Naming as a Process

Get Started Today!

Contents

Technique #1 | Make an Honest Name 2
Template 2
Your Practice 2
Your Challenge 2
Your Reflection 2
The New Problem 2
Technique #2 | Use Obvious Nonsense 3
Template 3
Your Practice 3
Your Challenge 3
Your Reflection 3
The New Problem 3
Technique #3 | Expand the Known 4
Template 4
Your Practice 4
Your Challenge 4
Your Reflection 4
The New Problem 4
Technique #4 | Narrow the Unknown 5
Template 5
Your Practice 5
Your Challenge 5
Your Reflection 5
The New Problem 5

Technique #1 | Make an Honest Name

State exactly what you know, and not what you interpret or assume.

Template

probably_[TheOneThing]_AndStuff

Your Practice

When you find code that takes you longer than 3 seconds to understand, do the following:

1. Stop! Don't try to figure it out (even though you could).
2. Find one thing you are reasonably confident that it does.
3. Rename it according to the template, using a rename refactoring.
4. Commit just the rename and add the tag [honest] at the end of your commit message.

Your Challenge

- Become Familiar | Write 1 honest name a day
- Become Proficient | Write 8 honest names a day
- Become a Master | Write 24 honest names a day

Your Reflection

- Review your honest tags.
- How many were there?
- Upon reflection, where did you miss opportunities to be honest? How could you see those next time?
- Read some code that calls to one of the methods you renamed. How do you react to the new name? How did you react to the old name? What are the key differences you see?

The New Problem

Extracting a method interrupts our thinking flow. We must extract methods to make honest names, then we freeze for each one as we come up with an honest name. Continue to the next technique to address this!

Technique #2 | Use Obvious Nonsense

Delay naming until we have more information.

Template

Applesauce

Your Practice

When you find yourself having to read a chunk of code to understand it, or are surprised by what a chunk of code does.

1. Identify whether the name is **missing** or **misleading**.
2. If missing: rename it to applesauce (a nonsense name).
3. If misleading: extract the code that is missing a name to a method or variable expression and name the extracted code applesauce (a nonsense name).
4. Commit just the rename and add the tag [applesauce] at the end of your commit message.

Your Challenge

- Become Familiar | Use Applesauce 1 time a day
- Become Proficient | Use Applesauce 1 time a day
- Become a Master | Use Applesauce 10 time a day or with every Extract Method

Your Reflection

- Review the applesauce tags.
- How many times did you practice this technique?
- Did you rename any misleading names? How did your understanding of the code shift just by replacing the old name with applesauce?
- Was it easier to get to an honest name in two steps?
- Did you have any times where the applesauce stayed in the codebase for more commits than you expected? What would have happened if you had not done the applesauce commit?

The New Problem

Now we can quickly get to names that partially honest. They don't lie but they also leave out important information. It would take too long to get all the important information in one step, so we need an iterative process to collect all the information into the name. The next technique will start you on that iterative process!

Technique #3 | Expand the Known

Gather and record the information instead of trying to understand.

Template

Probably_[OneFact]And[OneFact]…_AndStuff

Your Practice

When you need to know more about what your honest method does:

1. Identify the question you are trying to answer about this code for your current story.
2. Quickly scan the method body to find the part that might answer that question.
3. See one fact. Stop with the first thing you notice that is not already in the name.
4. Add it to the name.
5. Look around for more facts until your question is answered.
6. Stop as soon as you have your answer. You don't need to record any more facts right now.
7. Commit just the rename and add the tag [add-fact] at the end of your commit message.

Your Challenge

- Become Familiar | Add a fact 2 times a day
- Become Proficient | Add a fact 8 times a day
- Become a Master | Add a fact 20 times a day

Your Reflection

- Review your add-fact tags.
- How many times did you practice this technique? What is the greatest number you managed for a single name?
- How did this cycle allow you to record the information needed for this story, and then stop?
- How would this naming cycle help your colleagues continue for the next story?

The New Problem

Sometimes the facts aren't obvious though. We can see an open question, but it takes it deep understanding to answer that question. As such, we need an incremental process to gather information and refine our question until it's easy to answer. Onward and forward to the last technique in this module that solves that problem!

Technique #4 | Narrow the Unknown

Eliminate rabbit holes that force you to read and verify.

Template

Probably_[OneFact]And[OneFact]And[OneFact]_[OneQuestion]And[OneQuestion]

Your Practice

When you are trying to figure out whether a method is impacted by your story and the name is not enough:

1. Identify which pieces of data are likely related to your story and which are likely not.
2. Use method signatures to make guesses about what the method could or could not do. What is potentially in its scope or out of its scope, given the data it has access to?
3. Record those guesses in the open questions list.
4. Pick one of the methods most likely to impact your story.
5. Follow the data within the method to make guesses about what this method uses the data for.
6. Update your previous guess in the open questions list for this method with your new, more specific guess.
7. Look back at the set of all names and choose where to look next. Repeat until you find the code most likely to be relevant.
8. Commit just the rename and add the tag [refine-question] at the end of your commit message.

When you realize a partial fact about a method, but it raises other questions, then do the following:

1. Add the question in the open questions list.
2. Record the fact in the facts list unless your question effectively already includes it.

Your Challenge

- Become Familiar | Add a question 2 times a day
- Become Proficient | Add a question 8 times a day
- Become a Master | Add a question 20 times a day

Your Reflection

- Review your refine-question tags.
- How many times did you practice this shift?
- How did this cycle help you discover that you didn’t have to read and understand a chunk of code?
- How will that help future readers?

The New Problem

Your name is completely explicit about what the code does, but that may not be what you actually want the code to do. Now you can fix it!