Instituto Superior de Engenharia de Lisboa

Mestrado em Engenharia Informática e de Computadores

Computação Distribuída

Semestre de Inverno 2020/2021

Laboratório #1

Docente: Luís Assunção

Realizado por: Grupo 6

- Edgar Alves nº 33017
- Paulo Pimenta nº 47972
- João Silva nº 42086

7.11.2020

Exercício 01

O objectivo deste exercício era criar uma aplicação cliente em Java RMI que permitesse ao utilizador enviar coordenadas para um servidor para encontrar 3 pérolas num rio (um array bidimensional com 35 posições).

Seguindo os slides dados nas aulas anteriores e as instruções dadas no enunciado, não existiram dificuldades neste exercício para a criação da aplicação cliente.

Obtivemos o *Registry* através do método *LocateRegistry.getRegistry* passando o *server ip* e o *registry port*, e, a partir desse *Registry* conseguimos fazer o *lookup* com o nome dado do enunciado ("*GameServer*") para conseguir chegar à referência do objecto remoto (*skeleton* que o servidor registou), onde propriamente fez-se o *cast* para *IPlayGame*, usando o contracto previamente estabelecido entre Cliente e Servidor que foi disponibilizado (*PlayGameContract.jar*). Finalmente, realizaram-se várias chamadas ao método *playGame* usando a referência do objecto remoto e passando um objecto *Bet* (que recebe duas coordenadas X e Y e um ID para identificar quem está a fazer a aposta). Este *playGame* retorna um objecto *Reply* que tem os campos *getNTries* (número de tentativas efectuadas), *isSuccess* (se houve sucesso) e *getThing* (que retornava se "pescámos" lixo, um peixe ou a pérola). Após várias tentativas, conseguimos encontrar as 3 pérolas nas posições X:2 Y:1 ; X:4 Y:2 e X:6 Y:0.

Exercício 02

Neste exercício era pedido que implementásse-mos uma aplicação cliente servidor onde o servidor é um agente de leilões. O contracto cliente servidor era disponibilizado (*ILeiloes*).

No servidor, usámos um *ArrayList<SomeObject>* para guardar os *SomeObjects* que estão a ser licitados e uma *Hashtable<String, List<INotification>>* para guardar a lista de *callbacks* e para emparelha-los a uma key, que é o *id* do objecto que está a ser licitado. No método *initLeilao*, estamos a adicionar o objecto que é passado por parametro no *ArrayList<SomeObject>* e depois estamos a criar uma nova *ArrayList<INotification>* que irá para o *Hashtable* em que o primeiro *callback* dessa lista é o *callback* passado por parametro (ou seja, o *callback* do criador do leilão). No método *getAllLeiloes*, estamos a retornar um *array* de *SomeObject* com os objectos todos que estão em leilão. No método *licitar*, adiciomos o *callback* passado por parametro à lista de *callbacks* do objecto com o *id* passado por parametro. Depois, é feita uma verificação se a licitação feita é superior à que já se encontra no campo *value* do objecto. Se for, o valor é actualizado e é passada uma mensagem para todos os *callbacks* desse objecto.

No cliente, foi usada uma estrutura idêntica à do exercício 1, extendendo algumas funcionalidades para que quando o cliente é corrido possa haver a opção para o utilizador se quizer licitar ou criar um leilão.

Neste exercício sentimos alguma dificuldade na passagem dos *callbacks* (*INotification*). Começámos por passar uma instância duma *nested class* que implementava *INotification*, mas acabava por fazer com que a consola do servidor apresentasse a notificação em vez da consola do cliente. Após verificação dos slides das aulas anteriores e alguma pesquisa, chegámos á conclusão que a própria classe *Client* é que tinha que implementar *INotification*, e que depois era então passada no método *licitar* uma instância dessa classe *Client.*

Sobre o resto da implementação, usámos o que aprendemos no exercício 1 sem mais dificuldades.