# 2. Static Code Analysis of Triangle program

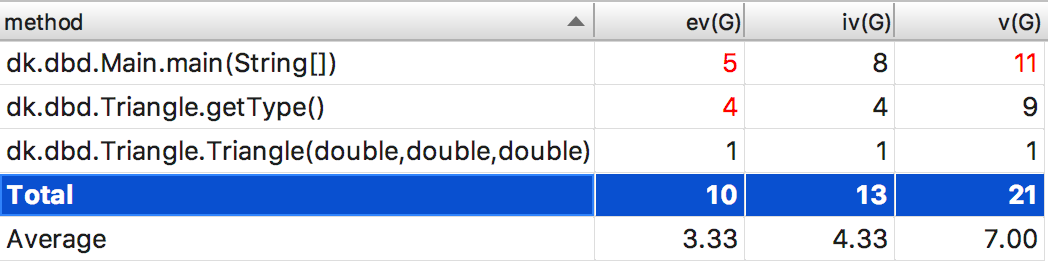
## Calculate central metrics in Triangle program

Depth of nesting

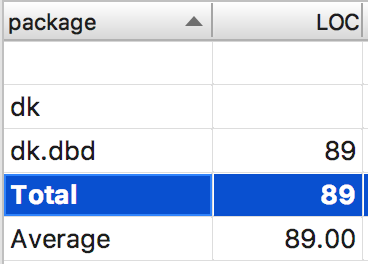
I wanted to include this, but the plugin I was using crashed every time I tried, so there may be a bug with the plugin.

Manually looking over it, I do not think that my code is overly nested though. There are no loops inside of loops, or any complex-to-read code, but that could just be my opinion. It might be because the program, as is, is not that large.

Cyclomatic Complexity



Lines of Code

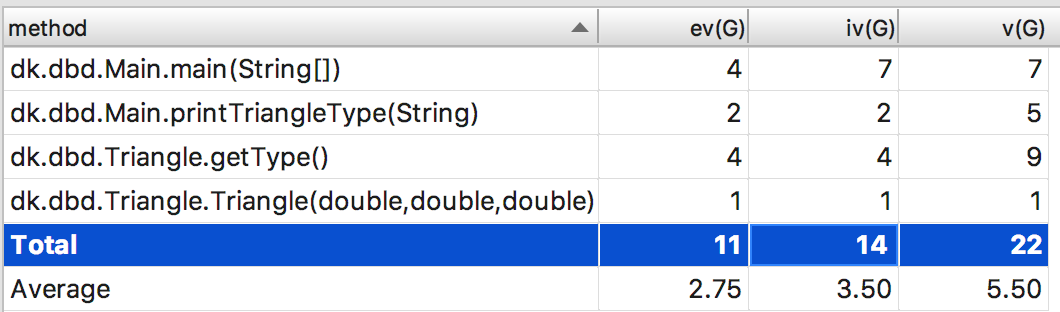


## Calculate central metrics in Triangle program

As seen in the CC screenshot in the section above, the metrics plugin that I’m using actually shows all three CC variants. I figure that the first one is the CC3, the second is the CC and the third is the CC2.

## Refactoring the code

Looking at the CC metrics from the previous section, it seemed to be the *main()* method that had too much cyclomatic complexity. Therefore I chose to split the last switch-case handling the prints into its own method. Afterwards the CC metrics looked a lot better (no red numbers and more diversity):



## xUnit test

With the xUnit tool, I managed to write a few tests on each type of triangle and get them all to pass. The main code is written in Java, so duplicate code couldn’t be avoided when writing tests in the .NET framework.

