Style Guide for Java

Your Java code must comply with the following Style Guide for Java. This document describes the guidelines. Note that the examples discussed in the lectures, labs, and readings may not comply with these guidelines as they were prepared by different people and at different times.

1 Organization/Structure

1. Each class must be in its own file.

   The name of the file and the name of the class must coincide exactly. For example the `Queue` class should be in a file named `Queue.java`. Note that Java is case-sensitive.

2. Each file must end with a blank line.

   This is so that automated testing tools have a place to insert comments/remarks if necessary.

3. Methods within a class must be listed in alphabetical order.

   The only exception to this rule is that all constructors must be listed first. For example, in a `Queue` class, the constructor(s) should be first, the `pop` method should be second, and the `push` method should be third.

4. *All variables must be declared in the smallest possible scope.*
5. *All variables must be explicitly initialized.*
6. Immutatbility should be favoured at all times.
7. Class variables (i.e., static attributes) must be declared before instance variables (i.e., non-static attributes).

   Within each category, private variables must be declared first, followed by protected variables, followed by package variables, followed by public variables.

8. All variables must be declared alphabetically by their class/type.

   For example, variables of type `double` should be declared before variables of type `int` which should, in turn, be declared before variables of type `Node`.

2 Names

1. Class names must start with an uppercase letter.

   In addition, each "word" within a class name should start with an uppercase letter. For example, `TextMessage` and `SimpleTrafficMonitor` are both appropriate class names.

2. The names of "constants" (i.e., static final attributes) must be in all uppercase.

   Further, "words" within a "constant" name must be delimited by an underscore character. For example, `EXTREMELY_UNHEALTHY` is an appropriate name for a "constant".

3. Other variable and method names that contain multiple characters must **not** start with an uppercase letter.

Further, each "word" within a variable name should start with an uppercase letter. For example, `importantMessage` and `campusMonitor` are both appropriate variable names.

4. Variable names that consist of a single character **may** be uppercase.

   In general, even single-character variable names should be lowercase. However, in some situations, mathematical notation uses uppercase letters. In such situations, uppercase variable names may be used. For example, matrices are often written using uppercase letters. So, an expression like `b = A*x` would be appropriate.

5. Variable and method/function names must be descriptive.

   Variable names like `aaa` are not appropriate. Index variables and counters can, however, have names like `i` and `j`.

3 Declarations

1. Visibility modifiers (other than package) must be explicit.

   Do not rely on default visibilities.

2. All (non-final) attributes should have private or protected visibility.

   Indeed, they must have private or protected visibility unless there is a very good reason for them to have public or package visibility.

3. *All methods that override a method in a superclass or implement a method in an interface must include the `@Override` annotation.*

4. Formal parameters should be declared final.

   Indeed, they must be final unless there is a very good reason for them not to be.

4 Comments

1. Each class must have a descriptive block comment.

   This comment should describe the complete class (rather than the methods in the class).

2. Each method/function must have a descriptive block comment.

   This comment should describe the methods, its parameters, and its return value.

3. Block comments must use the javadoc format.

   javadoc is a program (written in Java) that creates external documentation (in HTML) from block comments. All of the Java documentation was, in fact, created this way. javadoc block comments start with `/**` and end with `*/`.

4. Comments in the body of a class should use `//` rather than `/* ... */`.

   This isn't a requirement but it will make your life easier since you can't nest block comments. There is nothing more annoying then trying to "comment out" a section of code while you are debugging and being unable to do so because it contains block comments.

5. *In the block comment for each class you must include an `@author` element.*

5 White Space

1. *Lines must be short (i.e., less than 100 characters).*
2. *Unary operators must not be separated from their operands by any white space.*

6 Indentation and Blocks

1. *Two spaces must be used for each indentation level.*
2. *Empty blocks are prohibited.*
3. *Do not have multiple clauses on a single line in* `switch` *statements.*

7 Intelligibility

1. `switch` *statements must have a* `default` *clause.*
2. *The* `default` *clause must be after all* `case` *clauses in a* `switch` *statement.*
3. `case` *clauses in* `switch` *statements must not fall-through*
4. *Actual parameters must not be assigned values.*
5. Multiple `return` statements within a single method should be avoided if at all possible.

   Early `return` statements for error-checking are encouraged, but other uses of multiple `return` statements are discouraged. Complicated Boolean return statements must be avoided.
6. *Boolean expressions must be simplified.*

8 Realiability and Robustness

1. Local variables and (most) parameters must not shadow class attributes.

   Parameters in constructors and setters may.
2. Numeric literals (not defined as a constant) should be avoided.

   Such literals are often referred to as "magic numbers".
3. *Classes that define a covariant* `equals()` *method must override* `equals(java.lang.Object)`.
4. *Classes that override* `equals(java.lang.Object` *must override* `hashCode()`.
5. *String literals must not be used with the* `==` *or* `!=` *oeprators.*
6. `java.lang.Exception`, `java.lang.Error`, *and* `java.lang.RuntimeException` *must never be caught.*
7. *The same* `String` *literal must not be used multiple times in the same class.*
8. *An overriding* `clone()` *method must call* `super.clone()`.
9. *An overriding* `finalize()` *method must call* `super.finalize()`.
10. *Assertions must be used to indicate that a statement will not be executed.*