Java Style Guide

Formatting

- Lines of code should be no longer than 100 characters.
- Always use curly brackets when optional.
- Every statement should be on its own line. Do not put code onto one line that would be more easily read on multiple lines.
- Use white space to delineate logical sections of code
- Auto-format your code before submitting it.
- Never use named blocks of code (e.g. using break/continue/goto to jump to a named section)

Commenting

- Write comments in complete English sentences.
- Comments should describe the intent of the code. Comments should not reveal implementation details.
- Provide a Javadoc comment at the top of every class and for every public method.
- Provide inline comments to explain non-obvious code.
- Do not overcomment.
- Do not comment obvious lines of code.
- Delete any commented out code.

This is the format for the Javadoc comments:

- **Classes**:

```
/**
 *  Brief description of the class
 *  (Any extra credit)
 * @author Your name
 * @version Creation date
 */
```

- **Methods:**

```
/**
 * Tell what the method does. Do not reveal implementation details.
 * @param paramName1 what the parameter represents
 * @param paramName2 what the parameter represents
 * ...
 * @return what is returned
 * (An @return line is needed only if the method returns anything but void)
 */
```

Variables

- Variable names should be `lowerCamelCase`. Only class names are capitalized.
- Variable names should be relevant and descriptive.
- Any `final` variables should be `UPPER_CASED`.
- Replace all numbers with a constant (i.e. a `static final` variable). The only exceptions are 0, 1, and 2 (if the meaning is clear)
- Replace all string literals (e.g. `"hello"`) with a constant.
- Instance variables (i.e. fields) should always be `private` or `protected`.
- Instance variables (i.e. fields) should be declared at the top of the class.
- Delete unused variables.

Methods

- Method names should be `lowerCamelCase`. Only class names are capitalized.
- Empty methods should be deleted.
- Do not call a method multiple times. Instead, save the return value in a local variable.
- A method should do only one thing. If a method has multiple functionalities, break it up into private methods.