Coding Conventions

Sang Shin
www.jPassion.com
"Learn with jPassion!"

Topics

- What are and Why coding conventions?
- Coding convention examples
- Naming conventions
- Programming practices

What are “Coding Conventions”?

What are Coding Conventions?

- Set of guidelines that recommend programming style, practices and methods for each aspect of a piece program written for the purpose of enhancing
  - Readability
  - Software maintenability
- These conventions usually cover
  - File organizations
  - Indentation
  - Comments
  - Declarations
  - Statements
  - White spaces,
  - Naming conventions

Why Coding Conventions?

- 80% of the lifetime cost of a piece of software goes to maintenance
- Hardly any software is maintained for its whole life by the original author
- Code conventions improve the readability of the software, allowing readers to understand new code more quickly and thoroughly

Code Convention Examples

File Organization

- Files longer than 2000 lines are cumbersome and should be avoided
- Order of parts that make up class/interface declaration
  - Class/Interface documentation comment
  - Class or interface statements
  - Class (static) variables
  - Instance variables
  - Constructors
  - Methods

Wrapping Arithmetic Expression

- Following are two examples of breaking an arithmetic expression

```
// Good practice - since the break occurs outside the parenthesized
// expression, which is at a higher level
longName1 = longName2 * (longName3 + longName4 - longName5)
          + 4 * longname6;

// Bad practice
longName1 = longName2 * (longName3 + longName4
                  - longName5) + 4 * longname6; // AVOID
```

Line Wrapping

- Line wrapping for if statements should generally use the 8-space rule, since conventional (4 space) indentation makes seeing the body difficult

```
// Bad practice
if ((condition1 && condition2)
    || (condition3 && condition4)
    ||!(condition5 && condition6)) { //BAD WRAPS
    doSomethingAboutIt();           //MAKE THIS LINE EASY TO MISS
}

// Good practice #1
if ((condition1 && condition2)
        || (condition3 && condition4)
        ||!(condition5 && condition6)) {
    doSomethingAboutIt();
}

// Good practice #2
if ((condition1 && condition2) || (condition3 && condition4)
        ||!(condition5 && condition6)) {
    doSomethingAboutIt();
}
```

Class and Interface Declarations

- No space between a method name and the parenthesis "(" starting its parameter list
- Open brace "{" appears at the end of the same line as the declaration statement
- Closing brace "}" starts a line by itself indented to match its corresponding opening statement, except when it is a null statement the "}" should appear immediately after the "{"

```
class Sample extends Object {
    int ivar1;
    int ivar2;

    Sample(int i, int j) {
        ivar1 = i;
        ivar2 = j;
    }

    int emptyMethod() {}
    ...
}
```

return Statement

- A return statement with a value should not use parentheses unless they make the return value more obvious in some way. Example:

```
return;

return myDisk.size();

return (size ? size : defaultSize);
```

Naming Conventions

Why Naming Conventions?

- Naming conventions make programs more understandable by making them easier to read.
- They can also give information about the function of the identifier-for example, whether it's a constant, package, or class-which can be helpful in understanding the code.

Naming Conventions

- Packages
  - All lower case

  com.sun.eng
  edu.cmu.cs.bovik.cheese

- Classes
  - Should be nouns
  - Mixed case with the first letter of each internal word capitalized (Camel)

  class Raster;
  class ImageSprite;

Naming Conventions

- Interfaces
  - Should be capitalized like class names

interface RasterDelegate;
interface Storing;

- Methods
  - Should be verbs, in mixed case with the first letter lowercase, with the first letter of each internal word capitalized.

run();
runFast();
getBackground();

Naming Conventions

- Variables
  - Variable names should be short yet meaningful.
  - The choice of a variable name should be mnemonic- that is, designed to indicate to the casual observer the intent of its use.
  - One-character variable names should be avoided except for temporary "throwaway" variables. Common names for temporary variables are i, j, k, m, and n for integers; c, d, and e for characters.

```
int     i;
char    c;
float   myWidth;
```

Naming Conventions

- Constants
  - The names of variables declared class constants and of ANSI constants should be all uppercase with words separated by underscores ("_")

```
static final int MIN_WIDTH = 4;
static final int MAX_WIDTH = 999;
static final int GET_THE_CPU = 1;
```

Programming Practices

Access to Instance/Class Variables

- Don't make any instance or class variable public without good reason
  - Enables the encaptulation
- One example of appropriate public instance variables is the case where the class is essentially a data structure, with no behavior. In other words, if you would have used a struct instead of a class (if Java supported struct), then it's appropriate to make the class's instance variables public

Referring to Class Variables/Methods

- Avoid using an object to access a class (static) variable or method. Use a class name instead.

```
classMethod();                  //OK
AClass.classMethod();           //OK
anObject.classMethod();         //AVOID!
```

Variable Assignments

- Avoid assigning several variables to the same value in a single statement. It is hard to read

```
fooBar.fChar = barFoo.lchar = 'c'; // AVOID!
```

- Do not use embedded assignments in an attempt to improve run-time performance. This is the job of the compiler.

```
d = (a = b + c) + r;      // AVOID!
a = b + c;                // OK
d = a + r;                // OK
```

Thank you!

Check JavaPassion.com Codecamps!
http://www.javapassion.com/codecamps
"Learn with Passion!"