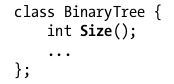
**Chapter 2 Packing Information into Names**

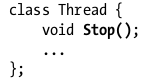
* we like to think of names as tiny comments.

**Choose Specific Words**

* the word “get” is very unspecific, as in this example:
  + def GetPage(url):
* it is hard to determine if it will get a page from;
  + a local cache
  + a database
  + or Internet
    - more specific name in getting page in internet is FetchPage() or DownloadPage().
* example of a BinaryTree class:



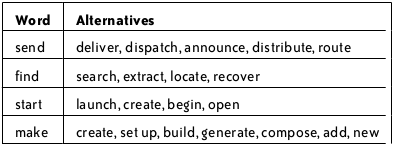
* size() doesn’t convey much information.
* more specific name would be:
  + Height(), NumNodes(), or MemoryBytes().
* another example, suppose a Thread class:



* the name Stop() is okay depending on what it does.
* may be called Kill() instead, implies it’s a heavyweight operation that can’t be undone.
* or may be Pause(), if can be Resume()

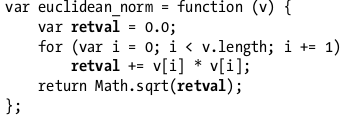
**Finding The Colorful Words**

* some examples of a word and more colorful versions that might apply:

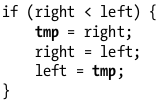


**Avoid Generic Names Like tmp and retval**

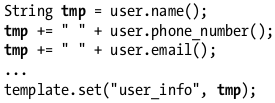
* instead of using a name like this, pick a name that describes the entity’s value or purpose.
* example of JavaScript that uses retval:



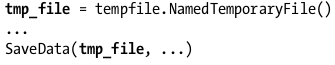
* retval doesn’t contain much information other than a return value.
* better name would describe the purpose of the variable or the value it contains
* for this instance the better name is sum\_squares
* generic name that contain meaning:



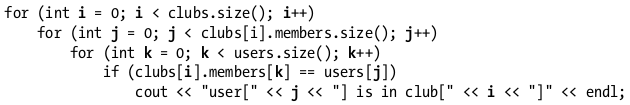
* the variable’s sole purpose is temporary storage, with lifetime of only a few lines.
* the name tmp conveys specific meaning to the reader that this variable has no other duties.
* here’s a case of bad use of tmp:



* a name like user\_info would be more descriptive.
* tmp should be in the name, but just as a part of it:



* if the variable was named tmp it will not be clear of the tmp is a file, filename or a data being written.
* name tmp should be used only in cases when being short-lived and temporary is the most important part of the variable.
* names like i, k iter and it are commonly used as indices and loop iterators so don’t use this names for other purpose.
* but there are better iterator names than mentioned:



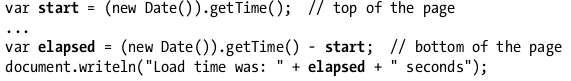
* using more precise names may help:



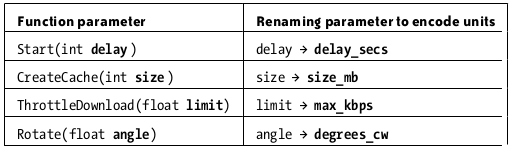
* if you are going to use a generic name like tmp, it, or retval have a good reason for doing so.

**Prefer Concrete Names over Abstract Names**

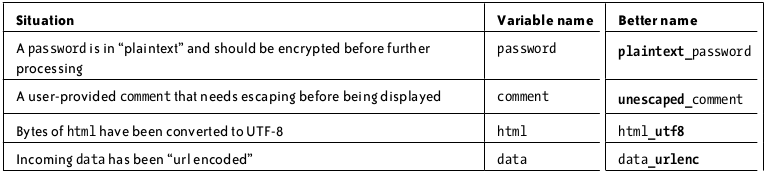
* when naming a variable, function or other element, describe it concretely not abstractly.
* ServerCanStart() → CanListenOnPort()
* if there’s important information about the variable reader must now and you can attach an extra “word” to the name.
* suppose you had a variable that contained hexadecimal string
* string id;
* it’s better to use hex\_id instead if it’s important for the reader to remember the ID’s format.
* if variable is measurement, it’s helpful to encode the units into the variable's name.



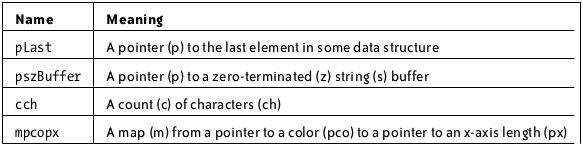
* get time may return seconds or milliseconds so though it looks correct it is better to include the measurement: start\_ms



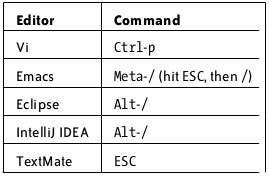
* extra information is not limited to values with units. do it any time there’s something dangerous or surprising about the variable.
* examples where when extra information should be encoded in the name:



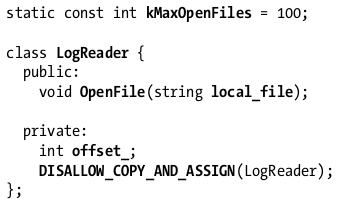
* do not use attributes like unescaped\_or\_utf8 for every variable in program.
* Hungarian notation is a system of naming used widely inside Microsoft. It encodes “type” of every variable into the name’s prefix.



* when picking a good name, there’s an implicit constraint that the name shouldn’t be too long. the longer a name is, the harder it is to remember, and more space it consumes on the screen.
* shorter names are okay for shorter scope
* typing long names is not a problem anymore because of programming editor that has a word completion built in.
* commands:



* using acronyms and abbreviations are only good if you think that the new member in the project would understand that.
* there are words inside a name that can be removed without losing any information at all.
* use underscores, dashes, and capitalization can pack more information in a name.



* having different formats for different entities is like a form of syntax highlighting it helps you read the code more easily.
* class names uses camelCase format and variables uses low\_separated.
* class member variables must end with an underscore like offset\_.
* depending on the language or project, there may be other formatting conventions you can use to make names contain more information.
* in javascript constructor should be capitalized and that ordinary functions should start with a lowercase letter.



* when giving a HTML tag in an id or class attribute, both underscores and dashes are valid characters to use in the value. possible convention is to use underscores to separate words in IDs and dashes to separate words in classes:
* 
* pack information into the names.
* use specific words
* avoid generic names
* use concrete names
* attach important details
* use longer names for larger scopes
* use capitalization underscores, etc in meaningful way