Coding Standards

Classes

Each Java source file contains a single public class or interface. (3.1)
Java source files have the following ordering: Package and Import statements, beginning comments, Class and Interface declarations. (JD)
The first non-comment line is a package statement (when applicable). After that, import statements can follow. (3.1.2)
The declarations in a class or interface are ordered as follows: Class/interface documentation comment, class or interface statement, Class/interface implementation comment, Class (static) variables and constants, Instance variables, Constructors, Methods (3.1.3)

Layout
Four spaces should be used as the unit of indentation. Never place tab characters in source code files.
Avoid lines longer than 80 characters. (4.1)
When an expression will not fit on a single line, break it according to the principles in Sun standard 4,2.
Each line contains at most one simple statement. (7.1)
Adhere to the rules for compound statements (statements that contain lists of statements enclosed in braces). (7.2)
Adhere to the forms given in Section 7 for if, for, while, switch, and try statements, with the exception that open-close brace pairs must always be vertically aligned. Dr. Dalbey provides a tool that will reformat code written in the Sun style to code which adheres to this requirement. (JD)
Blank lines improve readability by setting off sections of code that are logically related.(8.1)
Two blank lines are used between sections of a source file and between class and interface definitions.(8.1)
One blank line is used in the following circumstances:(8.1)

- Between methods
- Between the local variables in a method and its first statement
- Before a block (5.1.1) or single-line (5.1.2) comment
- Between logical sections inside a method to improve readability

Blank spaces should be used in the following circumstances (8.2)

- To separate a keyword followed by a parenthesis.
- A blank space should appear after commas in argument lists.
- All binary operators except. should be separated from their operands by spaces.
- The expressions in a for statement should be separated by blank spaces.
- Casts should be followed by a blank space.

Comments

Implementation-style comments are used to describe implementation.
Documentation-style comments are used to describe the specification of the code, from an implementation-free perspective, that will appear in the javadocs.
All comments should be clear, correctly spelled, grammatically correct, and use simple English phrases. (JD)
Comments should be meaningful and relevant. They should assist the reader in comprehending the code by adding
helpful annotations. Avoid comments that do not significantly enhance readability and comprehension. The comment should not simply restate the code. (JD)
Comment any statement whose intent is not immediately obvious. (5)
Every control structure has a descriptive comment on the preceding line. (JD)
Comments should not be enclosed in large boxes drawn with asterisks or other characters. (5)
Comments should never include special characters such as form-feed and backspace.(5)
Proper use of the four styles of implementation comments: block, single-line, trailing, and end-of-line. (5.1)
Doc comments describe classes, interfaces, constructors, methods, and fields. Each doc comment is set inside the comment delimiters /**…*/, with one comment per class, interface, or method. This comment should appear just before the declaration.(5.2). See also, How to Write Doc Comments for Javadoc.
Each declaration must be on a separate line (6.1) and must have an explanatory comment. Class instance variables should use a javadoc style comment. Variables local to a method may use end-of-line comments.

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

A trailing comment after a close brace may be used to assist matching the closing brace with the corresponding open brace in IF statements, loops, classes, and compound statements. Examples: "end if" "end class" "end loop" (JD)
In most cases, 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. (6.2)
Put declarations only at the beginning of blocks. Don't wait to declare variables until their first use. The one exception to the rule is indexes of for loops, which in Java can be declared in the for statement.(6.3)
When coding Java classes and interfaces, the following formatting rules should be followed (6.4):

- No space between a method name and the parenthesis "(" starting its parameter list

- Open brace "{" appears on the subsequent line indented to match the declaration statement
- Closing brace "}" starts a line by itself indented to match its corresponding opening brace, except when it is a null statement then the "}" should appear immediately after the "{". If a method has a null body you may put the open and close braces on the same line as the method definition.

Naming Conventions (9)

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.

Interfaces Interface names should be capitalized like class names.

Methods Method names should be verbs or verb phrases, in mixed case with the first letter lowercase, with the first letter of each internal word capitalized.

Examples: `run(); isUpperCase(); getBackground(); findTotalCost(int Years);`

Variables Variable names are in mixed case with a lowercase first letter. If the name has multiple words, internal words start with capital letters. Variable names are nouns that succinctly convey as much information as possible about what the object is.

One-character variable names are not allowed. (JD)

Constants The first character is a lower case "k" followed by a mixed case name with no underscores. Example: `static final int kMinWidth = 4;`

Coding Style

No instance or class variables may be public. The only exception is where the class is essentially a data structure with no behavior. (10.2.1)

Numerical constants (literals) should not be coded directly, except for -1, 0, and 1, which can appear in a `for` loop as counter values. (10.3)

Don't assign several variables to the same value in a single statement.(10.4)

Put "catch" and "else" keywords on a separate line not following the preceding close brace.

Use parentheses liberally in expressions involving mixed operators to avoid operator precedence problems. (10.5.1)

Never use `break` or `continue` in a loop or decision.

Never compare a boolean value (or boolean function call) to a boolean constant. (JD)

Make the structure of your program match the intent.(10.5.2)

NEVER DO THIS:

```
if (booleanExpression)
{
  return true;
}
else
```

```
{
  return false;
}
```
It is much clearer and simpler to write: return *booleanExpression*;

The following keywords are used by convention as notes to other programmers (10.5.4):
TBD or TODO to flag sections where something is missing and needs to be completed later.
HACK to flag something that is bogus but works.
FIXME to flag something that is bogus and broken.