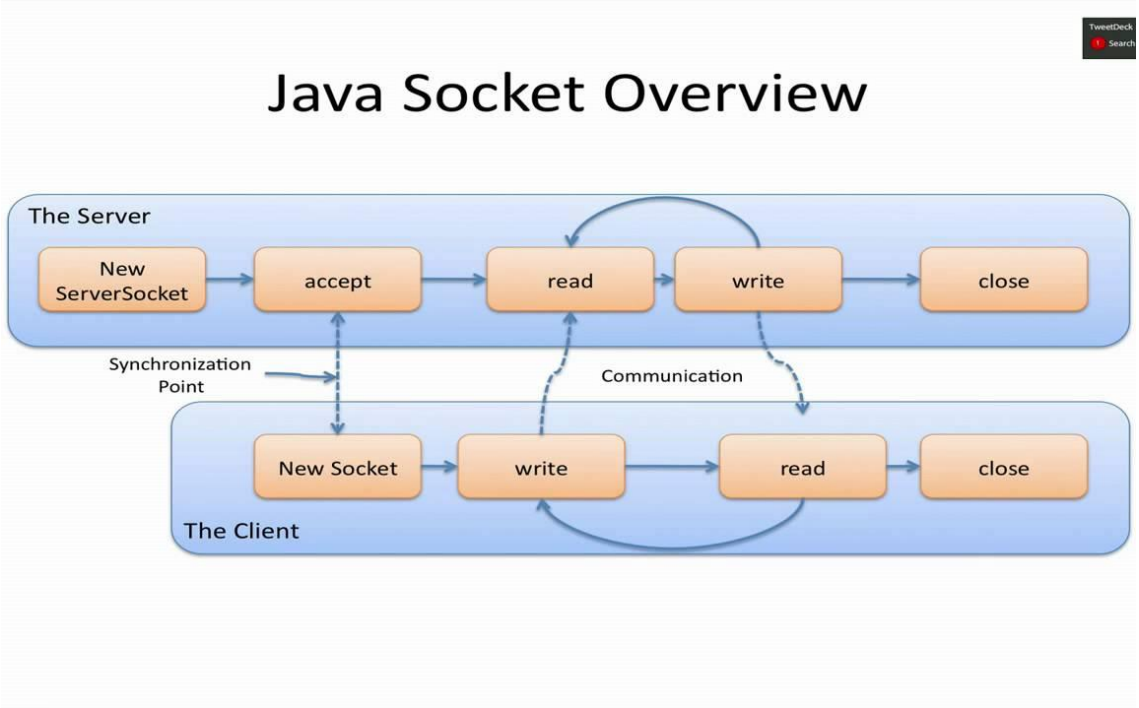

Figure 1- Lógica da comunicação entre Sockets

Esta figura apresentada em cima é perfeita para se perceber de uma forma muito simples o funcionamento da comunicação entre Sockets, basicamente **para existir uma ligação entre sockets temos de ter sempre pelo menos um cliente e um servidor**, no caso desta aplicação é uma ligação 1 para 1 (1 servidor para 1 cliente):

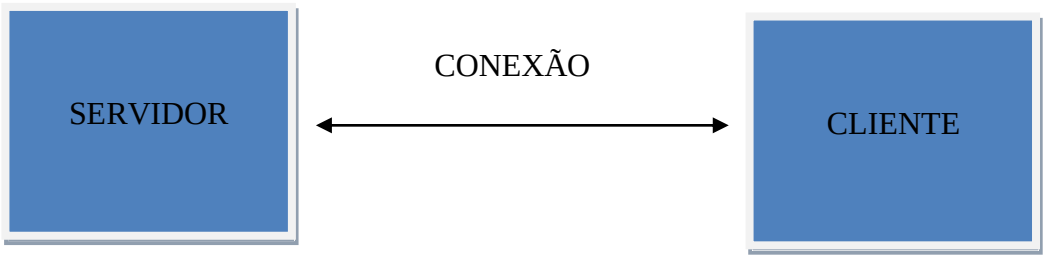

Figure 2- Ligação 1 para 1 entre cliente e servidor

Depois de se perceber que se trata de uma ligação 1 para 1 entre cliente e servidor, estes devem ser ligados um ao outro sendo que o **servidor vai ficar á espera da ligação numa determinada porta na rede, assim o cliente através do host e da porta consegue se ligar, ficando assim estabelecida a comunicação**.

