Principles of Clean Code

1. After making code work, spend time to make it clean.
2. Code must be intuitive for readers.

Functions

A function should do one thing

- To verify that your function does one thing, try to extract lines and create another function out of it.
- Too many well named functions are good for readability.
- Don't pass more than 3 arguments to a function.
- Don't pass true or false booleans in function.
- A well extracted function won't be more than 6 lines of code.
- Don't pass variable that stores output.

Don't use switch statements

- Modifying a small thing needs modifications in switch statements each time.
- Better to use classes and define functions in it.

Create functions to avoid side-effects

- Create a function that executes another function to avoid side-effects.
- Eg of side-effect is opening a file and forgetting to close it.
- To avoid forgetting to close a program, create function:

```
void process_file(char *filename,     function processing_function){
    FILE *f = fopen(filename, "r");
    processing_function(f);
    free(filename);
}
```

- Differentiate between commands and queries.

Have just try-catch block in function

- Try catch block should have just on function call.
- Never have nested try-catch statements.

Use functions to parse through and modify complex data structures

- Use function to walk through the complex data structure and pass function to modify it as a parameter.

Source Code

Comments

- Use very little comments.
- Instead try to make code more readable.
- Talk only about the code in the function in the comments.

No. of lines

- Source code may have less than 100 lines of code.
- Though this restriction is not that mandatory.

Line length

- Mostly line length is about 40 characters.

Naming

Length of variable names

- Length of variable name must be proportional to the scope of variable.
- If a variable is used just in one line or in small block, it can be of one letter.
- If a variable is used inside a class it should be bit long, maybe about two words.
- If the variable is used globally, it's name should be descriptive.

Length of function or classes names

- Length of variable name must be inversely proportional to the scope of its usage.
- If function is used globally, it will be called more often and must be abstract.
- As scope of function decreases, it should be more descriptive.

Have Tests

- Have tests to make modifying/cleaning codes easier.
- Each time one modifies code, he/she can test it and see if it didn't break.
- Calculate code coverage.

Three laws of test driven development

1. You cannot right a code until you have created a test code that fails if the code isn't there.
2. You cannot write more test than it is sufficient to fail.
3. You cannot wite more code than is sufficient to pass currently failing test.

How to do test driven development

- First write test case to fail code, then write code to pass test case.
- Each time test cases constraint the code and code becomes more general to pass test cases.
- Don't rush for gold (central behavior), first write test cases for all other small things surrounding the central behavior.

Timeline

- Have three dates of completing tasks instead of one: best-case, average-case and worst case.

Coding Style

Design Patterns

- Know about design patterns.
- Design patterns provide common language to tell about coding patterns.

sequence , selection and iteration

- All program is a combination of sequence , selection and iteration.