Exercício de Avaliação III

Java Sockets API

Data de entrega: 3 Maio 2021

Sistemas Informáticos – MIEBIOM/MIEF
Departamento de Engenharia Informática

1 2 9 0
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D COIMBRA

Regras de submissão do trabalho:

- Este trabalho deve ser realizado por um **grupo de 2 alunos** e conta para a avaliação;
- Deve submeter um ficheiro **.zip** com o **código** (ficheiros de extensão .java) desenvolvido para **este exercício** e um **relatório em formato PDF** em https://inforestudante.uc.pt. **Não use outro formato de compressão.**
- O nome do ficheiro **.zip** deve seguir o formato: **nome1-nome2.zip;**
- O aluno que submeter o trabalho tem **de associar o seu colega** de trabalho durante o processo de submissão;
- Após submeter o trabalho em inforestudante, tem de registar o **esforço despendido na realização do trabalho por cada aluno em horas gastas em aula e em horas extra-aula.** Deve contar o tempo desde o início do semestre, caso esteja a submeter o exercício I; caso contrário conta apenas o tempo desde a última submissão de exercício de avaliação**.** Para o efeito preencha o formulário disponível em: https://forms.gle/M1ZwdYMme1EakHweA

Introdução

Neste exercício pretende-se desenvolver uma versão distribuída do jogo implementado no Exercício I – Torre de Hanói (com algumas alterações). Para o efeito, devem ser criadas duas aplicações (cliente e servidor) que comunicam de acordo com um **protocolo baseado em texto** e ***case-sensitive***. Para simplificar o exercício, e se concentrar nos métodos de envio/receção de informação em aplicações distribuídas, deve seguir as seguintes regras:

- Use o código distribuído com os exercícios de treino (diretoria *3-code-sockets*) como base para a resolução do problema.
- Não deve existir suporte para múltiplos clientes em simultâneo.
- **O cliente não deve efetuar qualquer validação dos dados a enviar para o servidor**. Isto é, assume-se que os dados estão sempre corretos em formato e tipo (podem, no entanto, conter valores incorretos). **A aplicação servidor (e apenas esta aplicação)** deverá verificar sempre os **valores dos dados** que lhe são enviados.
- Não deve fazer tratamento de exceções relacionadas com comunicação pela rede em qualquer das aplicações (a estudar nas últimas aulas). Assume-se que não existem erros na comunicação ou problemas associados a sincronização de eventos de comunicação.

A sua tarefa consistirá então em **implementar a versão distribuída do jogo criado no Exercício I**, o que inclui **definir o protocolo de comunicação, que deve ser baseado em texto,** a ser usado entre cliente e servidor. No contexto deste trabalho, **deve ser o servidor a manter todo o estado do sistema e a implementar as regas do jogo**. O cliente, para além de ser responsável por construir os pedidos a enviar ao servidor e processar as respostas, terá essencialmente o código responsável pela interação com o utilizador, o que inclui a **definição da interface** a usar com o utilizador e **apresentação de resultados** de operações.

O cliente é, na verdade, um *thin-client* sendo apenas um ponto de contacto fácil com o servidor.

Note ainda que o utilizador da aplicação cliente tipicamente não conhece o protocolo de comunicação. Assim, a aplicação cliente deve oferecer uma interação amigável e simples para que o utilizador possa realizar todas as operações necessárias e terminar o programa de forma fácil.

Descrição do trabalho

O servidor executa num ciclo infinito, aguardando por novas ligações TCP de clientes. Cada cliente que se liga ao servidor poderá então comunicar com este usando um **protocolo de comunicação baseado em texto**. A definição do protocolo deve ter em atenção as funcionalidades descritas nos tópicos seguintes.

- O cliente começa por estabelecer uma ligação com o servidor através do **envio de um login e password** em formato String. O servidor deve aceitar o cliente, **somente se as suas credenciais coincidirem com credenciais válidas e conhecidas por ele**. Caso as credenciais sejam inválidas, **a comunicação deve terminar**, e o programa cliente deve avisar o utilizador perguntando se quer tentar de novo: em caso afirmativo deve pedir de novo credenciais; e em caso negativo a aplicação cliente deve terminar.