Chegamos assim á zona onde se nota claramente a existência de um ciclo:

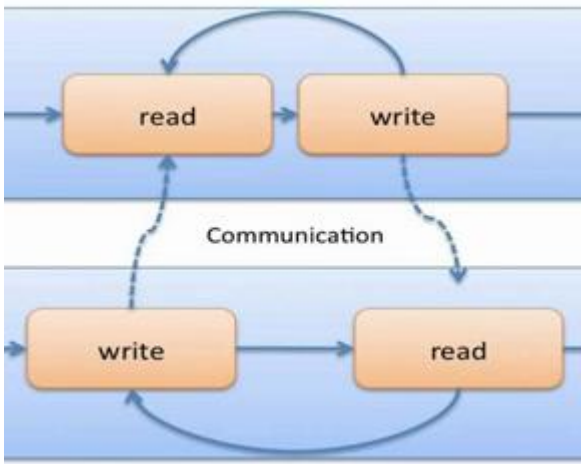

Figure 3- Ciclo na comunicação entre sockets

Para ser possível explicar o ciclo apresentado na figura é necessário antes perceber uma coisa muito importante neste tipo de ligações:

- **Quando o server emite alguma mensagem o cliente tem de estar á escuta.**
- **Quando o cliente emite alguma mensagem o server tem de estar á escuta.**

Percebendo esta regra facilmente se percebe que tem de existir aqui um ciclo para que tudo funcione bem, é exatamente isso que está descrito na figura acima.

Para finalizar a conexão entre as duas partes deve-se fechar ambas as ligações, tanto cliente como servidor.

2.3 Ficheiros usados na aplicação:

2.3.1 Motivo para o uso de ficheiros:

Neste projeto decidi usar ficheiros por as seguintes razões:

- Relembrar tudo o que foi dado e utilizado.
- Vantagens em termos de manipulação embora algumas desvantagens em escrita e leitura dos mesmos.
- Conseguir testar com mais facilidade pois posso testar o mesmo programa em locais diferentes com ficheiros diferentes, o que não sucedia se fosse utilizado base de dados onde teria vários acessos á mesma base de dados ao mesmo tempo, o que poderia causar erros e dificuldades se essa questão não fosse bem trabalhada.

2.3.2 Tipo de ficheiros utilizados e sua respetiva função:

Foram utilizados ficheiros de texto e 1 ficheiro de objetos.
Ficheiros de texto utilizados e suas funcionalidades:

jogadores.txt: serve para guardar as informações dos vários jogadores.

jogoStats.txt: serve para guardar os recordes relacionados ao jogo.

log.txt: serve para guardar os registos das ações dos utilizadores.

Ficheiro de objetos:

registoJogos: Serve para guardar objetos da classe GerirRegistoJogos que por sua vez contem objetos da classe RegistoJogo, esta por sua vez contem objetos do tipo Jogada.

2.4 Algoritmos e estruturas de dados implementados

Sobre os algoritmos e as estruturas de dados implementados é sempre complicado de falar porque às vezes há dezenas de formas de resolver um problema não havendo uma solução ideal, eu tentei sempre criar algoritmos eficazes e reutilizáveis de acordo com a solução que para mim era a ideal.

Para a realização destes algoritmos foram utilizadas várias estruturas de dados, todas elas que para mim facilitaram muito:

Uso de vetores: foram utilizados várias vezes vetores para retornar jogadas entre outros pelo simples facto de estar bastante habituado a utilizar os mesmos e também porque muitas vezes não era possível saber o tamanho que a estrutura iria ter.

Uso de Comparator: foi utilizado Comparator, inicialmente introduzido em java 8 pois com optei por usar apenas o método CompareTo para o nickname, sendo que o seguintes parâmetros que tive necessidade de organizar foram todo organizados com o Comparator.

Metodo random(): Este método foi utilizado dentro de outro método para ser possível obter um número aleatório num intervalo de números.

Objetos duration: foram utilizados por serem na minha opinião a melhor opção para se calcular tempo entre alguma ação.

Hashmap: sendo esta estrutura uma peça chave no meu código seria difícil não a referir, a hasmap serve essencialmente para reter todas as jogadas com as posições das mesmas como “key” na estrutura e o valor associado á “key” é um tipo de dados boolean que nos diz se aquela jogada é ou não fixa, para este tipo de implementação a hashmap encaixou muito bem.

