Use Intention-Revealing Names

The passage talks about how important it is to choose good names when writing code. It says that names should clearly show what they represent, like variables, functions, or classes. It suggests picking names that quickly tell you what something does or is used for. Using clear names helps people understand the code better and makes it easier to change later on. It gives an example of how using unclear names can make code hard to understand. Then, it shows how renaming things and improving names can make the code much clearer. Overall, it says that taking the time to pick good names for things in code makes a big difference, making it easier to read and work with in the long run.

Avoid Disinformation

This paragraph emphasizes how important it is to choose clear and meaningful names when writing code. It warns against using names that could confuse others. For example, using short forms or vague terms can make it hard to understand what the code does. It also advises against using similar-looking names that might be mistaken for each other, like using lowercase "l" and uppercase "O" which look like the numbers one and zero. The text stresses the need for consistent and easy-to-understand naming, especially when using tools that help write code automatically. Good naming helps developers understand and use code without needing lots of comments or explanations. Overall, clear naming is crucial for making code understandable and improving how software is developed.

Make Meaningful Distinctions

This passage talks about how important it is to choose good names when writing code. It warns against using names that don't make sense or aren't helpful, which can make the code harder to understand and work with. Instead, it suggests using names that clearly show what each part of the code does. For example, it's better to use names that describe the purpose of a variable or function rather than just satisfying the computer. It also says that names should be unique and not include unnecessary words that don't add meaning. Confusing or similar-sounding names should be avoided to make it easier for programmers to read and understand the code, especially when multiple people are working on the same project. Overall, the passage encourages developers to be thoughtful when naming things in their code to make it easier to read, maintain, and collaborate on.

Use Pronounceable Names

This passage stresses how important it is to choose good names when writing code. It says that using names that are easy to understand and say helps programmers work together better. If a name is hard to pronounce, it can make discussing the code tricky and slow down teamwork. It gives an example of a company that had trouble with confusing names and how it made communication harder. It compares badly named things in code with well-named ones to show how much of a difference it makes. Overall, it says that using clear and easy-to-say names in code makes it easier for programmers to understand each other and work together well.

Use Searchable Names

This passage emphasizes how important it is to give your variables meaningful names when writing code, and warns against using short or cryptic names like single letters or numbers. It explains that using vague names can make it hard to understand code, especially when searching for specific terms. Instead, it suggests using longer and descriptive names to make code clearer and easier to search through. Single-letter names should only be used for short parts of code, like small functions. By comparing different code examples, it shows that using clear names makes code easier to read and maintain, even if it makes the code a bit longer. Overall, it argues that the benefits of having understandable code outweigh the slight inconvenience of longer names, leading to code that is easier to work with and understand.

Avoid Encodings

The passage talks about the problem of adding extra codes to variable names unnecessarily. It disagrees with practices like Hungarian Notation and adding "m_" before member variables. It says that while these practices were useful in older programming languages with limited features, they're not needed in modern languages like Java. Instead, they just make the code harder to understand and change. The text suggests that programmers should focus on giving clear and meaningful names to variables and use the features of modern coding tools to make their code easier to understand. It also discusses how to name interfaces and implementations clearly without adding unnecessary codes. Overall, it stresses the importance of keeping code naming simple and readable, without adding extra burdensome codes.

Avoid Mental Mapping

The passage emphasizes the importance of using clear and descriptive names for variables in code. It argues against using confusing or very short names that make it hard for people to understand the code without extra effort. It says that using single letters for variables, except in specific cases like loops, is not a good idea because it makes the code harder to understand. Choosing obscure names just because other names are already taken is also criticized. The passage suggests that even though programmers might be good at figuring out unclear names, it's more professional to prioritize clarity over showing off intelligence. It says that professional programmers know that clear and easy-to-understand code is really important so that others can understand what the code is doing. Overall, it recommends using names that make sense and describe what the variable is for, to make the code easier to read and work with in the long run.

Class Names

This guideline suggests using names for classes and objects in software development that are nouns or noun phrases. For example, names like Customer, WikiPage, Account, and AddressParser are recommended, while names like Manager, Processor, Data, or Info are not recommended. The reason for this advice is probably to make the code clearer and to follow object-oriented principles. Nouns typically represent things or objects in a system. By avoiding

names based on verbs for classes, developers can keep their code consistent and better reflect that classes represent objects, not actions. Following this guideline helps make code easier to understand, read, and maintain in the long term, which improves communication among developers and keeps the code cohesive.

