JAVA Naming Conventions

Algorithms & Data Structures (ADS)

Contents

Files

- Java source files have the following ordering:

1. Beginning comments
2. Package and Import statements
3. Class and interface declarations

Comments

All source files should begin with a c-style comment that lists the class name, version information, date, and copyright notice:

```
/*
* Classname
*
* Version information
*
* Date
*
* Copyright notice
*
*/
```

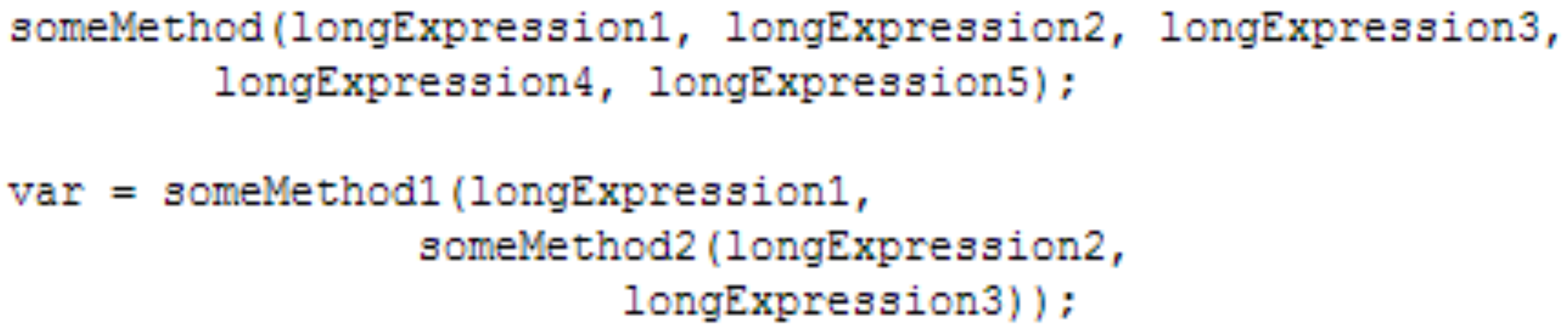
The first non-comment line of most Java source files is a package statement. After that, import statements can follow. For example:

```
package java.awt;
import java.awt.peer.CanvasPeer;
```

Indentation

- Avoid lines longer than 80 characters, since they're not handled well by many terminals and tools.
- **Wrapping Lines**
- When an expression will not fit on a single line, break it according to these general principles:
  - Break after a comma.
  - Break before an operator.
  - Prefer higher-level breaks to lower-level breaks.
  - Align the new line with the beginning of the expression at the same level on the previous line.
  - If the above rules lead to confusing code or to code that's squished up against the right margin, just indent 8 spaces instead.

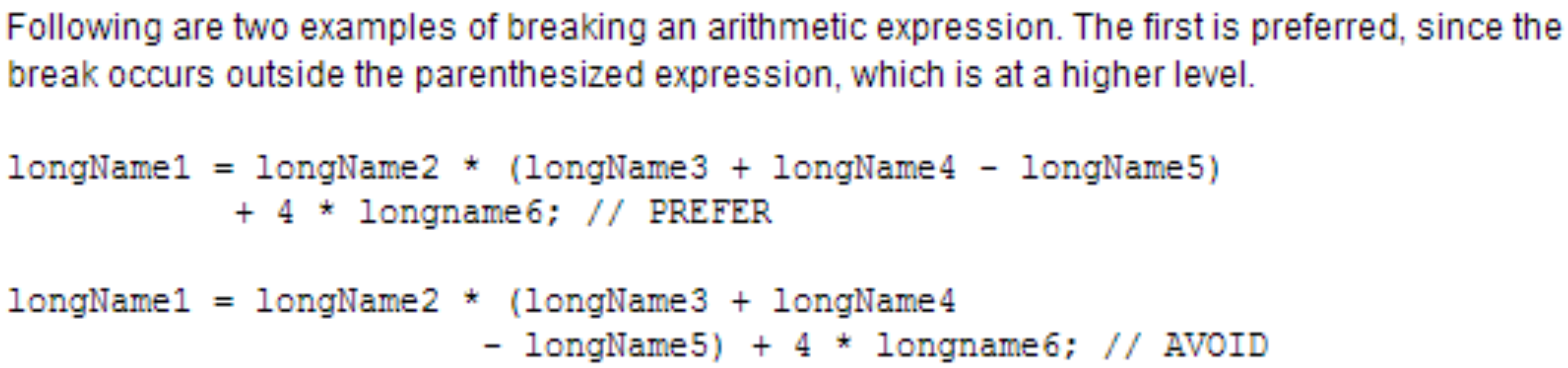
Indentation. Examples



```
someMethod(longExpression1, longExpression2, longExpression3,
        longExpression4, longExpression5);

var = someMethod1(longExpression1,
                someMethod2(longExpression2,
                        longExpression3));
```

Indentation. Examples (II)

