Competence Test - Programmer

Task 1

Given the Java example code snippet (file Task1.java) please modify it to better adhere to Object Oriented Programming principles. This is so called code refactoring task, please apply as many code refactorings as you can.
The result should be modified file Task1.java that compiles and is functionally the same as the attached example code.
Download Test1.java from:
Task1_v2.java

Task 2

Write a short (console) program in C, C++ or Java that will allow defining a troop and locating it on a virtual battlefield. The program design should follow as closely as possible and practicable the Object Oriented Programming methodology (even if it is written in C). The virtual battlefield should have a name (any text) and contain any number of the following units:

- soldier,
- tank

Each unit has its own strength and location on the battlefield. Tank in addition has an attribute which tells how many soldiers are in the tank (occupancy).

a. Initialize the battlefield with "Sample" as a name and the following participants:

- 2 soldiers with strength of 10,
- 2 soldiers with strength of 5,
- 1 tank with strength of 100 and occupancy of 3

Place soldiers in different arbitrary locations within the battlefield. Place the tank in the center of the square
b. Add XML export functionality and dump the XML to stdout, the XML output should look like this:

```
<battlefields>
  <battlefield name="Sample">
    <soldier x="100" y="100" strength="10"/> <tank x="100" y="110"
strength="100" occupancy=”3”/> etc...
  </battlefield>
</battlefields>
```

c. Move each soldier by vector +10,+15 (x,y) and the tank by vector -10,-15 (x,y) and dump XML export to stdout
d. Calculate (and print) the total strength of the troop.
e. Calculate (and print) the bounding box surrounding all units of the troop (bounding box:min x, min y, max x, max y)
f. Calculate (and print) the total number of soldiers (including those in the tank indicated by occupancy attribute of the tank)
Note: Please try to use the same coding conventions as it was used in the example code snippet in the Task 1.
The result should be a zip file (Task2.zip) containing all source code and a script(s) (Makefile or Ant) that allows compiling the application on Linux (for C/C++ please assume that gcc will be used to compile the application).

Task 3

Given a database with tables washing machines and load sheet (structure and example data below) please:
a. Write an SQL query that will list all washing machines (model and serial nr) ordered descending by model
b. Write an SQL query that will show those washing machines that loaded in total more than 10 kgs
c. Write an SQL query that will show how many kgs each washing machine has loaded (the output should be 3 rows!)

Table washing_machines_<your initials>

id (primary key)	model	serial_nr
1	Whirlpool	111
2	Bosh	222
3	Zanussi	3

Table load_<your initials>

wm_id (foreign key to washing_machines.id)	t_date	load
1	2012-10-01	4
2	2012-10-02	2
1	2012-10-03	11

The result should be a capture log from MySQL with these 3 SQL statements and the result of their execution (using the example data above). Alternatively, if you have no direct access to MySQL DB server you can also run all your queries online here:
goo.gl/LCB6v

Task 4

Please enhance the example Lua script that currently counts the number of words in given text to also calculate and display the sum of all numbers (total number of animals) and displays animals list as „All animals: psy,koty,krowa,konie”.

```
text = "2 psy 3 koty 1 krowa 4 konie"
wordCount = 0
for word in string.gmatch(text, "%w+") do
      wordCount = wordCount + 1
end
print("Words in text: " .. wordCount)
```

The script can be tested here: http://www.lua.org/cgi-bin/demo
The result should be a Task4.lua file containing the written Lua script (including the example from the above).
Hint 1: Lua has a global function called tonumber that converts its argument (could be text) to a number
Hint 2: Lua string.gsub function replaces all occurrences of given pattern.