SIENA*college*
Computer Science

Computer Science 225
Advanced Programming
Siena College
Spring 2017

Topic Notes: Readability and Javadoc

You have likely been asked and encouraged to write *readable* code since your very first introduction to programming.

Before we talk in detail about *readability* and how to improve your code's readability, let's look at code that's intentionally unreadable, or *obfuscated*.

Obfuscated code is source code that is very hard to read and understand, in many cases done intentionally.

For example, let's see what this code does:

See Example: Obfuscated

This one was obfuscated by hand to make it unreadable.

Some code is obfuscated automatically before distribution in an attempt to protect proprietary source code or to reduce the size of the code. For example, all identifiers could be replaced with very short randomly-generated names.

Generally, this is not desirable. Usually we want readability.

Readability

Readability is a measure of the comprehensibility or understandability of a written text.

When speaking of code, we can consider *inline readability* and *exterior documentation*. Exterior documentation is anything that contributes to the understanding of the code that is not written directly in the code. This can include user manuals, README files, and Makefiles.

We will focus on inline readability here. Three ways that programmers can enhance their source code's inline readability:

1. comments
2. indentation
3. identifier names

Indentation is generally easy, as most IDEs have tools to facilitate good indentation.

Choosing good identifier names is also usually not hard, but not all programmers make the effort to do so. Good identifiers should follow established naming conventions and should be meaningful to the context in which they are used.

Which brings us back to comments. We all know very well by now that Java supports comments by ignoring any text between a `/*` and `*/` pair, and any text on a line after a `//`.

Starting today, we also want to focus on *Javadoc* comments, which allow a programmer's comments to be used in an automatic documentation generation process.

See: `http://docs.oracle.com/javase/8/docs/technotes/tools/windows/javadoc.html`

The Javadoc tool, which can be run at the command line, but is also available in BlueJ and other IDEs, generates HTML pages based on the Javadoc-format comments in a program.

Javadoc comments are placed before various parts of a Java program to generate the documentation for those parts:

- fields (instance variables, class variables)
- methods
- constructors
- interface or class definitions (but **not** before `import` statements)

A Javadoc comment has two parts: the *main statement* and the *tag section*. The Javadoc comment begins with `/**` and ends with `*/`.

The main statement is text, interpreted as HTML. that completely but concisely describes the element being documented. This means that you can include HTML tags, but it also means you need to be careful for special characters. Most commonly, be sure to use `<` for <, `>` for >, and `&` for &.

The tag section contains 0 or more Javadoc *tags*, each of which provides some specific information about the entity being documented. A tag begins with the name of one of the tag types, prepended with the `@` character. The text that follows is then associated with that tag.

The most common tags used to document classes and interfaces include

- `@author` – Information about the author of the class or interface
- `@version` – A revision date or version number
- `@see` – Adds a "see also" heading which can include a link

and the most common tags for constructors and methods include

- `@param x` – Describes the parameter `x`
- `@return` – Describes the possible return values of a non-`void` method

Writing Javadoc comments consistently will lead to an appropriate amount of comments and has the desirable side-effect of being able to generate the HTML documentation. Use them!