Newton-Raphson Inverse

March 3, 2020

1 Algorithm

This code defines a complex data type and prints it.

2 What to focus on

1. how COBOL implements complex data
2. difference between COBOL implementation and object oriented programming
3. notice the companion object used in kotlin

3 Notes

The code defines a complex data type starting from basic data types. As usual, pay attention to the PICs used in cobol and the data types of the other languages. We define a data type called Employee, which in turns include another complex data type called SSN. The fact that one element is ”included” in another one is given by the 2 digits 01,02,03. Data starting with 02 are part of 01,while 03 is part of 02.
In cobol you can access *directly* all the data, no matter their level 01,02 or 03. Data can be read and modified. In such respect, they may appear similar to a structure in other languages. However, as the ”vector” example shows, it is a bit different. In fact, cobol uses a contiguous block of memory to store the data. In this way, it is more similar to an array of char.

4 Translation

As discussed above, there is not an equivalent data type in java. Thus, you need to define your own. I decided to use a string with a fixed length. Of course, once you define an object, you *must* define the setters and getters to work with that object. Thus, you cannot directly modify data as in cobol. In principle, this should add a level of security.
One difference is the use of a string for SSN, even though cobol uses numbers. As SSN is not used for doing math this works fine for this example. Of Course, such choice must be evaluate case by case.
Be aware of the way vectors are accessed by cobol. Cobol uses the starting point followed by how many elements should be selected. Java uses starting and ending points. Also, notice that java has 0 based numbering, while cobol is 1 based.