Project Task: Design a directory for handling of person names, phone numbers and e-mail addresses

A directory is a hierarchically organized database optimized to enable fast and efficient reading and searching for information but with lower requirements on fast updates of such information. It is a common tool used to logically organize information of various kinds so that you easy can look up what you search for. You have almost for certain already been exposed to many different directories; for example a telephone catalogue. You can also think about your mobile phone or laptop computer where you via its operating system support or web-browser have access to different catalogues or registers where you handle hardware device parameters, person names, phone numbers and e-mail addresses etc.

A directory can be seen as a server indented to be used by two kinds of clients:

1) the ordinary users of the directory
2) the administrator of the directory

Ordinary users are typically only allowed to perform a restricted set of operations, while the administrators have full authority to apply all operations possible on the directory. There may in a practical system application be a need also for other client categories but you can limit the scoop to these two.

The data that we store in a directory is information that can be handled in well organized and logical information structures. To understand those structures we need a model or a so called schema as a guide over the information types and structures. Such a model is in LDAP context called a directory information tree (DIT). In object oriented terminology you can make a design of a directory built on the use of class inheritance and class composition structures defining the classes and attribute types involved and how they are related in structures.

Directories are typically organized hierarchically in tree like structures. The nodes in such a data tree are objects distinguished by their names and attribute values. For identification of an object in such a tree, in a safe and sure way, different forms of unique names or identifiers can be used. Examples of such unique identifiers are domain names, telephone numbers (complete with area and country code) but also automatically generated ID numbers such as Universally Unique Identifier (UUID) – a technique commonly used to identify objects, messages and sessions in a distributed system without the need for synchronization or other coordination to avoid naming conflicts.

Since your studies are related to computer networks and communication protocols you are in this project task supposed to use terminology defined in the light directory access protocol (LDAP) aimed for access to network distributed directories. By studying what LDAP is and can do, you can somewhat indirectly get an understanding what a directory is. To further complement your understanding, what a directory is about, you may to take

a look at the documentation of an open source implementation of a directory server implemented in Java, for example OpenDS. Another source to more information is the Java Naming and Directory Interface (JNDI) made for applications written in Java.

In network distributed applications the operations to be performed on a directory must be possible to do remotely from any of the above kinds of clients by sending of messages to the server over a computer network. It is for this purpose that LDAP has been designed, the current version is called v3 and is defined in an RFC called 2251 and its update 4511 (see also 4510 for an overview of related protocols). To get information about LDAP and its development history you can also read Wikipedia or use Google to find other popular descriptions of LDAP where you also can find information about directory structures in general.

A directory intended to be accessed by use of the LDAP protocol must have some specific properties fulfilled. For example, all entries stored in the directory must have a unique "Distinguished Name" (DN). Such a DN is composed of two parts: the “Relative Distinguished Name” (RDN) and its location within the LDAP directory, i.e. where it is placed in the hierarchical, directory tree, structure. The DN can be compared to and has a similar role as the complete path name to a file. The RDN is thus like a file name, meaning that you need more information to access the file.

Each actual operation made to an LDAP directory, doing something with the data in the directory, must in a real network scenario, be part of a session starting with the setup of a secure TLS connection, using the operation *StartTLS* over which the client also is authenticated using the operation *Bind*. Then when a secure session is established “ordinary” data operations such as *Search, Compare, Add, Modify, Delete, Abandon,* etc can be made. The different operation messages, including also parameters, need to be associated with a unique message ID since answers are allowed to be delayed and can come in any order from the server or chain of servers involved in answering. The secure connection, embedded and carried by a TCP/IP session, is finally turned down, when no more messages need to be exchanged, by use of an *Unbind* operation.

Your mandatory, core, task is to make an implementation of a simple directory limited to its data handling parts that can be accessed and operated upon via the two types of clients mentioned above. To test the function of the directory as a server it can useful to implement a simple version of the clients and especially their API:s as well.

To be able to solve this task and get full credit you:

1. Gather information about directories and directory structures that can be accessed via LDAP. Create a requirement specification that tells what you need to fulfill.
2. Make an object oriented directory design that is detailed enough to work as an implementation proposal. From this design proposal it shall be possible to divide and share the implementation work in your project group.

3. Create implementation classes and a main program that builds an empty directory that latter can be filed with directory entries (objects).
4. Test the result using a set of operations that covers all functionality. Start with operations that create entries in the directory and then do different search, read, compare and update operations.
5. Explain and demonstrate your design, its implementation and its function by use of test cases, and be ready to modify these tests if asked for, to the examiner.

The project task has thus a mandatory core part, as defined above, that all groups must do including the functional requirements on actual directory reading and updating related operations. An ambitious group can however do extra work on one of the possible extensions that are listed below:

1. Data structures and algorithms that gives good search performance.
2. Client tools (graphical or textual) providing benefits in terms of simplified use or better overview to the ordinary user and/or the administrator.
3. Make the directory “network attached” by adding LDAP functions for setup, binding and closure of secure sessions over a network etc.
4. Provide a RMI based remote access to the directory (somewhat simpler than 3).
5. Make your solution compliant with JNDI which is an API that provides standardized naming and directory functionality to applications written in Java.