Chapter 2: Meaningful Names

Chapter 2 explains the importance of good naming when dealing with variables, files, etc. in code. First, it is necessary to use names that clearly reveal their intent. It is important to express things explicitly so that the code is easily understandable. Also, we should avoid inconsistent naming. When using different names, they should represent different things. We shouldn't just number variables or add noisy words. Instead, make them distinctly identifiable. Since programming languages are still languages, names should be pronounceable. Whether something is easy to pronounce becomes important when discussing the code, as programming is a social activity. Additionally, names should not be too long, making them easy to search for. Except in specific cases, encoding in names should be avoided, as encoded names are often hard to pronounce and prone to typos. Also, avoid using well-known terms to prevent collisions within the scope. For class names, use nouns, and for method names, use verbs. When naming, avoid using overly unusual names. For abstract concepts, stick to one consistent term to maintain context. Creating meaningful context is crucial, and meaningless context should be excluded. Abbreviations should only be used when they are clear.

Chapter 3: Functions

Chapter 3 discusses how to write functions well. The first rule of writing functions is that they should be small. To keep function smalls, it's bet when the if, else, or while statements is a single line made of function calls. Avoiding nested structures enhances readability and comprehension. Additionally, a function should perform only one action very well. It should achieve the one action described by its name through a series of steps. To ensure a function performs one action, it's important that the statements within the function correspond to the same level of abstraction. Mixing different levels of abstraction within a function creates confusion. People generally read code from top to bottom, so it's important to write it so that the levels of abstraction gradually decrease. In cases like switch statements, where multiple actions are unavoidable, put them in low-level classes to avoid code repetition. Naming is also important when writing functions. We should focus on expressing the intended action rather than worrying about the length of the name. The number of arguments in a function should be as small as possible. The more arguments there are, the more combinations need to be tested and the harder it becomes to understand. Reducing the number of arguments through objects or argument lists is important. Another important aspect of writing functions is eliminating side effects. Since a function should only perform one action, side effects should not exist. It's important to separate commands from queries when writing functions. From this perspective, instead of returning error codes, exceptions should be used to separate concerns. Reducing duplication is also important. Functions should be written structurally, with clear inputs and outputs.