http://www.yegor256.com/2015/01/12/compound-name-is-code-smell.html

A Compound Name Is a Code Smell

12 January 2015 modified on 8 February 2015 Yegor Bugayenko

Do you name variables like `textLength`, `table_name`, or `current-user-email`? All three are compound names that consist of more than one word. Even though they look more descriptive than `name`, `length`, or `email`, I would strongly recommend avoiding them. I believe a variable name that is more complex than a noun is a code smell. Why? Because we usually give a variable a compound name when its scope is so big and complex that a simple noun would sound ambiguous. And a big, complex scope is an obvious code smell.

© The Meaning of Life (1983) by Terry Jones and Terry Gilliam

The scope of a variable is the place where it is visible, like a method, for example. Look at this Ruby class:

```
class CSV
  def initialize(csvFileName)
    @fileName = csvFileName
  end
  def readRecords()
    File.readLines(@fileName).map |csvLine|
      csvLine.split(',')
    end
  end
end
```

The visible scope of variable `csvFileName` is method `initialize()`, which is a constructor of the class `CSV`. Why does it need a compound name that consists of three words? Isn't it already clear that a single-argument constructor of class `CSV` expects the name of a file with comma-separated values? I would rename it to `file`.

Next, the scope of `@fileName` is the entire `CSV` class. Renaming a single variable in the class to just `@file` won't introduce any confusion. It's still clear what file we're dealing with. The same situation exists with the `csvLine` variable. It is clear that we're dealing with CSV lines here. The `csv` prefix is just a redundancy. Here is how I would refactor the class:

```
class CSV
  def initialize(file)
    @file = file
  end
  def records()
    File.readLines(@file).map |line|
      line.split(',')
    end
  end
end
```

Now it looks clear and concise.

If you can't perform such a refactoring, it means your scope is too big

and/or too complex. An ideal method should deal with up to five variables, and an ideal class should encapsulate up to five properties.

If we have five variables, can't we find five nouns to name them?

Adam and Eve didn't have second names. They were unique in Eden, as were many other characters in the Old Testament. Second and middle names were invented later in order to resolve ambiguity. To keep your methods and classes clean and solid, and to prevent ambiguity, try to give your variables and methods unique single-word names, just like Adam and Eve were named by you know who :)