- Em caso de credenciais válidas, deve ser possível iniciar um jogo. Tenha sempre em mente que **cada mensagem enviada pelo cliente deverá corresponder uma resposta do servidor** que poderá representar sucesso ou insucesso. Por exemplo, uma jogada bem-sucedida de um utilizador resultará numa mensagem de sucesso, enquanto que uma jogada inválida (ex: movimentos de discos inválidos) resultará numa mensagem de insucesso. O(s) código(s) que descrevem o(s) erro(s) devem ser **definidos pelos programadores** do jogo e mostrados de forma legível e detalhada do lado do cliente. Neste contexto, as mensagens de insucesso não terminam ou reiniciam a aplicação cliente, apenas fazem com que volte a pedir dados ao utilizador. O utilizador tem o poder de em qualquer instante terminar o jogo, sendo-lhe sempre dada a hipótese de começar um jogo novo, **mantendo-se a mesma ligação**. Caso não queira deve ser **terminada a ligação atual** e o programa cliente **deve dar a hipótese de outro utilizador se ligar através de um login/password**.

- Após o utilizador decidir terminar um jogo, ou após ter ganho um jogo, deve ser disponibilizado **um menu**, perguntando: se quer jogar de novo; se quer ver estatísticas simples do seu jogo, ou se quer terminar a sua sessão e dar a hipótese a outro utilizador. As estatísticas a apresentar são simplesmente a **média de movimentos e o número de jogadas por dificuldade (número de discos)**.

- A interação processa-se, em geral, da mesma forma que no Exercício I, porém todo o estado do sistema é mantido e controlado no servidor. Há, no entanto, algumas modificações a fazer:

    - Nesta versão, durante a configuração do jogo, para além do utilizador poder escolher o número de discos, também deve poder definir qual o pino onde devem estar todos os discos no início do jogo, e qual o pino onde devem estar todos os discos no final do jogo. Para tornar o jogo mais amigável, **o pino de destino deve ser indicado na representação gráfica do jogo**.
    - O número de discos deve ser um número entre 3 e 10 (inclusive).

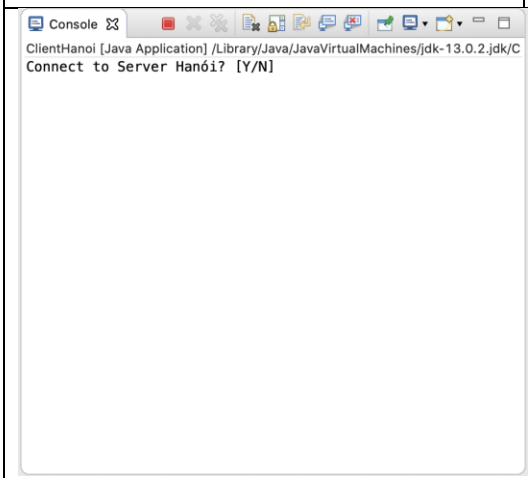
Relatório

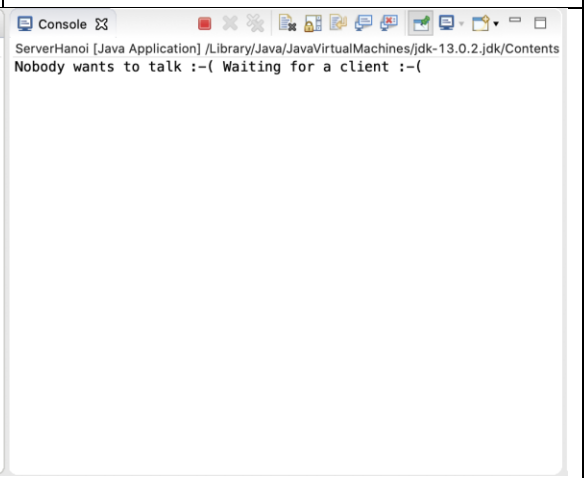
Deve entregar junto com o código (ver regras de submissão), um relatório curto em formato **PDF** que descreve o seu protocolo de comunicação (i.e., descreve o formato das mensagens, esclarecendo em que circunstancias são trocadas).

