Universidade do Minho
Escola de Engenharia

Sistemas Distribuídos

2016/2017

Trabalho Prático

Gestor de Leilões

João Coelho
A74859

Luís Fernandes
A74748

Pedro Cunha
A73958

José Silva
A74601

Grupo 14

Introdução

É de conhecimento geral que os leilões são uma forma fácil e na sua maior parte das vezes, muito vantajosa a nível monetário, de vender artigos que sejam dispensáveis em casa dos seus proprietários, ou, por outro lado, adquirir produtos que se possam vir a manifestar úteis por preços mais reduzidos do que o que seria se estes fossem comprados em loja.

Podemos portanto afirmar que este método é útil e para tal é necessário um sistema que o vá gerir, podendo este sistema ser acedido por dois tipos de clientes: vendedores e compradores. O sistema funcionará de maneira a que um cliente que pretenda leiloar um artigo crie um novo leilão indicando uma breve descrição deste artigo, bem como o preço base para este. Será atribuído um número pelo sistema a cada novo leilão criado. Os compradores podem, por sua vez, a qualquer momento aceder a um leilão em curso e licitar o artigo, indicando o número do leilão e o valor que pretende pagar. Um vendedor pode também, a qualquer momento, encerrar um leilão em curso e o servidor declarará o respetivo vencedor desse leilão, bem como a quantia oferecida.

Classe SistemaLeiloes.java

Não entrando em detalhes sobre as variáveis e métodos da classe, vamos focar-nos numa explicação mais ao nível da concorrência e controlo da mesma.

Nesta classe utilizamos o *lock* implícito de forma a podermos aceder ao sistema sem que existam conflitos na escrita e leitura de dados, bem como um *lock* implícito para cada leilão criado. Para além destes *locks* são também usadas variáveis de condição para possibilitar a troca de mensagens entre os clientes.

Para que não hajam conflitos quando um cliente tenta aceder a um leilão, é necessário que primeiro seja requisitada exclusividade no sistema de forma a poder aceder a este sem que hajam perdas de informação ou erros a nível das leituras e escritas. Posto isto, é possível posteriormente aceder a um certo leilão em concreto e para isso se utiliza o *lock* desse leilão, podendo agora libertar o sistema para que outros clientes o possam utilizar e mantendo apenas aquele leilão em especial exclusivo para si.

Classe Server.java

O servidor é constituído por duas *threads*, uma com a função de enviar avisos do sistema e outra para responder a pedidos do cliente.

Existe no servidor uma classe auxiliar UserBool que permite registar se o cliente já efetuou o login, caso este processo ainda não tenha sido efetuado, as únicas operações disponíveis ao Utilizador serão fazer login ou fazer registo. Após efetuado o login no servidor, todas as restantes opções são desbloqueadas, ao passo que o Utilizador pode agora listar todos os leilões em curso, iniciar um novo leilão, licitar um artigo à venda ou, em caso de ter iniciado um leilão, pode terminá-lo.

Classe Client.java

O nosso cliente é composto por duas threads, sendo que uma delas efetua a receção de mensagens vindas do *standard input* (interação com o utilizador) e após a análise e respetiva validação do input envia esta para o servidor. A outra thread é, então, responsável por ler mensagens vindas do servidor e envia-as para o *standard output*.

Protocolo Cliente-Servidor

Pedido (Cliente)	Resposta (Servidor)	Descrição
r *username password*	sucesso	Registo efetuado com sucesso
	erro	Erro ao registar
l *username password*	sucesso	Início de sessão efetuado
	erro	Erro ao iniciar sessão
i *descrição valor_base*	iniciarLeilao x	Leilão criado com id x
ls		Listagem de todos os leilões em curso
lc *id_leilão valor*	sucesso	Licitação efetuado com sucesso
	valorReduzido	Valor inferior à maior licitação
	erro	Licitação não efetuada
f *id_leilão*	sucesso	Leilão terminado e vencedor anunciado
	erro	Leilão não terminado
t	sucesso	Sessão terminada com sucesso
	erro	Erro ao terminar sessão

Conclusão

Com este trabalho pudemos aprofundar os conhecimentos na criação de *Threads* e seu funcionamento, de modo a explorar a programação concorrente, bem como as várias técnicas para controlar esta mesma concorrência e evitar conflitos.

Todos os mecanismos como *Locks* ou variáveis de condição foram importantes para a resolução do problema apresentado de forma a garantir exclusividade na hora de escrita.

A exploração do modelo cliente-servidor e a comunicação via *Socket TCP* e servidor *multi-threaded* foi também abordado e no cômputo geral aprendido.

De salientar que houve uma tentativa de implementação do sistema de avisos para o cliente, mas sem sucesso, verificando que a forma de implementação pode não ter sido a mais correta, no entanto deixamos o código em comentário de forma a poder analisar melhor até à data de apresentação e tentar então explicar o que realmente correu mal, caso o consigamos detetar.