Houve vários outros algoritmos e estruturas implementadas, mas estas são na minha opinião as mais importantes de falar e explicar porque optei por elas.

3 Distribuição de tarefas e respetivo tempo

O planeamento inicial projetado foi o seguinte:

1. Perceber todo o enunciado e esboçar a sua execução.
2. Decidir entre o uso de base de dados ou ficheiros.
3. Construir classes modelo para representação dos principais objetos.
4. Fazer a autenticação dos jogadores carregando as estatísticas necessárias através do nickname escolhido.
5. Fazer os menus necessários para fazer de seguida o jogo contra o computador.
6. Implementar os métodos necessários para criar o jogo contra o computador.
7. Criar as classes necessárias para a comunicação entre sockets.
8. Fazer a comunicação entre sockets inicial, apenas perceber se comunicam.
9. Implementar toda a comunicação entre sockets seguindo os protocolos fornecidos.
10. Fazer notificações.
11. Fazer listagens.
12. Fazer o log
13. Fazer a listagem das sequências de jogadas para cada jogo, mantendo as mesmas de forma permanente.
14. Fazer a simulação dos vários jogos.

Tarefas como **relatório, manual de utilizador** e **javaDocs** foram feitas durante a implementação do código. Dentro das tarefas utilizadas a que tive mais dificuldade e que por acaso foi a que demorei mais a implementar foi a tarefa número 9.

4 Possíveis melhorias

Apesar de ter cumprido a grande maioria dos requisitos pedidos, acho que o meu trabalho não está organizado de forma totalmente correta sendo que eu reconheço e irei melhor futuramente alguns dos aspetos que irei referir, portanto para além da **implementação do que falta**, há espaço para melhor ainda os seguintes requisitos:

- Melhor prevenção contra o que o utilizador vai escrever.
- Métodos mais curtos e objetivos.
- Menos repetição de código
- Algoritmos mais eficiente
- Modelo MVC.
- Funcionalidade não pedidas, mas que podiam e deviam estar presentes.
- Acrescentar funcionalidades pedidas, mas não implementadas.

5 Testes

Foram realizados testes a praticamente tudo de forma a confirmar tudo que foi feito.

- Guardar nickname do jogador caso seja a primeira vez que joga:

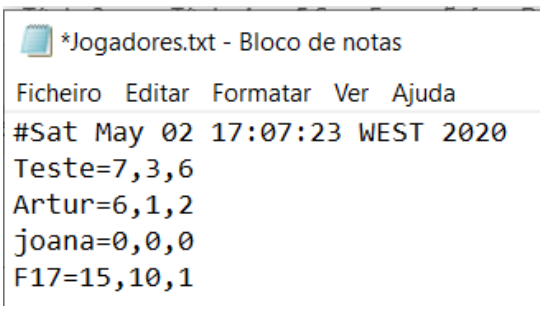

Figure 4- documento jogadores.txt antes da inserção de um novo nickname

Vou proceder á inserção do novo jogador:

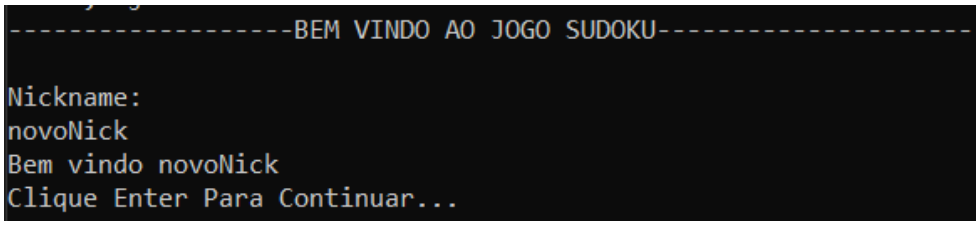

Figure 5- inserção do novo nickname

Depois de inserido o nome de utilizador pode-se verificar no documento dos jogadores:

