University of Leeds
School of Computing
COMP3900 Distributed Systems

Artur Oliveira Rodrigues
SID: 200622082

Coursework 2

All the code of this coursework is available as a public GIT repository at: https://github.com/arturhoo/ds_cw2

Question 1:

Submitted online (tagged as 1.1)/not relevant for this document.

Question 2:

The server program in the file CityServer.java is a simple implementation of a RMI server that expects one argument, which will be the name of the City, represented by the object. This object is then registered and the user is informed of the process. Boilerplate error handling is also present in the file.

No files were submitted online as no changes were made to the server.

Question 3:

The code was submitted online (tagged as 2.1). What was done was the implementation of the client. It was based on the Bank example of Lab Sessions week 4. It basically locates the registry based on the specified host, and then looks for the City object passed as argument. It then updates the objects private fields, country, minimum temperature and maximum temperature through the public setter functions. Reports are printed on the screen in order to inform the user.

Question 4:

Submitted online (tagged as 3.3)/not relevant for this document. See CityClient2.java

Question 5:

Using RMI to implement the City Server proved itself to be very straightforward and I was able to put everything up and running with minimum effort. As the RMI architecture operates on top of sockets, it is on a higher level. Instead of having to explicitly send bytes through the network I only dealt with Java objects and its methods invocations. It is important to note that some overhead is expected and sockets should be the choice for ultimate performance.

Question 6:

The code was submitted online (tagged as 4.1). For this task it was necessary to query the Database to define the attributes for the City object the Server was initiated with. Therefore, the client from Question 4 could be used as soon as the server is running, as the client from Question 3 is not needed anymore. The information is queried from the Database.

The changes necessary were basically boilerplate regarding JDBC connectivity, building an statement based on the city passed as argument to the server, and querying the DB for the information about the city in question. This way, the remote object had its attributes ready. It is important to note that it was not implemented a way to identify cities which are not in the DB.

It is also worth mentioning that as we don't have writing permissions on DB the client from Question 3 only overwrites the object's attributes and not the information stored on the DB.

Question 7

The code was submitted online (tagged as 5.2). For this question was implemented a Factory that basically has a method to retrieve information from a city based on its name from the database. Then, this instance of City is returned and the client can show its info. See CityClient2.java