# Object Oriented Programming, 2014/04/03, 15:30
# Laboratory evaluation - Duration: 45 min

| | | | |
|---|---|---|---|
| Machine name | ______________ | | |
| Student number | ________ | Student name | ______________________________ |

You must:

- Provide proof of your identity.
- Only have on your desk your identification and a pen.
- Ensure mobile phones or any other electronic devices are switched off and placed with personal belongings apart from the desk. Any student who does not switch off its phone, or who retains one in its possession, will be disqualified.
- Do not attempt to use the internet. The internet in your computer is off.
- Do not attempt to use any USB pen, to put/take info in/from your computer.
- **Do not turn this evaluation sheet before instructed**.

Examination instruction:

- The evaluation consists in a program to be developed within a package named

  lab1_NUMBER

  where NUMBER is the student number (e.g.: student number 12345 should create a package named lab1_12345 and provide java classes inside this package). This package must be found inside an Eclipse project named `LAB1`. Before stating ensures everything is in place (`/home/poo/workspace/LAB1/src/lab1_NUMBER`).
- When the evaluation ends leave the computer on and in session, that is,

  **DO NOT LOGOUT**,

  and **leave the examination sheet on the desk**.
- Skim all the evaluation before beginning to best plan your time.
- If the program does not compile, or crashes, it will be evaluated to a maximum of 1.0 points (out of 2.0 points).

The program should implement the following functionalities:

[0.1] • Create a package named lab1_NUMBER where NUMBER is your student number. Add three public classes to this package named `Lab1`, `Lab2` and `Main`.

[0.1] • Add a `main` method in the `Main` class. It receives as parameter an integer `n` greater than 1. If the integer received is not in this condition terminate the program with the message "Input number must be greater than 1"and status code 1.

[0.2] • Provide a directed association from `Lab2` to `Lab1`; the multiplicity of this association is 0..10, with role `labs1`. This association should be visible only inside the class where it is defined. Do not provide any constructor in `Lab2`.

[0.2] • In `Lab1`, add two instance fields of type integer, named `x` and `y`, with package visibility. Provide a public two-arg constructor to correctly initialize objects of type `Lab1`.

[0.2] • Provide equivalence between objects of type `Lab1` according solely to the values of `x`. Redefine any other method required by Java in this context.

[0.2] • In `Lab2` provide an instance method, with visibility package, named `associateLab1`, that receives an object of type `Lab1` as parameter and returns a boolean. If (1) there is still less than 10 objects associated with the corresponding `Lab2` object and (2) the `Lab1` object received as argument is not equivalent to any other `Lab1` object already associated with the corresponding `Lab2` object, establish the association and return true. If only (2) fails return false. If (1) fails print to the terminal "Trying to associate more than 10 objects!" and terminate the program with status code 2.

[0.1] • Provide a textual description of `Lab1` as "Lab1[x = NUM1, y = NUM2]", where NUM1 is the value of `x` and NUM2 is the value of `y` in the corresponding object.

[0.1] • Provide a textual description of `Lab2` as "Lab2[labs1 = ARRAY]", where ARRAY is the string with the contents of the array stored in the reference `labs1`.

[0.1] • In the `main` method create an array, named `labs2`, to store `n` objects of type `Lab2`. Provide an infinite loop where:

[0.3] (1) `n` objets of type `Lab2` are built and stored in `labs2`. Build `n`×10 `Lab1` objects passing to the `Lab1` constructor a random value between 0 and 9 (use the method `nextInt(int)` from class `Random`) to store in `x` and 0 to store in `y`. Attempt to establish associations (via `associateLab1` method) between them, that is, the 1st `Lab2` object is associated with the first 10 `Lab1` objects, the 2nd `Lab2` object is associated with the second 10 `Lab1` objects, and so on.

[0.2] (2) In the end of the previous process, if there is a `Lab2` object with 10 `Lab1` objects associated with it print to the terminal "Found it: LAB2 after NUM iteration(s)", where LAB2 is the textual description of the `Lab2` object and NUM is number of iterations needed to achieve it, and terminate the program with status code 3. Otherwise, continue doing (1) and (2) until finding it.

[0.2] • Provide an executable .jar file of your application with sources and place it in the following directory: `/home/poo/workspace/LAB1/src`.