Method Names

The rules provided suggest that when you're naming things like methods, accessors (to get values), mutators (to change values), and predicates (to check conditions) in programming, it's best to keep it clear and consistent. They say to use action words or phrases for method names, like 'postPayment' or 'deletePage', which makes it easier to understand what they do. Accessors should start with 'get', mutators with 'set', and predicates with 'is'. For example, 'getName()', 'setName("Mike")', and 'isPosted()' follow these rules. Also, when you have different ways to create something, it's better to use descriptive names for the ways you can create it, like 'Complex.fromRealNumber(23.0)' instead of 'new Complex(23.0)'. They also suggest making constructors private to encourage using these descriptive ways of creating objects. Overall, these rules help make code easier to read, consistent, and easier to maintain, which makes software development better.

Don’t Be Cute

The passage highlights how important it is to choose clear names for things in code. It warns against using names that are too clever or informal because they can make it hard for others to understand. While funny names might be memorable to some, they might confuse others who don't get the joke. Instead, it suggests using names that clearly describe what something does, so more people can understand it easily. The main idea is to pick names that make it easy to understand what the code does, rather than focusing on being entertaining. This helps with teamwork and keeping the code easy to work with over time.

Pick One Word per Concept

This passage stresses how important it is to use the same names for things in computer programming. It talks about how confusing it can be when different words are used for the same idea in different parts of a program. Remembering which word goes with which idea can slow down your thinking and make it harder to work efficiently. Even though some computer programs can help you find your way around code, they can't always understand what the different words mean. For example, if one part of a program calls something a "controller" and another part calls it a "manager," it can make things messy. So, it's best to stick to one set of names for things throughout your code. This makes it easier for programmers to understand what's going on and work faster.

Don’t Pun

This paragraph emphasizes how important it is to name things clearly and consistently in code. It warns against using the same name for different things, which can make the code confusing. Even though using a different name for each concept might make the code seem cluttered, it's crucial for people to understand what each part of the code does. For example, instead of always using "add" for adding things, it suggests using more precise terms like "insert" or "append" depending on what you're adding. The main aim is to make the code easy to understand without having to spend a lot of time figuring it out, similar to reading a clear and simple book rather than a complicated academic paper.

Use Solution Domains

This paragraph stresses the importance of using words and phrases that other programmers are familiar with when naming things like variables and functions in code. When developers use terms from computer science, like names of algorithms or common patterns, it helps everyone understand the code better. If they use words specific to a certain field, it can confuse people and lead to extra questions. For example, instead of naming something "AccountVisitor," which only makes sense to people who know about the Visitor pattern, it's better to use a more general term like "JobQueue" that everyone on the team can easily understand. This helps teams work together efficiently and makes it easier to read and update the code later on.

Use Problem Domains

This advice suggests that when you're writing software, it's important to use words and phrases that people in the field you're working on understand, rather than using complex technical language. This helps everyone involved in the project understand what's going on and makes it easier for people to work on the code in the future. It's like speaking the same language as the experts in the problem you're trying to solve. Good programmers know how to do this and use it in their code, which helps improve teamwork between technical and non-technical people and makes the software better in the long run.

Add Meaningful Context

The passage highlights how important it is to give clear names to things in software development. It says that we should name classes, functions, and other parts of code in a way that shows what they do. For example, if we have variables for parts of an address, it's better to name them in a way that shows they're related to addresses, like putting a prefix or grouping them inside a class for addresses. It also talks about how sometimes, without the right context, names can be confusing. So, it suggests making smaller, clearer parts of code, like creating a class called GuessStatisticsMessage instead of having a big, confusing function. The passage uses a code example to show how putting variables inside a class not only makes things clearer but also makes the code easier to work with and understand. Overall, it's saying that giving things clear names and organizing code well is really important for making good software.

Don’t Add Gratuitous Context

The paragraph talks about how important it is to choose good names when writing computer programs. It shows that if names aren't chosen carefully, it can cause problems. For example, using long prefixes for class names can make tools harder to use and slow down work. It says that names should be short but still explain what they're for, and they shouldn't repeat unnecessary information. The text gives examples to show how this works and says that shorter names are better as long as they make sense. In general, it says that picking good names is really important for making code easy to read, maintain, and work with efficiently.