CS502052: Enterprise Systems Development Concepts

Lab 4: Java - Remote Method Invocation (Part 2)

I. Introduction

Java Remote Method Invocation (RMI) enables the programmer to write distributed Java programs. RMI makes it possible for Java programs to invoke methods of (remote) objects that possibly reside on a different (remote) computer.

The architecture that makes RMI possible consists of three parties: (1) a client, (2) a server, and (3) a naming service (***rmiregistry***). The client is the process that invokes a method on a remote object. In contrast, the server is the process that owns the remote object. This remote object is just another ordinary object in the address space of the server. Finally, the ***rmiregistry*** is a name server that has a table that maps objects to remote references. To clarify, servers that want to provide RMI services register their remote objects at the ***rmiregistry***. In addition, clients that look for a specific remote object can contact the ***rmiregistry*** to get a remote reference for the required remote object.

II. Exercises

This program will be a room reservation system using RMI. For this exercise, you can assume that everybody is reserving the room for a specific period. In other words, our system does not give the guests the liberty to choose the dates for the reservation. There are **5** room types with different prices:

- We have 10 rooms of type 0 which are single rooms that costs 55 Euros a night;
- We have 20 rooms of type 1 which are double rooms that costs 75 Euros a night;
- We have 5 rooms of type 2 which are twin rooms that costs 80 Euros a night;
- We have 3 rooms of type 3 which are triple rooms that costs 150 Euros a night;
- We have 2 rooms of type 4 which are quad rooms that costs 230 Euros a night.

To implement this system, you will have to write a server ***HotelServer***, a client ***HotelClient***, a remote interface ***RoomManager***, and the implementation of the remote interface ***RoomManagerImpl***.

The hotel client ***HotelClient*** can be run by entering the following commands:

- **java HotelClient host port [help]** - If no options are supplied, ***HotelClient*** just prints and shows how this command can be used.
- **java HotelClient host port list** - List available number of rooms in each price range. The output should look like the following[1]:

```
+  v rooms of type 0 are available for 55 Euros per night
+  w rooms of type 1 are available for 75 Euros per night
```

[1] The values of *v*, *w*, *x*, *y*, and *z* should be the current number of available rooms for each type of room

```
+  x rooms of type 2 are available for 80 Euros per night
+  y rooms of type 3 are available for 150 Euros per night
+  z rooms of type 4 are available for 230 Euros per night
```

- **java HotelClient host port book <room_type> <guest_name>**: books a room of the specified type (if available), and registers guest's name.
- **java HotelClient host port guests**: list the names of all registered guests.

For the *list*, *book*, and *guests* options, the ***HotelClient*** remotely invokes associated methods in a remote object. The ***HotelClient*** gets the results of these remote methods back from the ***HotelServer***, and prints them out on its standard output.

Before implementing this program, you should analysis the system, especially functions and classes.