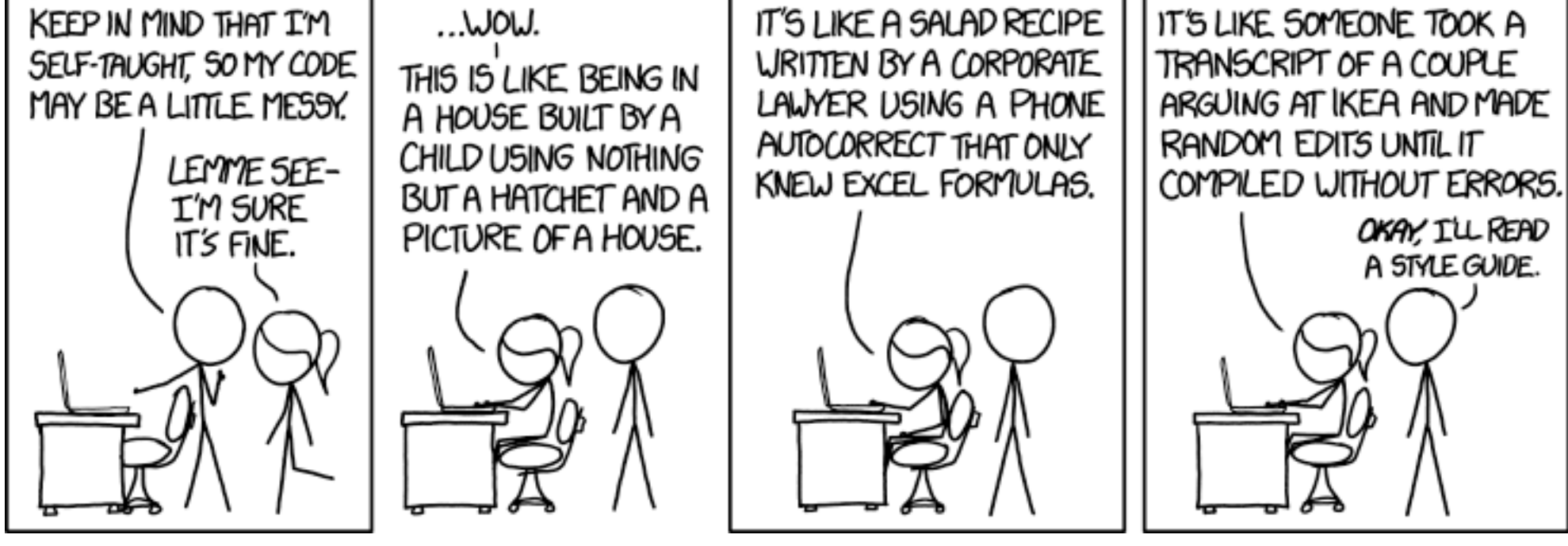


```
/**
 * Object-Oriented Programming
 * Computer Science and Technology
 * Department of Science and Technology
 * BNU-HKBU United International College
 * @version Spring 2022
 */
```

OOP Style Guide



Source: <https://xkcd.com/1513/>

Background

Virtually every organization that does software development has strict guidelines on how to format source code. Here are two examples of widely used Java style guides:

Sun Java Code Conventions (April 1997)

Google Java Style (March 2014)

This course will more or less follow a simplified version of these guidelines. Rather than memorize a long list of rules, you should develop good habits and intuition when it comes to style.

Style Guide

A. Comments

- Every class must contain a Javadoc comment with the following three elements.

```
/**
 * Overall description of the class goes here.
 *
 * @author Your name goes here
 * @version Due date goes here
 */
```

- Every programming assignment must contain the following statement or must cite any sources used (such as a TA). This code must come directly beneath the class comment.

```
/*
 * References and Acknowledgments: I received no outside help with this
 * programming assignment.
 *
 */
```

OR

```
/*
 * References and Acknowledgments: TA Jenny helped me with the foo method.
 *
 */
```

- All methods (including main) must contain an applicable Javadoc comment.

```
/**
 * Overall description of the method goes here.
 *
 * @param paramterName describe each input parameter
 * @param anotherParam use a separate line for each
 * @return describe the value that this method returns
 */
```

- All comments should use normal English spelling and grammar. Phrases are okay.
- In-line Comments must come before the code that they are describing or on the same line.

B. Names

- All names should be descriptive and readable. (`subTotal` rather than `s`, `grade` rather than `grd`)
- Multiple-word names should use capital letters to separate words. (`subTotal`, not `sub_total`)
- Variable and method names should begin with a lowercase letter, and:
 - Variable names should be nouns or noun phrases. (`studentName` OR `subTotal`)
 - Method names should be verbs or verb phrases. (`printLine` OR `addColumn`)
- Class names should begin with a capital letter and use title case. (`HelloWorld`)
- Constant names should be all caps with an underscore separator. (`PI` OR `INTEREST_RATE`)

C. Declarations

- All constants should be named and initialized at the top of the method in which they are used.

```
final double CENTIMETERS_PER_INCH = 2.54;
centimeters = inches * CENTIMETERS_PER_INCH; // NOT inches * 2.54;
```

- All variables should also be declared at the top of the method, directly after any constant declarations.
- It is strongly recommended (in OOP) to separate variable declaration and initialization statements.

```
Scanner input = new Scanner(System.in); // discouraged

Scanner input;
input = new Scanner(System.in);
```

- There should be at most one variable declaration per line. Comment to the right if the name is not self-explanatory.

D. Literals

- Numeric literals should be of the correct type for the context in which they are used.

```
// integer expressions should use integer literals
int count = 2;
double value = 2.0;

// double expressions should use double literals
double x = 2.0;
double y = 5.4;
double average;
average = (x + y) / 2.0; // NOT 2, which is an integer
```

E. Indentation and Whitespace

- Subsections of code should be indented consistently with a tab character.
- Always use tab characters, not space characters, for indentation.
- Statements too long for one line should be indented on subsequent lines.
- There should be a space after cast operators, commas, and `//`'s.
- Use whitespace to separate logical segments of code. There should be a blank line after variable declarations.
- Binary operators should be separated from their operands by a single space.

```
(sum = myGrade + yourGrade;)
```

- One exception is the dot (`.`) operator, which should not have space surrounding it.

```
(System.out.println();)
```

- Unary operators should not be separated by a space.

```
(myGrade++;)
```

F. Structure

- Lines should be kept short (< 80 chars). You should be able to see the full line in your text editor.
- All blocks of code (even if one line) should be surrounded by curly braces.
- Left braces must appear on the same line as the structure header.
- If a method returns a value, it should have a single return statement.
- Break statements should not be used except in the case of a switch.
- You must not have any unused variables or constants or lines of code that do nothing (like `a = a;`)
- You must NOT have any empty if/else blocks:

```
if (condition) {
    // This block is empty. Not OK.
} else {
    System.out.println("Condition not true!");
}
```