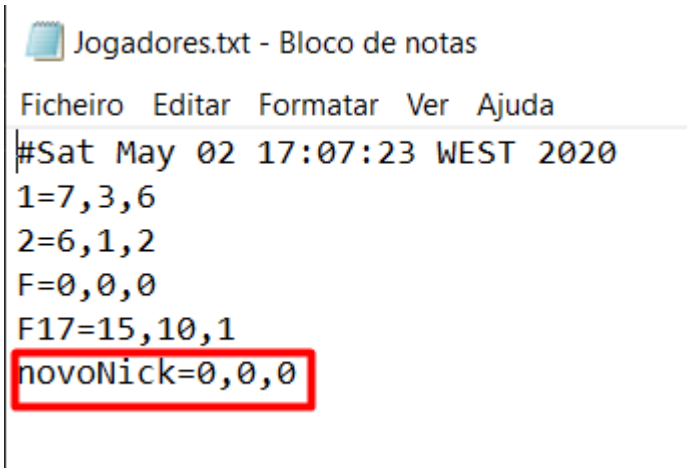

Figure 6- confirmação do novo nickname no ficheiro

Testes no jogo contra o computador:

- Teste de inserção de uma jogada:

Primeiro é necessário colocar a jogada e perceber se está tudo a correr bem e a ser mostrada a informação necessária:

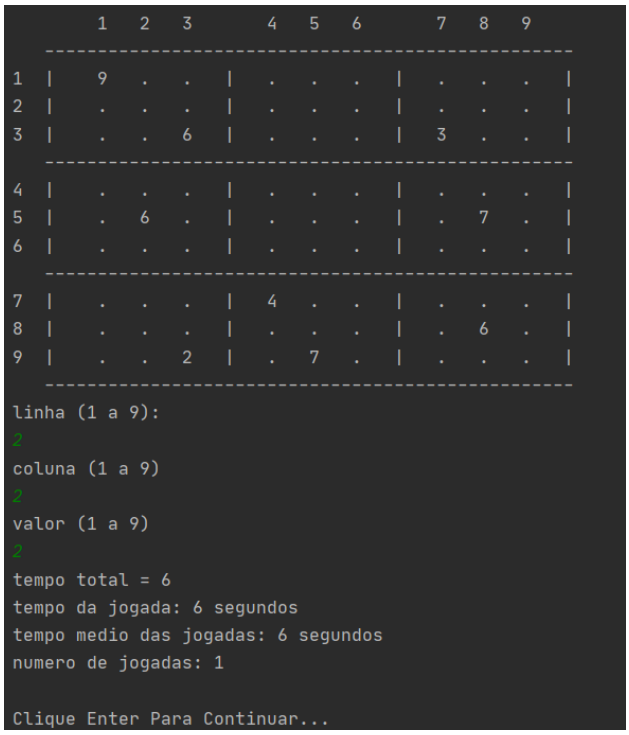

Figure 7- Inserir uma jogada contra o computador

Depois de feita a jogada, é necessário perceber se o novo valor ficou no tabuleiro:

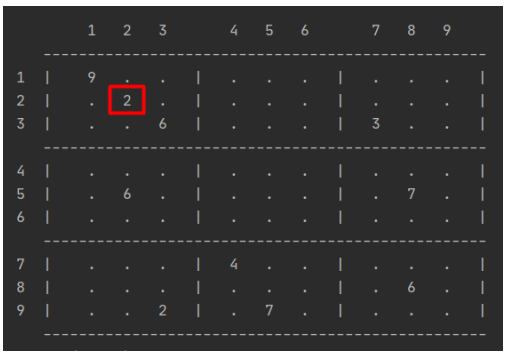

Figure 8- novo valor no tabuleiro

É assim possível verificar que a jogada foi feita com sucesso. Depois da inserção da jogada e da confirmação da mesma foram feitas mais umas rodadas e depois o jogo foi terminado através da opção para desistir.

Teste do jogo através de sockets:

- Estabelecer comunicação entre sockets

Para testar isto é preciso primeiro iniciar o servidor para ficar á espera de uma ligação:

Figure 9- Servidor á espera de ligação

Depois de colocar o servidor é então necessário abrir a aplicação em outro local para estabelecer a ligação:

