***1. Static Code Analysis of Triangle program***

* 1. Install Metrics software in your IDE (see tool examples in slides)

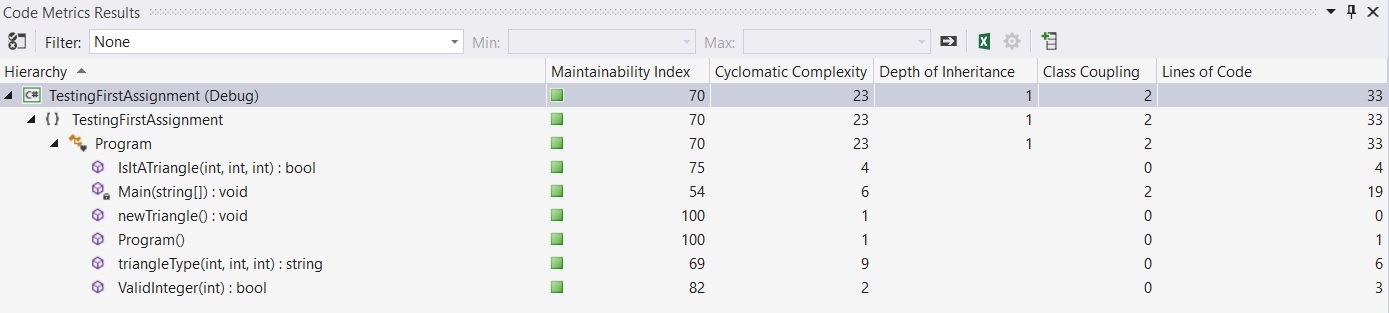
***Done***

* 1. Check coding standards in your Triangle program

***The coding standards is fine in program, every test is isolated in a method with a name relevant to the function. There is also used a try catch statement to catch a input error, if the used doesn’t get the input have to be a integer.***

* 1. Calculate central metrics in your Triangle program

***You see the metrics score in the figure.***



* 1. Find out what CC variation that your metrics tool uses

***Think VS2015 uses a CC2 as variation.***

* 1. Possibly refactor your code based on static testing results

***Don’t need to refactor general good CC score cc lower than 10, and MI higher than 20%. LoC is also low.***

1. ***Peer Review of your Triangle program***
   1. Exchange your Triangle solution with another student

***Exchanged code with Yoana Dandarova.***

* 1. Inspect the other student’s implementation and write down your comments

***My comments on Yoana’s code:***

***It’s a Java application, it contains 3 methods:***

***main(String[] args),***

***void printTriagleType(int x, int y, int z),***

***int enterNumber(Scanner in)***

***The printTriagleType method is a little messy with many lines, else fine working code.***

* 1. Hand over feedback result to each other

***Send Yoana a mail with review, and got:***

*Rune's Triangel Programme Review*

*The programme does work and covers the cases where:*

*-triangle side is negative*

*-string object*

*It is really good the programme checks if the triangle sides actually can*

*be triangle sides.*

*It could be:*

*- used instead of type int for the triangle sides type float*

* 1. Possibly refactor your code based on review results

***I haven’t changed the because the guide say we have to use integers and not float.***

***Yoana’s review of my code:***

1. ***Coding Standard Document***
   1. Create a coding standard document that describes the best practices and code conventions that you find most important for a team to follow.

***Have used the following coding standards (taken from Microsoft on c#, https://msdn.microsoft.com/en-us/library/ff926074.aspx):***

## Naming Conventions

* In short examples that do not include [using directives](https://msdn.microsoft.com/en-us/library/sf0df423.aspx), use namespace qualifications. If you know that a namespace is imported by default in a project, you do not have to fully qualify the names from that namespace. Qualified names can be broken after a dot (.) if they are too long for a single line, as shown in the following example.

## Layout Conventions

Good layout uses formatting to emphasize the structure of your code and to make the code easier to read. Microsoft examples and samples conform to the following conventions:

* Use the default Code Editor settings (smart indenting, four-character indents, tabs saved as spaces). For more information, see [Options, Text Editor, C#, Formatting](https://msdn.microsoft.com/en-us/library/03864tbz.aspx).
* Write only one statement per line.
* Write only one declaration per line.
* If continuation lines are not indented automatically, indent them one tab stop (four spaces).
* Add at least one blank line between method definitions and property definitions.

**Commenting Conventions**

* Place the comment on a separate line, not at the end of a line of code.
* Begin comment text with an uppercase letter.
* End comment text with a period.