Exemplos

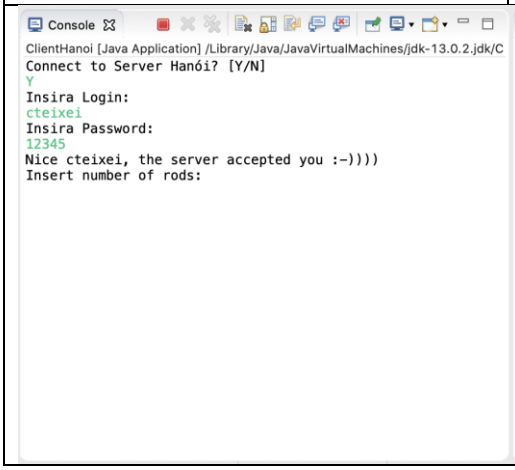
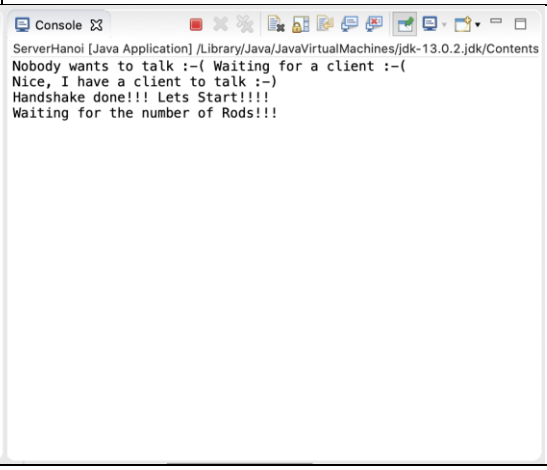
As imagens abaixo são só exemplos para clarificação. Não se espera que sejam reproduzidos exatamente, ou seja, é dada liberdade para a definição da representação gráfica que entender. Também de notar que os outputs do lado do Servidor servem somente para *debugging* e não é informação para o utilizador da aplicação, uma vez que o servidor pode nem estar a correr no seu computador e pode estar alojado noutro computador ligado em rede.

- Antes de qualquer tentativa de ligação:

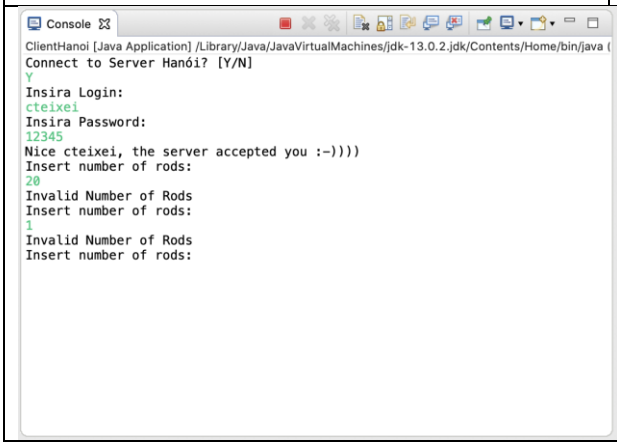
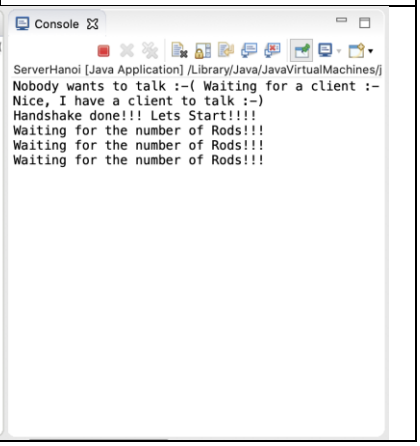
Cliente	Servidor
	



