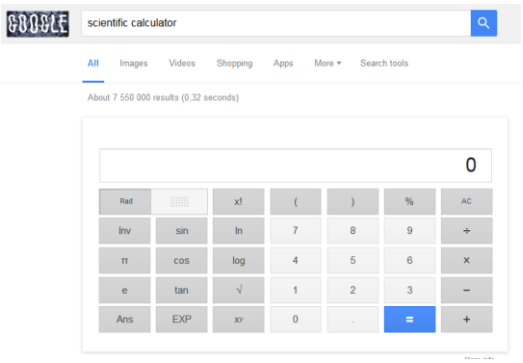
Rafael Silva – rafael.silva@fareoffice.com
Date:2017-12-05
Version: 1.0
Document: 027 – Final project – 171205
Deadline: Saturday, 2017-12-16, 11.55pm.

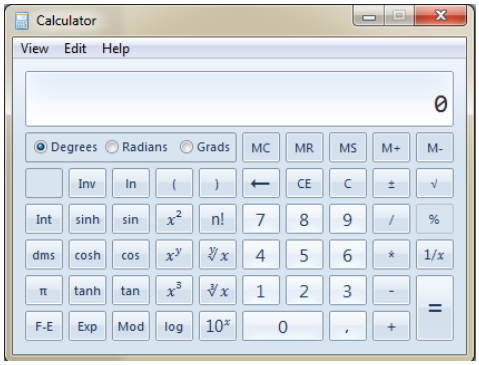
Final project:

Java OOP, UML, Git, GUI, Javadoc, tests, Logs, Test reports

The program that you will implement is a scientific calculator with a graphical user interface. The calculator should perform, at least, 10 different operations. The idea is that you work with very poor written requirements so that you think about the design and the implementation of the program yourself.

Either design your own GUI or use as example, any scientific calculator that you find. E.g. :





Rules for submission:

- Your project and your remote repository should be named following the standard below:
  - E.g. “calculator-rafaels” (in the case, Rafael Silva worked alone)
  - E.g. “calculator-rafaels-andersh”. (in this case Rafael Silva and Anders Holm worked together)

NOTE: If you work in groups of two, it is a requirement that you will use git collaboration!!!

- The only thing you should upload to the student portal is a link to your remote repository on GitHub. Your repository should include a maven project with source code and tests, Javadoc, a class diagram and the test reports of your project. In case you work in pair with somebody, both students should upload the solution, even if the URL is to the exact same repository that you collaborated on.

Test4Quality NACKADEMIN

Requirements:

- As a part of your design, you should include a main class and two interfaces. You should implement the interfaces by two separated classes. Feel free to add any other extra classes in your design. The interfaces are:
    - CalculatorBasicOperationsInterface, where you have the four fundamental operations of mathematics. i.e. addition, subtraction, multiplication and division.
    - CalculatorAdvanced OperationsInterface, where you have at least six extra advanced operations in maths. E.g. random, sqrt, x^2, x^3, absolute value, etc.
- You should design the project and upload the class diagram of your project to your remote repository.
- The project should be a maven project.
- You should add a GUI to your program.
- You should re-write the code generated by WindowBuilder and separate the methods in smaller methods. (This is not a mandatory requirement! But consider doing it!)*
- You should add tests to your project with logging, 100% coverage at unit level and unit tests that uses random input values in a for loop. Set the for loop to 50 iterations to each test method.
- Your tests should test, at least, zero, positive and negative values in a range that you choose.
- You should generate a Javadoc for your project.
- You should generate the test reports by editing the pom.xml file of your project.
- All reports and documentation should be uploaded to GitHub together with all source code and resources of the project.

OBS:

- You can decide how you will take inputs, e.g. two text fields, one text field, etc.
- There is no requirement on how “beautiful” the GUI of your solution is. The only requirement is that your solution follow the requirements and is fully functional.
- I will upload a simple calculator project with a GUI, together with this document, in case you need inspiration to get started.

*OPTIONAL: if you want to design and implement an own project and not a calculator, I am open for that. But you need to show me your idea before you implement it so that I can judge if your project is enough to replace this assignment.

Examples of alternative projects:

- A currency converter, given a value in SEK, it converts the amount to a couple of other currencies of your choice with static conversion rates.
- A converter of Units, e.g. Given a value in meter you can convert it to the following units: millimeter, centimeter, decimeter, decameter, hectometer, kilometer.
- Etc.

*DO NOT PANIC: I will be in the class, during lecture time, and I will try to help as much as I can with hints on how you can succeed with your project. Thus, if you feel that you need help, be present during the lecture time and use my help wisely!