Task 3: a DNS client
SKJ (2016)

The aim of the task is to write a cient of the DNS protocol. The author should write a single application, which will connect with a DNS server given as a parameter and then ask it for an `ANY` record for its second parameter (so all the data related to the parameter, which are stored on the server). We assume, that the client **does not work** according to the iterative scheme, but it must properly interact with the server according to the DNS protocol and accet all possible types of server's response, which include in particular:

- The standard connection uses the UDP protocol to send a query to the port 53 of the server. We assume, that the query includes a recursive query request. If the serwer responds with the answer, in which the TC flag is **not** set, the query results should be shown on the screen.
- If the server's response has the TC flag set (which means, that the response was to long to fit within a single UDP packet), the client should reconnect with TCP protocol to the server's port 53 and resend the query. After the response is received, the query results should be shown on the screen.

The client program requires the following parameters to be accepted (their order is not obligatory but they both have to appear):

- `--server` *server address*
  denotes the server address, to which the client should connect to,
- `--query` *query string*
  denotes the data, which constitute the query contents.

Grading and Assessment

1. For full and correct implementation of the task a student can obtain up to **4 points** (plus 1 extra point):
   (a) implementation of the part, which allows to generate and perform UDP queries to the server is worth **up to 3 points**,

(b) implementation of the part, which allows to generate and perform TCP queries to the server is worth **1 point**,

(c) implementation of additional functionality, which allows to generate and perform queries for any other kind of record, other than `ANY` is worth **1 extra point** (under assumption that both previous requirements have also been poperly implemented).

2. The application must be written in JAVA, according to Java8 standard (compatible with JDK 1.8). For network communication only standard UDP sockets (DatagramPacket, DatagramSocket classes) and TCP sockets (SeverSocket, Socket classes) can be used. In addition, any components of Java standard library can be used. In particular, GUI can be built using Swing or JavaFX. Note that GUI is of secondary importance in this task.

3. Solutions should be saved to proper directories in the EDUX system until 21.I.2017.

4. The project archive should include:

   - The source files (for JDK 1.8) (together with all the used additional libraries, that do not belong to standard Java instalation) — the application must easily compile on the computers at PJA labs,
   - a *readme.txt* file with author's notes (what was implemented, what not, where are the errors that the author did not manage to correct and so on).

5. The teachers will firstly rate the correctness of the solution and the degree of satisfaction of the security requirements, but also also the conformance to the good rules of the software design and implementation quality (for instance, "everything in main()" is not a good idea).

6. IF NOT DETAILED, ALL UNCERTENTIES SHOULD BE DISCUSSED WITH THE TEACHER. OTHERWISE, THE SOLUTION MAY BE NOT ACCEPTED, IN CASE OF WRONG SELF-INTERPRETATION.

Additional resources

The task requires the student to learn about the DNS protocol.

1. RFC 1034: `https://tools.ietf.org/html/rfc1034`

2. Wikipedia: `https://en.wikipedia.org/wiki/Domain_Name_System`