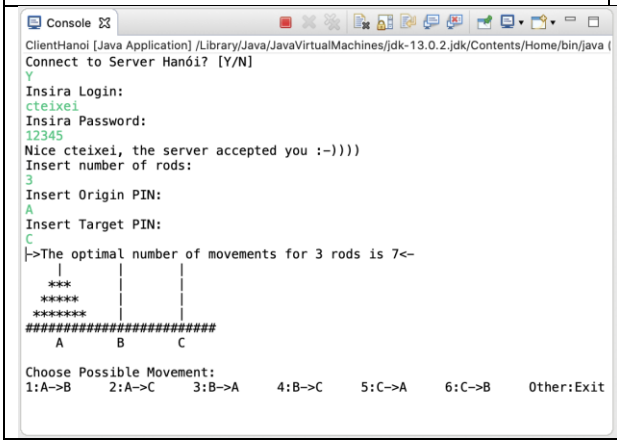
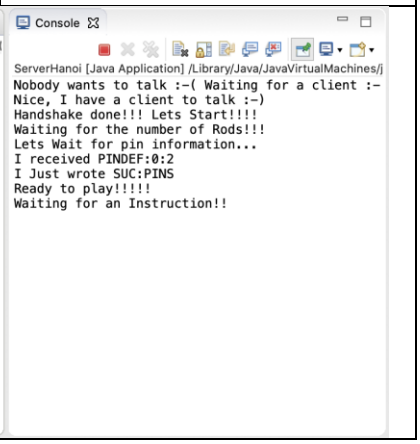
- Após Login bem-sucedido:

Cliente	Servidor
	

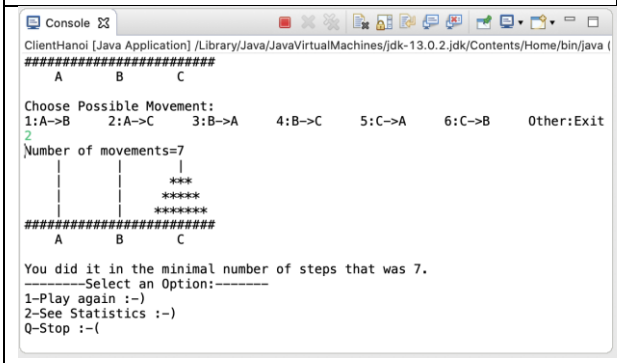
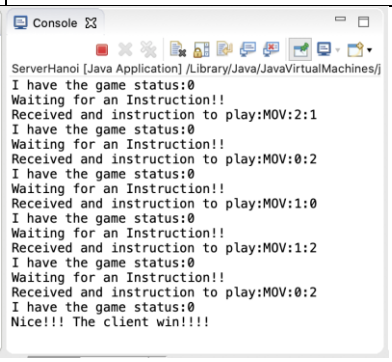
- Após problemas de configuração

Cliente	Servidor
	

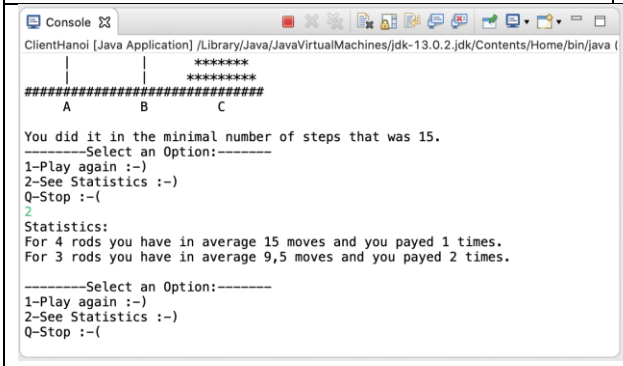
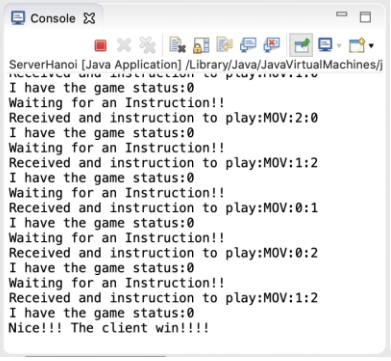
- Após configuração bem-sucedida

Cliente	Servidor
	

- Após finalização de um jogo

Cliente	Servidor
	

- Após o utilizador atual decidir ver as estatísticas

Cliente	Servidor
	

- Após o utilizador decidir abandonar um jogo e não fazer novo login, terminando o programa cliente. O Server continua à espera de novas ligações.

Cliente	Servidor