Learning from Writing Well

Code Quality

- Can come from different angles.
  - Is the code understandable?
  - Are the methods of a reasonable size?
  - Are the classes named well?

Cost and Consequences

- Poor quality code is more expensive to maintain.
- Programs aren't just written — they are rewritten. So, focus on writing good quality code.

Learning from "On Writing Well" book

> "Hard writing makes easy reading. Easy writing makes hard reading."
> -William Zinsser

Applying the principles to software development

There are four principles mentioned in the book for writing English. These apply to writing code as well.

1. Simplicity
   a. There are two kinds of complexity when writing code — inherent complexity and accidental complexity.

        i. Inherent complexity often comes from the problem domain. It doesn't go away too easily.
        ii. Accidental complexity is something that programmers tend to introduce in the solution space. We must strive very hard to eliminate or minimise accidental complexity as much as possible and make it as manageable as we can.
    b. Often we tend to write poor quality code in the name of good performance.
    c. A simple code is something that is going to fail a lot less.
    d. A simple code involves fewer mutability, fewer state changes.
2. Clarity
    a. Cohesive code is clearer because it is narrow and focused. It does one thing and it does it well.
        i. A code that is not cohesion does several things in one place. That violates the Single Responsibility Principle (SRP).
    b. A code that has less coupling is clearer to understand as well.
    c. Code can be made clearer by making it smaller — with smaller methods, which are easier to understand than larger methods.
    d. Code is more often read than written, so clarity is paramount.

> Programs must be written for people to read, and only incidentally for machines to execute. — Abelson & Sussman

3. Brevity
    a. Give variable short descriptive names. Eg. `distanceInMiles` instead of `d`.
    b. Shorter methods and smaller classes are better than long methods and large classes.
    c. A system with fewer number of classes is better than one with a lot of classes.

    d. A shorter hierarchy of inheritance is better than a deep hierarchy.

4. Humanity