Exercício de Avaliação III

Java Sockets API

Data de entrega: 13 Maio 2019

Sistemas Informáticos – MIEBIOM/MIEF
Departamento de Engenharia Informática

• U  C •

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://goo.gl/forms/tEjK3vUSt0iU3AC33**

Introdução

Neste exercício pretende-se desenvolver uma versão distribuída do jogo implementado no Exercício I (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 a lógica de negócio**. 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 um menu 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 parágrafos seguintes.

O cliente começa por estabelecer uma ligação com o servidor. Após o *handshake* inicial deve ser possível interagir com o sistema, sendo que a 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 fora do tabuleiro 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 no 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 (os dados que originaram o problema).

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 (que num sistema real, serviria múltiplos jogadores em simultâneo oferecendo funcionalidades avançadas como, por exemplo, *chat* ou visualização de outros jogos a decorrer na altura). Há, no entanto, algumas modificações a fazer:

- É possível escolher um nome para o jogador.
- As jogadas de um dos jogadores são feitas pelo computador (as peças são colocadas em colunas aleatórias), pelo que haverá apenas um jogador humano.
- O jogador cliente escolhe as duas dimensões (altura e largura) do tabuleiro de jogo.
- O jogo passa a ser N-em-linha, sendo N escolhido pelo cliente.
- Há um sorteio inicial sobre que jogador deve iniciar o jogo (o humano ou o computador)
- Há um nível de dificuldade adicional, que o cliente poderá selecionar antes de iniciar um jogo, onde o computador representa dois jogadores (pode usar 'B' como cor adicional a 'R' e 'Y').
- Deve ser mantida uma tabela de registo dos vencedores mais rápidos (i.e., os que precisam de menos jogadas para vencer jogos), devendo ser possível consultar essa tabela quando não se está a jogar. Os jogos de dificuldade adicional devem valer o dobro da pontuação para o vencedor.

Quando o jogo termina (com um vencedor ou um empate), a aplicação cliente deve perguntar ao utilizador se deseja voltar ao estado inicial (onde seja possível fazer as operações pré-jogo, por exemplo, escolher o nome do jogador). Em caso afirmativo, repete-se o processo de interação com o servidor descrito anteriormente. Caso contrário (ou sempre que o utilizador usa a tecla `x`), a aplicação termina (com fecho dos recursos alocados). Este comando faz com que o servidor feche a ligação e se torne disponível para um novo cliente. Após o envio deste comando, o cliente fecha os recursos que alocou e termina.

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).