CAMBRIDGE

Cambridge University Press
978-0-521-77768-1 - The Elements of Java™ Style
Al Vermeulen, Scott W. Ambler, Greg Bumgardner, Eldon Metz, Trevor Misfeldt, Jim Shur and Patrick Thompson
Excerpt
More information

1. General Principles

While it is important to write software that performs well, many other issues should concern the professional Java developer. All *good* software performs well. But *great* software, written with style, is predictable, robust, maintainable, supportable, and extensible.

1. *Adhere to the style of the original.*

When modifying existing software, your changes should follow the style of the original code.[1] Do not introduce a new coding style in a modification, and do not attempt to rewrite the old software just to make it match the new style. The use of different styles within a single source file produces code that is more difficult to read and comprehend. Rewriting old code simply to change its style may result in the introduction of costly yet avoidable defects.

2. *Adhere to the Principle of Least Astonishment.*

The *Principle of Least Astonishment* suggests you should avoid doing things that will surprise a user of your software. This implies the means of interaction and the behavior exhibited by your software must be predictable and consistent,[2] and, if not, the documentation must clearly identify and justify any unusual patterns of use or behavior.

To minimize the chances that a user will encounter something surprising in your software, you should emphasize the following characteristics in the design, implementation, and documentation of your Java software:

© in this web service Cambridge University Press www.cambridge.org

CAMBRIDGE

Cambridge University Press
978-0-521-77768-1 - The Elements of Java™ Style
Al Vermeulen, Scott W. Ambler, Greg Bumgardner, Eldon Metz, Trevor Misfeldt, Jim Shur and Patrick Thompson
Excerpt
More information

Simplicity	Build simple classes and simple methods. Determine how much you need to do to meet the expectations of your users.
Clarity	Ensure each class, interface, method, variable, and object has a clear purpose. Explain where, when, why, and how to use each.
Completeness	Provide the minimum functionality that any reasonable user would expect to find and use. Create complete documentation; document all features and functionality.
Consistency	Similar entities should look and behave the same; dissimilar entities should look and behave differently. Create and apply standards whenever possible.
Robustness	Provide predictable documented behavior in response to errors and exceptions. Do not hide errors and do not force clients to detect errors.

3. Do it right the first time.

Apply these rules to any code you write, not just code destined for production. More often than not, some piece of prototype or experimental code will make its way into a finished product, so you should anticipate this eventuality. Even if your code never makes it into production, someone else may still have to read it. Anyone who must look at your code will appreciate your professionalism and foresight at having consistently applied these rules from the start.

© in this web service Cambridge University Press www.cambridge.org

CAMBRIDGE

Cambridge University Press
978-0-521-77768-1 - The Elements of Java™ Style
Al Vermeulen, Scott W. Ambler, Greg Bumgardner, Eldon Metz, Trevor Misfeldt, Jim Shur and Patrick Thompson
Excerpt
More information

4. Document any deviations.

No standard is perfect and no standard is universally applicable. Sometimes you will find yourself in a situation where you need to deviate from an established standard.

Before you decide to ignore a rule, you should first make sure you understand why the rule exists and what the consequences are if it is not applied. If you decide you must violate a rule, then document why you have done so.

This is the *prime directive.*

1 Jim Karabatsos. "When does this document apply?" In "Visual Basic Programming Standards." (GUI Computing Ltd., 22 March 1996). Accessed online at http://www.gui.com.au/jkcoding.htm, Aug 1999.

2 George Brackett. "Class 6: Designing for Communication: Layout, Structure, Navigation for Nets and Webs." In "Course T525: Designing Educational Experiences for Networks and Webs." (Harvard Graduate School of Education, 26 August 1999). Accessed online at http://hgseclass.harvard.edu/t52598/classes/class6/, Aug 1999.

© in this web service Cambridge University Press www.cambridge.org

CAMBRIDGE

Cambridge University Press
978-0-521-77768-1 - The Elements of Java™ Style
Al Vermeulen, Scott W. Ambler, Greg Bumgardner, Eldon Metz, Trevor Misfeldt, Jim Shur and Patrick Thompson
Excerpt
More information

© in this web service Cambridge University Press www.cambridge.org

CAMBRIDGE

Cambridge University Press
978-0-521-77768-1 - The Elements of Java™ Style
Al Vermeulen, Scott W. Ambler, Greg Bumgardner, Eldon Metz, Trevor Misfeldt, Jim Shur and Patrick Thompson
Excerpt
More information

2. Formatting Conventions

5. *Indent nested code.*

One way to improve code readability is to group individual statements into block statements and uniformly indent the content of each block to set off its contents from the surrounding code.

If you generate code using a Java development environment, use the indentation style produced by the environment. If you are generating the code by hand, use two spaces to ensure readability without taking up too much space:

```
class MyClass {
··void function(int arg) {
····if (arg < 0) {
······for (int index = 0; index <= arg; index++) {
········// ...
····· }
···· }
·· }
}
```

In addition to indenting the contents of block statements, you should also indent the statements that follow a label to make the label easier to notice:

```
void function(int arg) {
··loop:
····for (int index = 0; index <= arg; index++) {
·····switch (index) {
```


© in this web service Cambridge University Press www.cambridge.org

CAMBRIDGE

Cambridge University Press
978-0-521-77768-1 - The Elements of Java™ Style
Al Vermeulen, Scott W. Ambler, Greg Bumgardner, Eldon Metz, Trevor Misfeldt, Jim Shur and Patrick Thompson
Excerpt
More information


```
        ........case 0:
        ..........//...
        ..........break loop; // exit the for statement
        ........default:
        ..........//...
        ..........break; // exit the switch statement
        ......}
        ....}
    }
```

Locate the opening brace '{' of each block statement in the last character position of the line that introduced the block. Place the closing brace '}' of a block on a line of its own, aligned with the first character of the line that introduced the block. The following examples illustrate how this rule applies to each of the various Java definition and control constructs.

