Principles of Clean Code

Keep it simple, stupid (KISS), *you aren't gonna need it* (YAGNI), and *don't repeat yourself* (DRY) are some of the most powerful digital product design principles. They lay the foundations for best practices that developers use to build better products every day.

Don't Repeat Yourself (DRY)

This principle suggests that code should not have unnecessary duplication. Instead, it should be organized in a way that avoids redundancy and makes it easy to maintain.

For example, instead of writing the same calculation in multiple places in the code, create a function that performs the calculation and call that function from the different places where the calculation is needed.

An example is given below. Beginners in programming will write the following program if they wish to display a multiplication table of 7 in different lines:

```
println(7 * 1);
println(7 * 2);
println(7 * 3);
println(7 * 4);
println(7 * 5);
println(7 * 6);
println(7 * 7);
println(7 * 8);
```

Supposing, someone wishes to display multiplication table till seven times a hundred, then this would add another 90 lines to the program!

This is where the DRY principle comes into the picture. Why repeat line items in a program when you can write the same program in three lines using the *Loop* command for n number of repetitions. Hence, the above example may be written as below:

```
for (var i = 1; i < 11; i++)  {
println(7 * i);
}
```

KISS:

Programmers are always asked to avoid complex programming or create difficulties in understanding the code. The Key is to program only as much as is required, neither less nor more.

Few ways by which a programmer can reduce complexity:

- The variable names in a program should serve their purpose well. They should be able to define the variable properly
- Method name should also be in line with the purpose for which the method is employed
- As mentioned in an earlier example in the DRY principle, comments in the method should be used only when your program is not able to define the method completely
- Classes should be designed in a way to own a single responsibility
- Delete redundant processes and algorithms as explained during the DRY process

Following are the advantages of applying the KISS principle in coding:

- It is vital to apply the KISS principle when you are working on the modification of an existing codebase. It helps clean the code and make it more readable and editable
- Application of this principle helps in maintaining continuity in the development of a code when the programmer changes.
- KISS principle enhances the efficiency during automated testing of the code.

- Fewer chances of errors during coding.

YAGNI:

YAGNI as a principle means the removal of unnecessary functionality or logic.

Often programmers have a tendency of over-seeing the future and overloading the program with unnecessary logic, algorithms, methods, and codes which will never be used. Some do it as a business requirement and some do it in fear of not being able to incorporate it later when a certain situation arises. A few situations where a programmer should consider following YAGNI are:

- If you are asked to do validation of email and password fields through a program, there is no need to add coding to validate the name field also. You may never need it.
- If a programmer is asked to link different databases of cancer patients of a hospital with a health application, the developer should not consider the hospitals which are already closed. They may never open, and the patient database must have already moved to active hospitals.
- Some programmers may create abstractions even when they are having just one well-defined case in anticipation of the addition of cases. So, they write unnecessary codes and make assumptions about further cases. This should be avoided.
- Use of If-else logic even if the “else” part is always going to be negative in all the test scenarios encountered.