Summary Report

Art of Readable Code

Chapter 02

KEY IDEA

Pack information into your names.

- **Choosing specific words**
  - Do not use generic words or names like, “size( ), print( )”. Make it more meaningful, and specific.

Word	Alternatives
send	deliver, dispatch, announce, distribute, route
find	search, extract, locate, recover
start	launch, create, begin, open
make	create, set up, build, generate, compose, add, new

  - ***It’s better to be clear and precise than to be cute.***
- **Avoid Generic Names Like tmp and retval**
  - Pick a name that describes the entity’s value or purpose.
  - A better name would describe the purpose of the variable or the value it contains.
    - The name **retval** doesn’t pack much information. Instead, use a name that describes the variable’s value.
    - The name **tmp** should be used only in cases when being short-lived and temporary is the most important fact about that variable.
    - Loop Iterators such as i, j, iter.
  - If you’re going to use a generic name like tmp, it, or retval, have a good reason for doing so.
- **Prefer Concrete Names over Abstract Names**
  - Assign names to a variable, class, function, etc. that best describes the its functionality.
- **Attaching Extra Information to a Name**
  - Value with units

Function Parameter	Renaming parameter to encode units

Start(int delay)	delay -> delay_secs
CreateCache(int size)	size -> size_mb

- Encoding other important Attributes

Situation	Variable name	Better name
A password is in “plaintext” and should be encrypted before further processing	password	plaintext_password

- Hungarian notation is a system of naming used widely inside Microsoft. It encodes the “type” of every variable into the name’s prefix.

- **How Long Should a Name Be?**
    - The longer a name is, the harder it is to remember, and the more space it consumes on the screen, possibly causing extra lines to wrap.
    - **Shorter Names Are Okay for Shorter Scope**
        - So if an identifier has a large scope, the name needs to carry enough information to make it clear.
    - **Typing Long Names—Not a Problem Anymore**
        - Every programming text editor we’ve seen has “word completion” built in.
    - **Acronyms and Abbreviations**
        - Project-specific abbreviations are usually a bad idea.
        - They appear cryptic and intimidating to those new to the project.
        - Given enough time, they even start to appear cryptic and intimidating to the authors.
    - **Throwing Out Unneeded Words**
        - Sometimes words inside a name can be removed without losing any information at all.
- **Use Name Formatting to Convey Meaning**
    - Depending on the context of your project or language, there may be other formatting conventions you can use to make names contain more information.
    - Whether you decide to use conventions like these is up to you and your team. But whichever system you use, be consistent across your project.

SUMMARY

The single theme for this chapter is: pack information into your names. By this, we mean that the reader can extract a lot of information just from reading the name.

Here are some specific tips we covered:

• Use specific words—for example, instead of Get, words like Fetch or Download might be better, depending on the context.
• Avoid generic names like tmp and retval, unless there’s a specific reason to use them.
• Use concrete names that describe things in more detail—the name ServerCanStart() is vague compared to CanListenOnPort().
• Attach important details to variable names—for example, append _ms to a variable whose value is in milliseconds or prepend raw_ to an unprocessed variable that needs escaping.
• Use longer names for larger scopes—don’t use cryptic one- or two-letter names for variables that span multiple screens; shorter names are better for variables that span only a few lines.
• Use capitalization, underscores, and so on in a meaningful way—for example, you can append “_” to class members to distinguish them from local variables.