Some Ideas on Implementation

Hui Chen [a]

[a]CUNY Brooklyn College, Brooklyn, NY, USA

May 4, 2023

Outline

Outline

Working Code vs. Beautiful Code

Working code doesn't necessarily mean good code.

Working Code vs. Good Code

What's wrong this the following code? (Source: Steve McConnell[1])

```
void HandleStuff( CORP_DATA inputRec, int crntQtr, EMP_DATA empRec,
  double estimRevenue, double ytdRevenue, int screenX, int screenY,
  COLOR_TYPE newColor, COLOR_TYPE prevColor, StatusType status,
  int expenseType ) {
  int i;
  for ( i = 0; i < 100; i++ ) {
    inputRec.revenue[i] = 0;
    inputRec.expense[i] = corpExpense[ crntQtr ][ i ];
    }
  UpdateCorpDatabase( empRec );
  estimRevenue = ytdRevenue * 4.0 / (double) crntQtr;
  newColor = prevColor;
  status = SUCCESS;
  if ( expenseType == 1 ) {
    for ( i = 0; i < 12; i++ )
      profit[i] = revenue[i] - expense.type1[i];
    }
  else if ( expenseType == 2 ) {
    profit[i] = revenue[i] - expense.type2[i];
  }
  else if ( expenseType == 3 )
    profit[i] = revenue[i] - expense.type3[i];
  }
```

[1] Steve McConnell. *Code complete*. Pearson Education, 2004.

What's Wrong with `HandleStuff`?

- The routine has a bad name; handleStuff tells you nothing about what it does
- The input variable inputRec is changed. If it's an input variable it should not be modified (and declared const).
- It uses global variables (e.g. profit)
- It doesn't have a single purpose. It does too many things: reading from DB, some calculation which don't have a common goal
- The routine does not defend itself against bad data
- Uses "magic" numbers (e.g. 100, 1, 2)
- Some parameters are unused
- Too many parameters
- Parameter names do not make their meaning obvious
- Others?

Beautiful Code

Long-lasting code that is easy to evolve.[2]

How do we create beautiful code? We discuss several ideas,

- Don't Repeat Yourself (DRY)
- Clarity via Conciseness
- Idiomatic Code

[2] Armando Fox, David A Patterson, and Samuel Joseph. *Engineering software as a service: an agile approach using cloud computing*. Strawberry Canyon LLC, 2013.

Writing DRY Code

Your code should be DRY!

DRY = Don't Repeat Yourself

- If you are about to write repetitive code, stop!
- Refactor into a function; or
- Refactor into a class / abstract class; or
- Create a library or a module

Modern IDEs can detect repetitive code – a code smell

Why DRY Code

- Modularity – Some parts of the code are easier to replace if they are in a routine, without disturbing the rest of the implementation
- Testing – Easier to perform tests
- Single Optimization Point – You can optimize the code in one place instead of several
- Reduce Complexity – Place code in the routine so that you don't have to think about it after it is written
- Introduce Abstraction – Putting a section of code into a well named routine/class/module/package is one of the best ways to document its purpose

Clarity via Conciseness

- Syntax: shorter and easier to read, e.g., (via Google Guava)
  `theDigits = CharMatcher.DIGIT.retainFrom(string);`
- Avoid long statements (if you can)
- Avoid redundant words (if you can)
- But YOU CAN and SHOULD give comprehensible variable, method and class names!

Self Documenting

Code should be self-documenting

- Comments should be useful high-level descriptions of what the program is doing. They should not restate something that is “obvious”.
- Self documenting code uses well-chosen variable names (and function names) to make the code read as close to English as possible
    - For example, naming a variable g has little meaning, but naming a variable gravity gives a much better description of what the variable should contain.
    - By using proper variable and function names, you should minimize the amount of "external" documentation that is necessary.

Self Documenting Code: Example

Compare the following two, which one is more self-documenting?

```
List<String> l = new ArrayList<String>();
l.add("hello");
l.add("world");
l.add("one");
```

versus

```
List<String> listOfWords = ImmutableList.of("hello", "world"
  , "!")
```

Writing Idiomatic Code

- Coding conventions detail:
  - how the code should look like (i.e., aesthetics)
  - how some specific aspects of the code should be handled (e.g. exceptions), e.g.,
    - Where and when should I place curly brackets?
    - How should I name variables?
    - How much space should give between lines, between parameters, etc?
- Various projects and domains have their own conventions
  - Check if you want to contribute code

Coding Convention: Example

Coding conventions for Android contributors

https://source.android.com/setup/contribute/code-style

Coding conventions for Android contributors: Examples

- Fully qualify imports (import A.B.c vs. import A.B.*)
- Order import statements (Android imports → third party imports → java and javax)
- Write short methods (must be less than 40 LOC)
- Define Fields in Standard Places (fields should be defined either at the top of the file, or immediately before the methods that use them)

Checkstyle Tool

Installed as a plugin in AndroidStudio

File -> Settings -> Plugins

Checks whether your program adheres to a set of style rules (e.g. Sun rules, Google rules)

Lint Tool

Analyzes source code structure for known errors, bugs, and stylistic problems.

Originates in Bell Labs in 1978, but many modern versions exist

For Android

https://developer.android.com/studio/write/lint

Outline

Summary

- Don't Repeat Yourself (DRY)
- Clarity via Conciseness
- Idiomatic Code
  - Tools: Checkstyle, Lint

What makes a great software engineer?

- Opinion. Practice – the 10,000 hours of deliberate practice
- Research papers, e.g.,
  - Li PL, Ko AJ, Zhu J. What makes a great software engineer?. In2015 IEEE/ACM 37th IEEE International Conference on Software Engineering 2015 May 16 (Vol. 1, pp. 700-710). IEEE.
- ...

Outline

"Engineering Software as a Service" by Armando Fox and David Patterson (2nd Edition)
"Introduction to Software Design with Java" by Martin P. Robillard
"Essentials of Software Engineering" by Frank Tsui, Orlando Karam, and Barbara Bernal(4th Edition)