**Meaningful names**

**Use names that reveal intentions:** In software names are ubiquitous, appearing in variables, functions, arguments, classes, and packages. Choosing correct names takes time, but it saves work and can change by finding better ones, indicating why it exists, what it does and how it is used, making it easier to understand and modify the code.

**Avoid misinformation:** False clues that hinder the meaning of the code, words whose meaning is not the sea, names with minimal variations, and incorrect spelling.

**Make meaningful distinctions:** Create code for a compiler or interpret, use the same name to refer to 2 different elements of the same scope, write them incorrectly prevent compilation, if they must be different, they must have different meanings, therefore, do not use words additional or prefixes.

**Use names that can be pronounced:** Create pronounceable names, if I can't explain it, I can't explain it, since programming is a social activity.

**Use searchable names:** If a variable or constant is used at various points in the code, you must assign a searchable name or in short methods.

**Avoid encoding:** encoding type or variation information in a difficult decoding name, unpronounceable affected encoding names, and misspelled.

**Hungarian notation (HN):** Currently HN and other forms of type encoding make it difficult to change the name or type of a variable or class, readability of the code causing the encoding system to confuse the reader.

**Member prefixes:** m\_ is unnecessary as a prefix to variable, class and function names, you can use an editing environment that highlights members to distinguish them.

**Interfaces and implementations:** Assign a name to a specific class, IShapeFactory or ShapeFactory, and encode using ShapeFactory since it is an excess of information.

**Avoid Mental Mapping:** What differentiates an intelligent programmer from a professional since a professional write a code that he can understand.

**Class Names:** The most important thing at this point is that to name a class is that it should not be a verb.

**Method Names:** These names must be verbs or verb phrases and have the prefix get or set and in case the constructors are overloaded, names that describe the arguments must be used.

**Pick One Word per Concept:** You should choose a word for an abstract concept and keep this as it can be confusing when remembering which method goes with what class therefore a consistent lexicon is a great help for this.

**Don’t Pun:** We must avoid making word games or using the same word for two uses. An example of this would be the use of adding and for a new method, for example, insert or append should be used.

**Domain Names:** We must remember that those who are going to get their code will be programmers, so technical names should be used since we do not want our partner to ask the client the meaning of each word.

**Don’t Add Gratuitous Context:** This refers to the fact that you should not add more context to a name than necessary; in fact, a shorter name is usually better than a long one as long as it is clear.

**Final Words:** Many people often have problems when putting the names since they must have a good description and a cultural background, the best way to solve this is usually the refactoring this will make them continue to pay in the long term