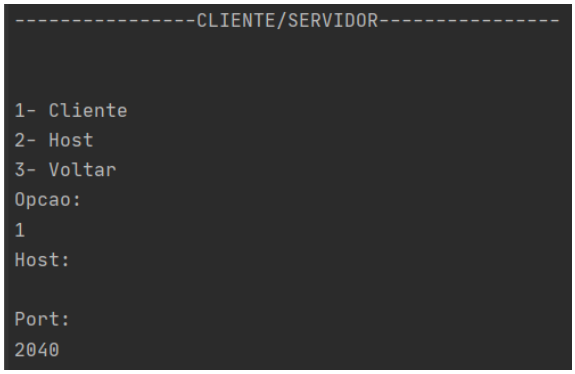

Figure 10- Cliente a ligar-se ao servidor

Depois de tudo inserido é altura de perceber se a ligação foi estabelecida em ambos os lados:

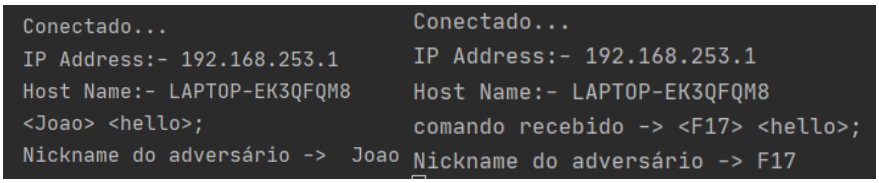

Figure 11- conexão entre cliente e servidor estabelecida

Tal como é possível observar na imagem acima a ligação foi assim estabelecida.

Testar inserção de jogadas iniciais e jogadas posteriores:

- Inserir jogadas iniciais por parte do lado do servidor:

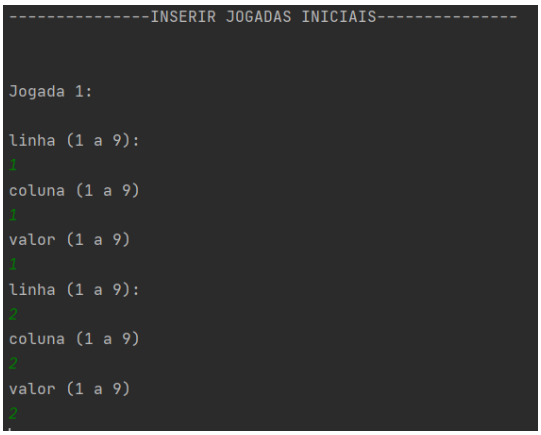

Figure 12- Inserir jogadas iniciais

- Inserir jogadas iniciais por parte do lado do cliente:

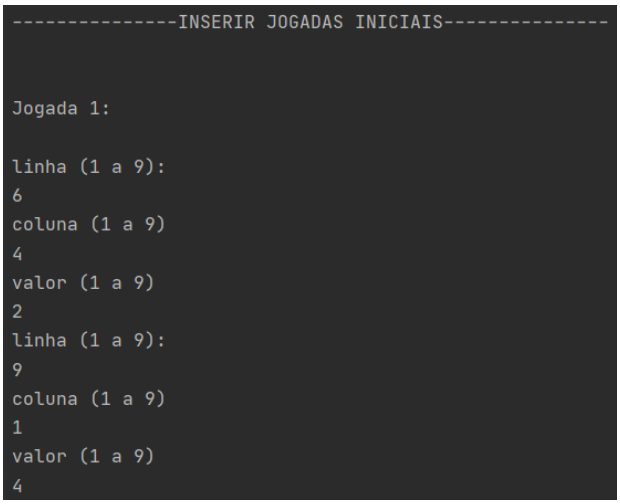

Figure 13- Inserir jogadas iniciais por parte do cliente

Depois disso é possível confirmar as jogadas inseridas visualizando os tabuleiros, o exemplo a seguir que irei mostrar são os tabuleiros visto da perspetiva do lado do servidor:

