

Solutions to Exercise Sheet 1

Exercise 1 - Metrics

1.1 Lines of Code Metrics

$$\begin{aligned} \text{i } LOC_{tot} &= 74 \\ LOC_{ne} &= 74 - 10 = 64 \\ LOC_{pars} &= 64 - 15 = 49 \end{aligned}$$

ii Example Haskell-Code for contrasting the given MyQuickSort.java:

```

1  module MyQuickSort where
2
3  -- This Code is self-Documenting
4  quicksort :: Ord a => [a] -> [a]
5  quicksort []      = []
6  quicksort (x:xs) = smaller ++ [x] ++ bigger
7      where
8      smaller = quicksort [y | y <- xs, y <= x]
9      bigger  = quicksort [y | y <- xs, y >  x]

```

$$LOC_{parsH} = 9 - 2 = 7$$

So there is LOC_{pars} with 49 as well as LOC_{parsH} with 7 (Order of magnitude: n^2 vs $n!$). These are obviously two entirely different Programs, yet they are semantically equivalent in that they offer an interface to a function capable of sorting a List of items with a Quicksort-Algorithm.

In this case the recognized Pattern is to use a library or tool requiring a lot of (hardcoded) configuration, where a simpler one would clearly suffice. That way you would have a lot of managing / organizing / configuring overhead, which can be overblown to the fullest if wanted (resulting in an LOC of literally any number you wish).

iii Metrics:

$$\begin{aligned} LOC_{tot} &= 178 \\ LOC_{ne} &= 136 \\ LOC_{pars} &= 120 \end{aligned}$$

It basically configures all the GUI-Elements (like buttons etc.), their Positions, Sizes, how they should behave when resizing the window or when being clicked, and more. The Program itself is a simulator for AI-Ants to find paths in a generated Maze, the whole Project written in C++.

1.2 Cyclomatic Complexity

1. $p = 3$
 $n = 13$
 $e = 17$
 $v(G) = 17 - 13 + 3 = 7$
The CFG will be on the last page.
2. Junction points in a CFG do not alter the cyclomatic complexity as it adds an edge for each node.