# Assignment 3
**22.10.2014**

LDAP

## Introduction:

LDAP stands for *Lightweight Directory Access Protocol*. As its name suggests, this protocol provides access to directories and it is widely used (eg: as address book repositories or authentication directories). Each object stored in the directory is created on the basis of a schema, which defines the attributes of the object. The directory is hierarchical, similar to conventional file systems (i.e.: trees).

## Purpose:

The goal of this assignment is to code (in your language of choice) a rudimentary client that can connect to an LDAP server, display the directory contents, and add/remove/modify entries from it.

For the duration of this assignment, an LDAP server will be available for testing your application (using simple authentication):

Host: *54.68.0.145*

DN (login): *cn=admin,dc=security,dc=ch*
Password: *security20142014!*

*(Warning – you are given **admin** rights to the LDAP server for *the sake of testing*. Please do not remove any existing data. I set up appropriate logging facilities to detect any possible misuse. The system is advanced enough to detect who is misusing it and deduct assignment points. Maybe.)*

## LDAP notes:

Each entity in the LDAP directory is identified by their Distinguished Name (*dn*), it is a unique identifier that also identifies where in the hierarchy the object belongs:

dc=ch
dc=security
ou=tutor
cn=Yarco Hayduk
cn=John Doe
cn=Herve Sanglard
For instance, the *dn* for the Tutor *Hayduk* would be described as:

*dn: cn=Yarco Hayduk, ou=tutor, dc= security, dc=ch*

And for John:

*dn: cn=John Doe, ou=students, dc=security, dc=ch*

For this assignment you'll be working with the organizational unit (ou) students.

Each entity can have a different number of attributes: some obligatory, others optional. The attributes are determined by the scheme, and each scheme specifies a particular *objectClass* (an attribute can also hold a multiple values).

For instance, the "*person*" objectClass is defined as:

*olcObjectClasses: ( 2.5.6.6 NAME 'person'*
*DESC 'RFC2256: a person'*
*SUP top STRUCTURAL*
*MUST ( sn $ cn )*
*MAY ( userPassword $ telephoneNumber $ seeAlso $ description ) )*

LDAP uses the *LDIF* format to print objects in plain text. For instance, the *Hayduk* entity is defined as:

*dn: cn=Yarco Hayduk,ou=tutor,dc=security,dc=ch*
*objectClass: person*
*cn: Yarco Hayduk*
*sn: Yarco*
*description: Course tutor*

## References:

LDAP: http://www.cru.fr/documentation/ldap/index

Introduction to LDAP:
http://quark.humbug.org.au/publications/ldap/intro_ldap/index.htm

JNDI Tutorial (LDAP access for Java):
http://download.oracle.com/javase/jndi/tutorial/trailmap.html

JNDI and OpenLDAP tutorial:
http://www.concentric.net/~adhawan/tutorial/

A small note if using JNDI: The bind method (to insert entries) has an Object parameter which can be used to store Java objects. This isn't needed for our purposes. We need only add the list of attributes:

```
students.bind("cn=John Smith", null, attrs);
```

## Hand-In:

A demonstration of the software. The source files and project files must be submitted to the appropriate Ilias assignment.

## Deadline:

One week (29.10.2014), before class (14:00).