# Naming as a Process

## The Path to Good Names?

Jörg Decker
25.03.2022

# Why is it so complicate?

- Name should describe what the code should do
- Describe side effects
- Should be different from others
- Should be homogeneous
- Should be understandable
- Not too long
- ...

# Why is it so important?

„Programmers spend around 60-70% of their entire programming time reading code.“

According to an analysis from Eclipse data

# Bugs come from incomplete understanding

Mistakes happen when our mental model doesn’t match the reality of the code.

# Naming as a process

The solution ... annoyingly simplified

At each step, I look at one part of the code, understand one kind of thing that is happening, have an insight, and write it down. I repeat this until I have moved one step down this list. I keep going until I have a good enough name for my purpose.

Arlo Belshee

# Get to Obvious Nonsense

The most terrible kinds of names are:

- Misleading names
- Missing (code is part of something else)

# Fixing Missing Names

Step 1: Look at long things (classes, methods, files, ...)

Step 2: Look for chunks that hangs together

Step 3: Extract it as Applesauce

# Fixing Missing Names

- The body of a control structure is often a good target
- If there are multiple control structures extract them into seperate methods

# Fixing Misleading Names

- Step 1: Look for under- or misinformation
  - Methods named by lifecycle
  - Variable named the same as its type
  - Method names that leaves out critical information
  - Any name that ends in -er or -utils
- Step 2: Rename it to Applesauce

# Get to Honest

- Look for one thing the code does
- Rename it to something better which includes
  - One thing it does
  - Clarity what we don‘t know yet

  *e.g. probably_doSomethingEvilToTheDatabase_AndStuff*
- Be specific
- it‘s honest but not complete

# Get to Completely Honest

This level makes it unnecessary to read the code. We want them to be able to trust that the name includes absolutely everything the method does so they can read and understand calling methods without having to read the method we’re fixing.

# Get to Completely Honest

- Expand the Known
- Narrow the Unknown

  e.g:
  - _AndDoSomethingToDatabaseAndStuff
  - Insight: the code doesn‘t write
  - New name: _AndWriteSomethingToDatabaseAndStuff

    Repeat these steps
- Be precise
- e.g:
  *parseXMLAndAddFlightToDBAndLocalCacheAndAddToScreenIfVisible*

# Get to Does the Right Thing

- Part 1: Look only at the name
- Part 2: Look for one responsibility to separate
- Part 3: Structural refactoring
  - Moving responsibility
  - Encapsulation

e.g.

- ParseXML
- SaveFlightToDB
- ShowFlightOnScreenIfVisible

# Get to Intent Revealing

- Part 1: Look at the calling methods
- Part 2: Look for the code‘s purpose
- Part 3: Write down the purpose by renaming the target

  e.g.
- StoreFlightToDatabaseAndShowOnScreenIfVisible → beginTrackingFlight()

# Get to Domain Abstraction

- Shared Context for some set of code
- Looking for Primitive Obesession
- Write down by adding new ValueObjects

# Naming as a process

- Get to Obvious Nonsense
- Get to Honest
- Get to Completely Honest
- Get to Does the Right Thing
- Get to Intent Revealing
- Get to Domain Abstraction

# Naming as a process

- Does it work?
- We scratched the surface
- Highly interwoven with several refactoring steps towards DDD
- Good naming is a process, not a single step

# Naming as a process

- Check it out at https://www.digdeeproots.com/articles/on/naming-process/ by Arlo Belshee

# Thanks for your audience

- Any questions?

Jörg Decker

joerg.decker@css.ch