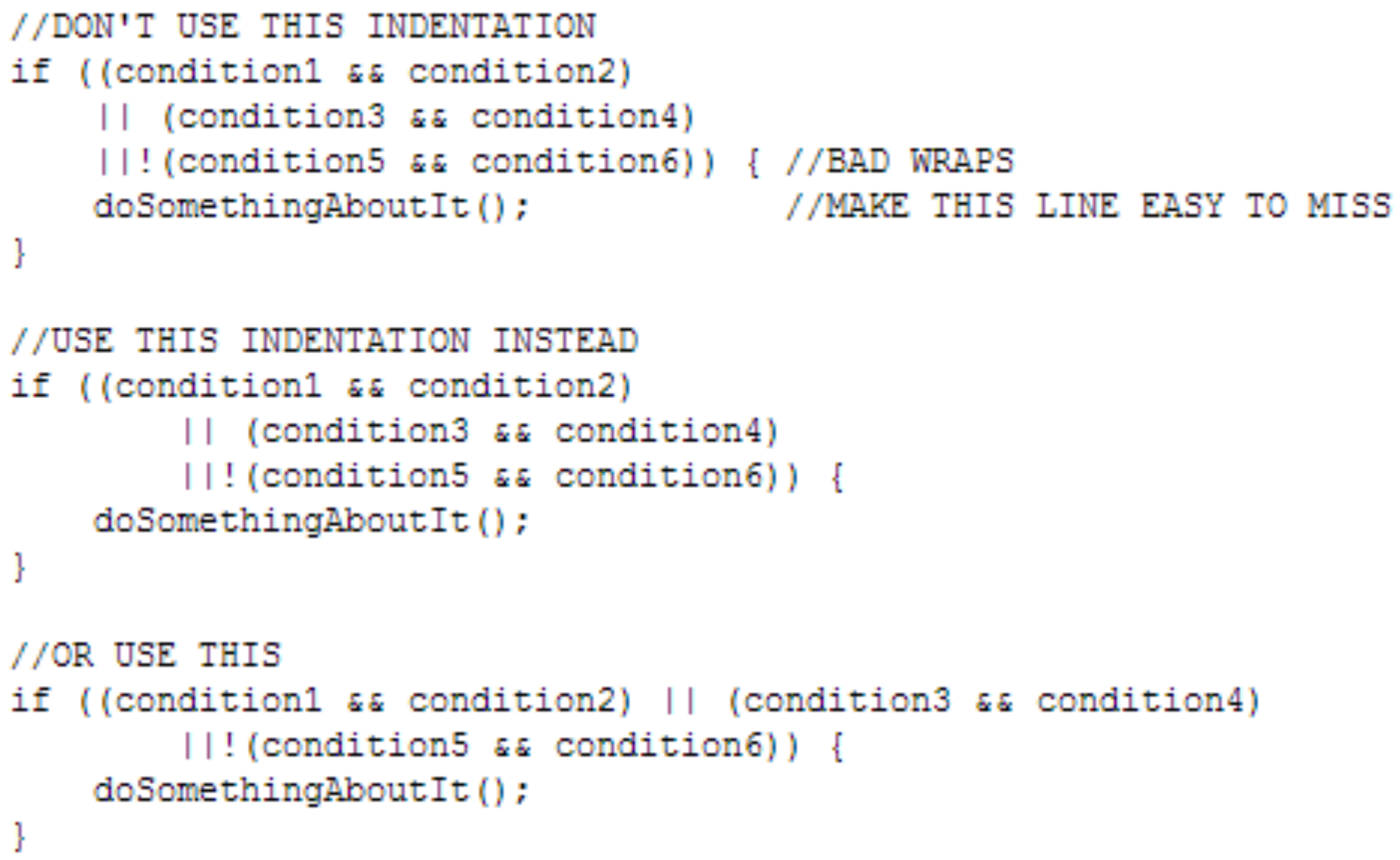


Following are two examples of breaking an arithmetic expression. The first is preferred, since the break occurs outside the parenthesized expression, which is at a higher level.

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

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

Indentation. Examples (III)



```
//DON'T USE THIS INDENTATION
if ((condition1 && condition2)
    || (condition3 && condition4)
    ||!(condition5 && condition6)) { //BAD WRAPS
    doSomethingAboutIt();            //MAKE THIS LINE EASY TO MISS
}

//USE THIS INDENTATION INSTEAD
if ((condition1 && condition2)
        || (condition3 && condition4)
        ||!(condition5 && condition6)) {
    doSomethingAboutIt();
}

//OR USE THIS
if ((condition1 && condition2) || (condition3 && condition4)
        ||!(condition5 && condition6)) {
    doSomethingAboutIt();
}
```

Declarations

- One declaration per line is recommended since it encourages commenting. In other words,

```
int level; // indentation level
int size; // size of table
```

- is preferred over

```
int level, size;
```

- Do not put different types on the same line. Example:

```
int foo, foo array[]; //WRONG!
```

Declarations

- Try to initialize local variables where they're declared. The only reason not to initialize a variable where it's declared is if the initial value depends on some computation occurring first. Example:

```
int count;
...
myMethod() {
   if (condition) {
         int count = 0; // AVOID!
   …}
…}
```

Declarations

- Classes and interfaces
  - 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 "{"

Naming conventions

Packages	The prefix of a unique package name is always written in all-lowercase ASCII letters and should be one of the top-level domain names, currently com, edu, gov, mil, net, org, or one of the English two-letter codes identifying countries as specified in ISO Standard 3166, 1981. Subsequent components of the package name vary according to an organization's own internal naming conventions. Such conventions might specify that certain directory name components be division, department, project, machine, or login names.	com.sun.eng com.apple.quicktime.v2 edu.cmu.cs.bovik.cheese

Naming conventions

Classes

Class names should be nouns, in mixed case with the first letter of each internal word capitalized. Try to keep your class names simple and descriptive. Use whole words-avoid acronyms and abbreviations (unless the abbreviation is much more widely used than the long form, such as URL or HTML).	class Raster;
	class ImageSprite;

Interfaces

Interface names should be capitalized like class names.	interface RasterDelegate;
	interface Storing;

Methods

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

Except for variables, all instance, class, and class constants are in mixed case with a lowercase first letter. Internal words start with capital letters. Variable names should not start with underscore _ or dollar sign $ characters, even though both are allowed.

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

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.

Naming conventions

Constants

The names of variables declared class constants and of ANSI constants should be all uppercase with words separated by underscores ("_"). (ANSI constants should be avoided, for ease of debugging.)

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