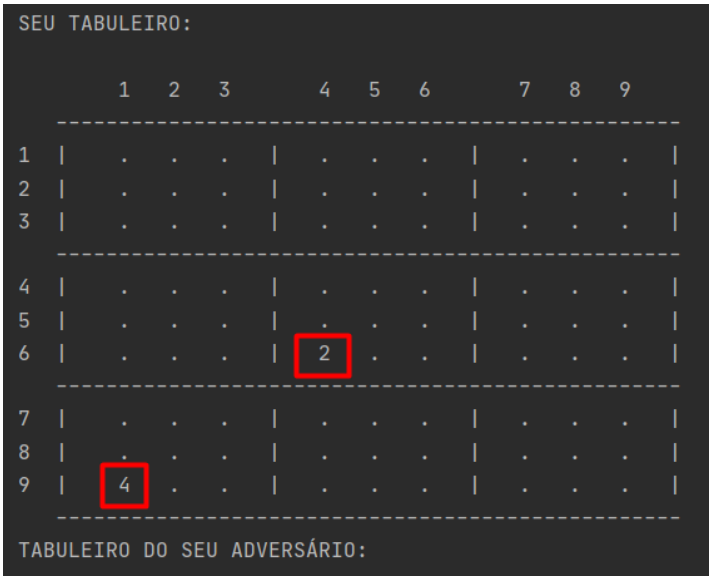

Figure 14- tabuleiro do lado do servidor

É possível perceber que os valores e posições correspondem aos números escolhidos pelo cliente para serem ali colocados, agora vendo o outro tabuleiro:

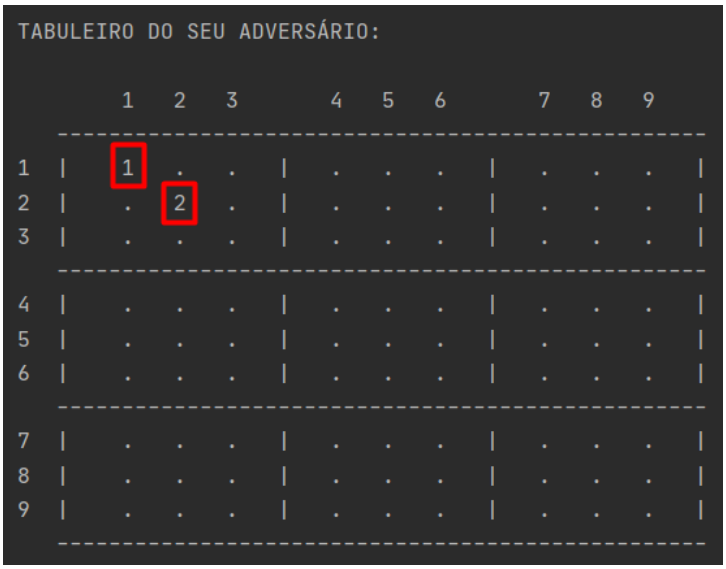

Figure 15- tabuleiro do lado do cliente

Mais uma vez confirma-se que tudo correu como o esperado sendo os números e respetivas posições exatamente iguais ao escolhido.

- Inserir jogada depois das iniciais:

Depois de inserir as jogadas iniciais cada um passa a jogar no seu tabuleiro, assim foi testada a jogada individual do servidor:

```
1- Inserir jogada
2- Ver tabuleiros
3- Desistir

Opcao:
1
linha:
9
coluna:
9
valor:
9
comando recebido -> <Joao> <result> <valid>;
```

Figure 16- jogada inserida

Depois de inserida a jogada, é altura de confirmar no tabuleiro:

```
      1   2   3       4   5   6       7   8   9
    -------------------------------------------------
1   |   1   .   .   |   .   .   .   |   .   .   .   |
2   |   .   2   .   |   .   .   .   |   .   .   .   |
3   |   .   .   .   |   .   .   .   |   .   .   .   |
    -------------------------------------------------
4   |   .   .   .   |   4   .   .   |   .   .   .   |
5   |   .   .   .   |   .   .   .   |   .   .   .   |
6   |   .   .   .   |   .   .   .   |   .   .   .   |
    -------------------------------------------------
7   |   .   .   .   |   .   .   .   |   .   .   .   |
8   |   .   .   .   |   .   .   .   |   .   .   .   |
9   |   .   .   .   |   .   .   .   |   .   .   9   |
    -------------------------------------------------
```

Figure 17- Número na posição pretendida

Mais uma vez tudo correu tal como o esperado.

- Testar fim de jogo:

Para motivos de testes não se justificava acabar um jogo pois tudo o que implica terminar o jogo pode ser testado com a função desistir:

```
1- Inserir jogada
2- Ver tabuleiros
3- Desistir

Opcao:
3
comando recebido -> <F17> <withdraw> <ack>;
1
comando recebido -> <F17> <new> <?>;
----------------COMEÇAR NOVO JOGO?---------------


1- SIM
2- NAO

Opcao:
```

Figure 18-jogo termina com sucesso

Depois de perceber que estava tudo certo com o terminar do jogo não foi iniciado novo jogo e foram feitos testes ás restantes funções do programa.