Class definitions:

```
public class MyClass {
  ...
}
```

Inner class definitions:

```
public class OuterClass {
  ...
  class InnerClass {
    ...
  }
  ...
}
```

Method definitions:

```
void method(int j) {
  ...
}
```

Static blocks:

```
static {
  ...
}
```

© in this web service Cambridge University Press www.cambridge.org

CAMBRIDGE

Cambridge University Press
978-0-521-77768-1 - The Elements of Java™ Style
Al Vermeulen, Scott W. Ambler, Greg Bumgardner, Eldon Metz, Trevor Misfeldt, Jim Shur and Patrick Thompson
Excerpt
More information

For-loop statements:

```
for (int i = 0; i <= j; i++) {
  ...
}
```

If and else statements:

```
if (j < 0) {
  ...
}
else if (j > 0) {
  ...
}
else {
  ...
}
```

Try, catch, and finally blocks:

```
try {
  ...
}
catch (Exception e) {
  ...
}
finally {
  ...
}
```

Switch statements:

```
switch (value) {
  case 0:
    ...
    break;
  default:
    ...
    break;
}
```

Anonymous inner classes:

```
button.addActionListener(
  new ActionEventListener() {
```

© in this web service Cambridge University Press www.cambridge.org

CAMBRIDGE

Cambridge University Press
978-0-521-77768-1 - The Elements of Java™ Style
Al Vermeulen, Scott W. Ambler, Greg Bumgardner, Eldon Metz, Trevor Misfeldt, Jim Shur and Patrick Thompson
Excerpt
More information


```
      public void actionPerformed() {
        ...
      }
    }
  )
```

While statements:

```
while (++k <= j) {
  ...
}
```

Do-while statements:

```
do {
  ...
} while (++k <= j);
```

If you are managing a development team, do not leave it up to individual developers to choose their own indentation amount and style. Establish a standard indentation policy for the organization and ensure that everyone complies with this standard.

Our recommendation of two spaces appears to be the most common standard, although your organization may prefer three or even four spaces.

6. Break up long lines.

While a modern window-based editor can easily handle long source code lines by scrolling horizontally, a printer must truncate, wrap, or print on separate sheets any lines that exceed its maximum printable line width. To ensure your source code is still readable when printed, you should limit your source code line lengths to the maximum width your printing environment supports, typically 80 or 132 characters.

First, do not place multiple statement expressions on a single line if the result is a line that exceeds your maximum allowable line length. If two statement expressions are placed on one line:

© in this web service Cambridge University Press www.cambridge.org

CAMBRIDGE

Cambridge University Press
978-0-521-77768-1 - The Elements of Java™ Style
Al Vermeulen, Scott W. Ambler, Greg Bumgardner, Eldon Metz, Trevor Misfeldt, Jim Shur and Patrick Thompson
Excerpt
More information


```
double x = Math.random(); double y = Math.ran
dom(); // Too Long!
```

Then introduce a new line to place them on separate lines:

```
double x = Math.random();
double y = Math.random();
```

Second, if a line is too long because it contains a complex expression:

```
double length = Math.sqrt(Math.pow(Math.random(),
2.0) + Math.pow(Math.random(), 2.0)); // Too Long!
```

Then subdivide that expression into several smaller subexpressions. Use a separate line to store the result produced by an evaluation of each subexpression into a temporary variable:

```
double xSquared = Math.pow(Math.random(),2.0);
double ySquared = Math.pow(Math.random(),2.0);
double length = Math.sqrt(xSquared+ySquared);
```

Last, if a long line cannot be shortened under the previous two guidelines, then break, wrap, and indent that line using the following rules:

Step one

If the top-level expression on the line contains one or more commas:

```
double length = Math.sqrt(Math.pow(x, 2.0),Math
.pow(y,2.0)); // Too Long!
```

Then introduce a line break after each comma. Align each expression following a comma with the first character of the expression proceeding the comma:

```
double length = Math.sqrt(Math.pow(x,2.0),
                          Math.pow(y,2.0));
```

Step two

If the top-level expression on the line contains no commas:

```
class MyClass {
  private int field;
```

© in this web service Cambridge University Press www.cambridge.org

CAMBRIDGE

Cambridge University Press
978-0-521-77768-1 - The Elements of Java™ Style
Al Vermeulen, Scott W. Ambler, Greg Bumgardner, Eldon Metz, Trevor Misfeldt, Jim Shur and Patrick Thompson
Excerpt
More information


```
    ...
    boolean equals(Object obj) {
      return this == obj || (obj instanceof MyClass
  && this.field == obj.field); // Too Long!
    }
    ...
  }
```

Then introduce a line break just before the operator with the lowest precedence or, if more than one operator of equally low precedence exists between each such operator:

```
  class MyClass {
    private int field;
    ...
    boolean equals(Object obj) {
      return this == obj
             || (this.obj instanceof MyClass
                 && this.field == obj.field);
    }
    ...
  }
```

Step three

Reapply steps one and two, as required, until each line created from the original statement expression is less than the maximum allowable length.

7. Include white space.

White space is the area on a page devoid of visible characters. Code with too little white space is difficult to read and understand, so use plenty of white space to delineate methods, comments, code blocks, and expressions clearly.

Use a single space to separate:

- A right parenthesis ')' or curly brace '}' and any keyword that immediately follows, a keyword and any left paren-

© in this web service Cambridge University Press www.cambridge.org