Chapter2: Meaningful names

Use intention-revealing names

Choosing good and revealing names for the variables and functions

Further, we can define simple (unnecessary) classes so that the code can be more readable

Avoid disinformation

Do not use standard abbreviations for other meanings

It is helpful if the names for similar things sort together alphabetically + obvious differences

Make meaningful distinctions

Naming a1,a2 … is a bad practice

Noise words redundancy is bad: never add “string” in the name of a string variable, same for tables or just, variables. Never put moneyAmount and money, or account and accountData so that the user does not get confused

Use pronounceable names

Use searchable names

Single-letter names are hard to locate via ctrl+F .They should only be used as local vars (for ex. Inside a smallfor loop)

Avoid encodings

Avoid mental mapping

Don’t be smart, be professional. Clarity is king

Class names

Choose Customer not customerData, AddressParser not AddressParserInfo (avoid info manager processor Data etc… in naming the class)

Method names

Methods should have verbs.. Get, set, is …etc : javabean standards

When constructors are overloaded use static factory methods

Don’t be cute

Don’t use slang or inside jokes to code

Pick one word per concept

Ex: choose either fetch, get or retrieve in all the code base for easy access/remembering

Don’t pun

Avoid using same word for different purposes. (should work equivalently)

Use solution domain names

Stick to conventional namings so that the customer can understand the code

Add meaningful context

For address, instead of using state use addrState

Don’t add gratuitous Context

Shorter names are better than longer ones as long as they are clear. Add no more context to name than is necessary

Final words

Good names require good descriptive skills and a shared cultural background

Renaming things to the better is a good thing, don’t be afraid of it

Chapter3: Functions

Small!

Functions should be always the smallest possible : 100 lines is too much

Blocks and indenting: blocks inside if statements should be one or two lines long not more

Do one thing

Functions should do one thing. They should do it well. They should do it only (one level of abstraction)

One level of abstraction per function

If you need another level, call a function

Switch statements

Sometimes we cannot avoid them, but make each switch statement buried in a low-level class and never repeated (polymorphism)

Use descriptive names

Don’t be afraid to make a name long, better than short enigmatic name or long descriptive comment

Spend time on changing names (sometimes helps restructuring the code)

Be consistent in the names, similar phraseology allows the sequence to tell a story

Function Arguments