Lab 1: Java API Essentials

Lab objectives:

- Refresh knowledge of Java;
- Get familiar with the best coding practices;
- Correct coding style if needed;

JDK

For the labs: the only applicable JDK will be 1.5.0 (you can download prepared package from http://www.zaval.org). It should be installed to: `c:\jdk`

Environment variables

Setup the following environment variables:

```
JAVA_HOME = c:\jdk
CLASSPATH = .;%JAVA_HOME%\lib;%JAVA_HOME%\jre\lib;
PATH = %JAVA_HOME%\bin;%PATH%;
```

Note: save previously set variables (if any) and restore them at the end of the lesson.

Tasks Specification

Note: Each task should be in package `it.unitn.web.`**`NAME`**, where **NAME** is corresponding task's name (streams, threads, etc).

I/O Streams (30 minutes)

Task objectives:

- Refresh the principles of streams usage in Java;
- Understand the difference between Streams and Reader/Writer objects;

Task description:

- On input you have string "Department of Information and Communication Technology";
- Compress the string utilizing GZIP compression;
- Uncompress the string;
- Make sure both strings are identical;

Hint: make chains GZIPOutputStream->ByteArrayOutputStream and ByteArrayInputStream->GZIPInputStream.

Threads (30 minutes)

Task objectives:

- Refresh the fundamentals of threads;
- Refresh threads' synchronization basics;

Task description:

- Scan TCP ports from **X** to **Y** on specified computer within single thread;
- Scan the same ports in parallel using **N** threads;
- Log scan results to file (using separate thread and a queue);
- Check and report time spent in each case for scanning.

Notes:

- *Scan* means check whether port is open or not;
- **N** is less or equal to **Y**-**X**;
- **N, X** and **Y** can be specified as input parameters.

Serialization (20 minutes)

Task objectives:

- Refresh the basics of serialization;

Task description:

- Make *Hashtable* object and fulfill it with 3-5 objects of type *String*;
- Save it to disk;
- Restore it from disk to another Hashtable object;
- Make sure initial and restored object are identical;
- Make custom object with fields of type *String*, *int []* and *long* (primitive types, **not** *Integer*/*Long*);
- Populate it with values and save to disk:
  - Using *ObjectOutputStream*;
  - Using *DataOutputStream*;
- Restore from disk;
- Make sure object is identical to the original.

Protocols (1 hour)

Task objectives:

- Understand how to organize simple communication utilizing specified exchange protocol;
- Refresh networking API;

Task description:

- Run server part for this task;
- Connect to server part with telnet (TCP port 2700 on **localhost**);
- Use **help** command;
- Login with login=admin and password=web;
- Request info about file **test.txt**;
- Retrieve file **test.txt**;
- Logout;
- Write simple client that does steps described above without human intervention (file should be saved to the location specified).

Protocol:

- Each command should end with new line character (“\n”);
- Command execution status can be verified by server response – in case of success it will return “OK”, in case of fail - “FAILED”, and “COMMAND_UNKNOWN” in case of unknown command.
- **Important Note:** LOGOUT and EXIT commands disconnects client without returning any value.

States diagram (states and state changes):

NOTAUTHORISED ->AUTHORISED (after successful LOGIN)
->NOTAUTHORISED (otherwise)
AUTHORISED ->NOTAUTHORISED (after LOGOUT or EXIT)
->AUTHORISED (otherwise)

Reflection API (optional) (20 minutes)

Task objectives:

- Refresh reflection API basics;

Task description:

- Create method that accept any object of class Object;

- Depending on the actual class of object that can be Integer, String or any other it returns values 0, 1 or 2 respectively;
- It prints actual class name of the object to the output;
- Create method that accepts class name as string;
- It prints out all constructors for this class of object;
- It tries to create the object of specified type in case there is empty constructor available and does nothing otherwise;
- Create methods that check if the specified method and field exit in the object of indicated type.