

Appendix C

IDENTIFIER NAMING: WRITING SELF-COMMENTING CODE

IDENTIFIER NAMING: WRITING SELF-COMMENTING CODE

Self-commenting code is an identifier-naming technique you can use to effectively manage both physical and conceptual complexity. An identifier is a sequence of characters or digits used to form the names of entities used in your program. Examples of program entities include classes, constants, variables, and methods. All you have to do to write self-commenting code is to 1) give meaningful names to your program entities, and 2) adopt a consistent identifier naming convention. C# allows unlimited-length identifier names, so you can afford to be descriptive.

BENEFITS OF SELF-COMMENTING CODE

The benefits of using self-commenting code are many. First, self-commenting code is easier to read. Code that's easy to read is easy to understand. If your code is easy to read and understand, you will spend much less time tracking down logic errors or just plain mistakes.

CODING CONVENTION

Self-commenting code is not just for students. Professional programmers (*real professionals, not the cowboys!*) write self-commenting code because their code is subject to peer review. To ensure all members of a programming team can read and understand each other's code, the team adopts a coding convention. The coding convention specifies how to form entity names, along with how the code must be formatted.

When you write programs to satisfy the exercises in *C# For Artists* I recommend you adopt the following identifier naming conventions:

CLASS NAMES

Class names should start with an initial capital letter. If the class name contains multiple words, then capitalize the first letter of each subsequent word used to form the class name. This is referred to as *camel case*. Table Appendix C-1 offers several examples of valid class names:

Class Name	Comment
Student	One-syllable class name. First letter capitalized.
Engine	Another one-syllable class name.
EngineController	Two-syllable class name. First letter of each word capitalized.
HighOutputEngine	Three-syllable class name. First letter of each word capitalized.

Table Appendix C-1: Class Naming Examples

CONSTANT NAMES

Constants represent values in your code that cannot be changed once initialized. Constant names should describe the values they contain, and should consist of all capital letters to set them apart from variables. Connect each word of a multiple-word constant by an underscore character. Table Appendix C-2 gives several examples of constant names:

Constant Name	Comment
PI	Single-word constant in all caps. Could be the constant value of π .
MAX	Another single-word constant, but max what?
MAX_STUDENTS	Multiple-word constant separated by an underscore character.
MINIMUM_ENROLLMENT	Another multiple-word constant.

Table Appendix C-2: Constant Naming Examples

VARIABLE NAMES

Variables represent values in your code that can change while your program is running. Variable names should be formed from lower-case characters to set them apart from constants. Variable names should describe the values they contain. Table Appendix C-3 shows a few examples of variable names:

Variable Name	Comment
size	Single-word variable in lower-case characters. But size of what?
array_size	Multiple-word variable, each word joined by underscore character.
current_row	Another multiple-word variable.
mother_in_law_count	Multiple-word variable, each word joined by an underscore character.

Table Appendix C-3: Variable Naming Examples

You may want to start field variable names with an underscore character to make them easy to spot in your code.

METHOD NAMES

A method represents a named series of statements that perform some action when called in a program. Since methods invoke actions, their names should be formed from action words (*verbs*) that describe what they do. Begin method names with an upper-case character, and capitalize each subsequent word of a multiple-word method. The only exception to this naming rule is for constructor methods, which must be exactly the same name as the class in which they appear. Table Appendix C-4 gives some examples.

Method Name	Comment
PrintFloor	Multiple-word method name. The first word is an action word.
SetMaxValue	Multiple-word method name. This is an example of a mutator method.
GetMaxValue	Another multiple-word method name. This is an example of an accessor method.
Start	A single-word method name.

Table Appendix C-4: Method Naming Examples

PROPERTY NAMES

The property name should accurately reflect the nature of the property. Table Appendix C-5 offers several examples.

Property Name	Comment
StudentCount	Multiple-word property name.
MaxValue	Multiple-word property name.
FirstName	Another multiple-word property name.
Length	A single-word Property name.

Table Appendix C-5: Property Naming Examples

