Department of Computer Engineering
University of Peradeniya
Lab 05

Programming Methodology

April 25, 2018

1 Introduction

Clean code is a key practice in professional software development. Most of the times, the graduates passing out from the universities lacks the practice of clean code. Being able to write a better code would not only make you a great programmer, but also an excellent candidate for any job interview you can stand out from any competitor. In software engineering, all the qualifications you have, all the skills you have gained, all the technological knowledge you have learned will finally expressed as a piece of code. This is where the idea of clean code comes into the picture. You do not write a piece of code to tell a computer how to do a task. As Martin Fowler said "**Any fool can write code that a computer can understand. Good programmers write code that humans can understand**"

There is a difference between being a smart programmer and professional programmer. "In general, programmers are pretty smart people. Smart people sometimes like to show off their smartness by demonstrating their mental juggling abilities." (Which would create unreadable, complicated programs.) "One difference between a smart programmer and a professional programmer is that the professional understands that clarity is the king. Professionals use their powers for good and write code that others can understand."

These words are quoted from "A Handbook of Agile Software Craftsmanship" by Robert C. Martin (colloquially known as Uncle Bob). The book shows an excellent guide how to write a good piece of code. This exercise, lab 05 will heavily rely on the content provided by the mentioned book. Though you do not have learned enough to understand the full context of the book, the knowledge upto the functions is enough to understand a fair amount of the details (Chapter 1 - Chapter 5).

2 Few key points as directly quoted from the book

- Indeed, the ratio of time spent reading vs. writing is well over 10:1. We are constantly reading old code as part of the effort to write new code.
- The Boy Scouts of America have a simple rule that we can apply to our profession. Leave the campground cleaner than you found it.
- Use Intention-Revealing Names If a name requires a comment, then the name does not reveal its intent.
- Use Pronounceable Names.

- Methods (or functions) should have verb or verb phrase names like postPayment, deletePage, or save.
- Shorter names are generally better than longer ones, so long as they are clear. Add no more context to a name than is necessary.
- The first rule of functions is that they should be small. The second rule of functions is that they should be smaller than that. Functions should hardly ever be 20 lines long.
- FUNCTIONS SHOULD DO ONE THING. THEY SHOULD DO IT WELL. THEY SHOULD DO IT ONLY.
- One Level of Abstraction per Function.
- Dont be afraid to make a function name long. A long descriptive name is better than a short enigmatic name. A long descriptive name is better than a long descriptive comment. Use a naming convention that allows multiple words to be easily read in the function names, and then make use of those multiple words to give the function a name that says what it does.
- The ideal number of arguments for a function is zero (niladic). Next comes one (monadic), followed closely by two (dyadic). Three arguments (triadic) should be avoided where possible. More than three (polyadic) requires very special justification and then shouldnt be used anyway.
- Using an output argument instead of a return value for a transformation is confusing. If a function is going to transform its input argument, the transformation should appear as the return value.
- Functions should either do something or answer something, but not both. Either your function should change the state of an object, or it should return some information about that object.
- Indeed, comments are, at best, a necessary evil.
- The proper use of comments is to compensate for our failure to express ourself in code.
- First of all, lets be clear. Code formatting is important. It is too important to ignore and it is too important to treat religiously.
- Each line represents an expression or a clause, and each group of lines represents a complete thought.
- If one function calls another, they should be vertically close, and the caller should be above the callee, if at all possible. This gives the program a natural flow

3 Lab task

Your task is to re-implement the Lab04 source code using functions (and best practices).

4 Submission

Submit a single .c file rename it as the following pattern where xxx is your registration number.
15xxxlab05.c

5 Important

During the exercise, you are expected to spend a considerable amount of effort and time to learn and apply the proper usages of functions and best practices. Since the lab carries equal mark as any other lab, the work you have done should appear from the source code. During marking the lab, the source code will be given 100 marks and for each flaw noticed would result reducing some marks.

Under no circumstance, you should copy somebody else's code. Copying someone else's code (including your group mate's) or showing your source code to anyone else will earn you zero mark for the whole lab exercise.

6 Deadline

The deadline for the submission is Saturday (28th May 2018) 23:55h.

7 Be ready for next lab session

1. Create a hackerrank account and try to solve some problems there
2. We will use hackerrank platform for the lab 06 and lab 07 exercises

8 References

- http://ricardogeek.com/docs/clean_code.pdf A Handbook of Agile Software Craftsmanship" by Robert C. Martin.