Teste á listagem do log:

- **Listar log geral:**

```
<2020-05-02> <21:43:27> <F17> <listou os utilizador por numero de jogos realizados de forma crescente com sucesso>
<2020-05-02> <21:43:25> <F17> <listou os utilizador por tempo de jogo de forma crescente com sucesso>
<2020-05-02> <21:43:23> <F17> <listou os utilizador por numero de vitorias de forma decrescente com sucesso>
<2020-05-02> <21:43:21> <F17> <listou os utilizador por ordem crescente com sucesso>
<2020-05-02> <21:42:51> <F17> <iniciou sessão>
Clique Enter Para Continuar...
```

Figure 19- Pequeno exemplo de listagem do log

- **Listar log de utilizador**

```
<2020-05-02> <21:43:27> <F17> <listou os utilizador por numero de jogos realizados de forma crescente com sucesso>
<2020-05-02> <21:43:25> <F17> <listou os utilizador por tempo de jogo de forma crescente com sucesso>
<2020-05-02> <21:43:23> <F17> <listou os utilizador por numero de vitorias de forma decrescente com sucesso>
<2020-05-02> <21:43:21> <F17> <listou os utilizador por ordem crescente com sucesso>
<2020-05-02> <21:42:51> <F17> <iniciou sessão>
Clique Enter Para Continuar...
```

Figure 20- pequeno exemplo de listagem do log de utilizador

No caso calhou que ambos os logs eram iguais, mas poderiam não ser!

Teste de algumas listagens escolhidas ao acaso (não foram colocadas todas porque apesar de serem apenas 4 tipos de listagens estas podem ser crescentes ou decrescentes e não achei necessário colocar testes ás 8 listagens).

- Listagem por ordem alfabética crescente:

```
----------ORDEM DE LISTAGEM------------
1- Crescente
2- Decrescente
3- Voltar

Opcao:
1
Jogador{nickname='2', numeroJogos=6, numeroVitorias=1, tempoJogo=2}
Jogador{nickname='ana', numeroJogos=0, numeroVitorias=0, tempoJogo=0}
Jogador{nickname='F', numeroJogos=0, numeroVitorias=0, tempoJogo=0}
Jogador{nickname='F17', numeroJogos=16, numeroVitorias=10, tempoJogo=1}
Jogador{nickname='Joao', numeroJogos=1, numeroVitorias=0, tempoJogo=29}
Jogador{nickname='novoNick', numeroJogos=0, numeroVitorias=0, tempoJogo=0}
Clique Enter Para Continuar...
```

Figure 21- Listagem ordem alfabética crescente

- Listagem por maior número de vitorias de forma decrescente:

```
Jogador{nickname='1', numeroJogos=7, numeroVitorias=3, tempoJogo=6}
Jogador{nickname='2', numeroJogos=6, numeroVitorias=1, tempoJogo=2}
Jogador{nickname='Joao', numeroJogos=1, numeroVitorias=0, tempoJogo=29}
Jogador{nickname='ana', numeroJogos=0, numeroVitorias=0, tempoJogo=0}
Jogador{nickname='F', numeroJogos=0, numeroVitorias=0, tempoJogo=0}
Jogador{nickname='novoNick', numeroJogos=0, numeroVitorias=0, tempoJogo=0}
```

Figure 22- listagem maior número de vitorias de forma decrescente

- Listagem por maior número de jogos de forma decrescente:

```
Jogador{nickname='1', numeroJogos=7, numeroVitorias=3, tempoJogo=6}
Jogador{nickname='2', numeroJogos=6, numeroVitorias=1, tempoJogo=2}
Jogador{nickname='Joao', numeroJogos=1, numeroVitorias=0, tempoJogo=29}
Jogador{nickname='ana', numeroJogos=0, numeroVitorias=0, tempoJogo=0}
Jogador{nickname='F', numeroJogos=0, numeroVitorias=0, tempoJogo=0}
Jogador{nickname='novoNick', numeroJogos=0, numeroVitorias=0, tempoJogo=0}
```

Figure 23- maior número de jogos por ordem decrescente

Agora e por fim passo a apresentar o teste feito ao sistema de simulação e apresentação das jogadas feitas em um jogo:

- Ver sequencia de jogadas:

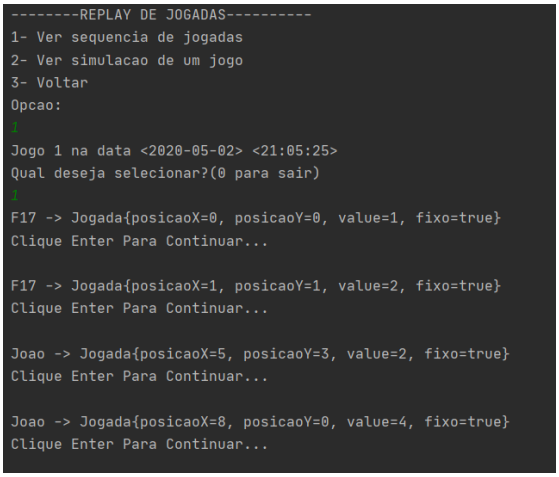

Figure 24- Ver sequencia de jogadas

- Ver simulação de um jogo

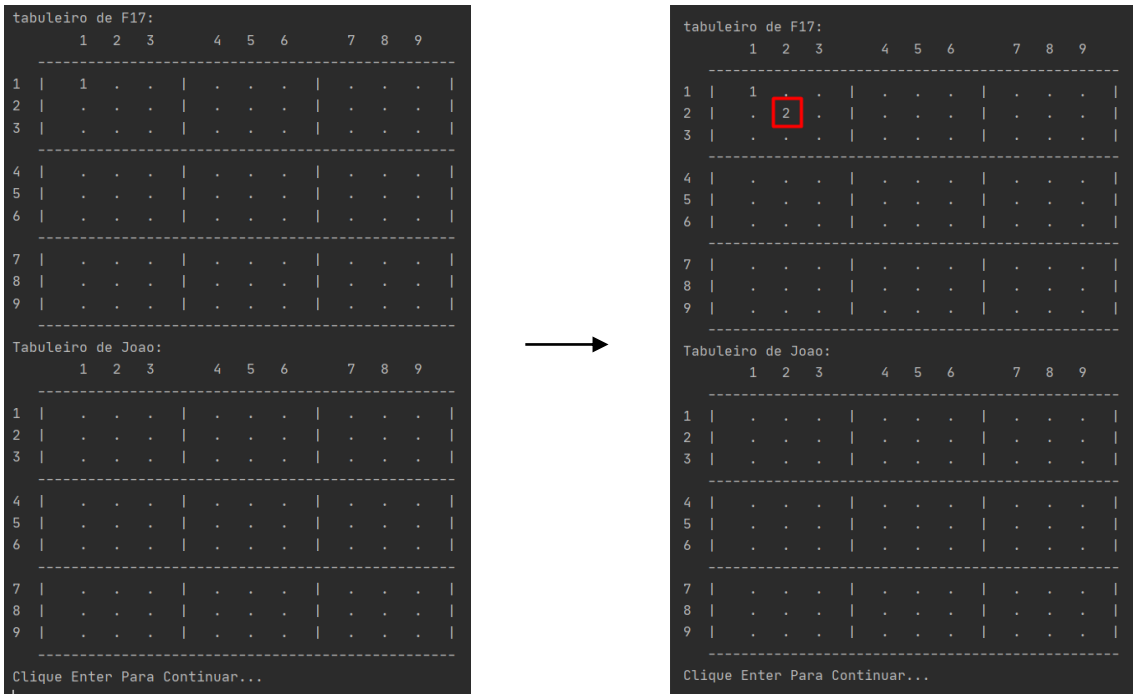

Figure 25-simulação de um jogo (2 primeiras simulações)

Com este teste finalizo assim todos os testes necessários a fazer á aplicação.

6 Conclusão

A realização deste trabalho apesar de demorada e muito extensa foi aprendizagem importante no contexto da linguagem Java e tudo que demos até agora com foco especial na comunicação em rede por sockets.

Consegui cumprir a grande maioria dos objetivos propostos sendo que a maior dificuldade que tive foi conseguir inicialmente compreender o funcionamento e como fazer bom uso dos sockets, depois da compreensão básica nesse tema o resto dos objetivos foram feitos sem grandes dificuldades.

Foi um trabalho muito importante e essencial no meu percurso académico.