Introduction

This document contains **rules** and **preferences** for you to follow in all of your Java coding assignments for COMP 3760. The **rules** are required; the **preferences** are optional.

Coding rules (required)

You MUST follow all of the rules in this section. Usually 20-25% of every individual coding assignment grade is reserved for compliance with these requirements. *This translates to about 5 to 6% of your overall course grade!*

Rule #1

Put your name in *every* source code file.

Student ID and set# would be nice also. But not having your name on your work is incomprehensible and inexcusable.

If you are using a file that originally came from me (this may happen a time or two), but you have added some code of your own to it, then you must *add your name* as an additional author.

Rule #2

Include a description of the purpose in *every* source code file.

The best place for this is at or near the top of the file.

It's also OK if you put this with the `main()` method (if there is one), although in that case the comment needs to be more than just "main driver for the program" (which is obvious); it should be something *about the program*.

You should think to yourself: "If I found this Java code lying around my old hard drive in 10 years, what could I write to my future self now so that I would understand it then?"

This might be as short as a single sentence or phrase, or it might be more. It just has to be *something*.

Rule #3

Include a description for *every* function/method.

Suppose your friend asks you "Hey, what does this method do?" and you answer them "Oh, it does blah-blah-blah." *THAT* is exactly what you should write as a comment in your code.

This, too, can be as short as a single sentence or phrase, but it has to be *something*.

Exceptions: sometimes if the name of the method really is so good that it says all that needs to be said. But remember: the person reading your code may not be as smart as you are!

"main()" is usually an okay exception to this rule, too.

Another exception is code that you did not write, e.g. if I give you some "starter code". In that case *I'm* the one who is supposed to write a description for those methods!

Rule #4

Use meaningful variable names.

This is a little subjective. There are always reasonable exceptions, e.g. for things like loop indices. But considering the capabilities of modern coding environments there is literally *no effort involved* in having meaningful names (as long as they need to be) on all significant variables, and therefore there is no excuse for doing otherwise.

Rule #5

Never display unidentified/unlabeled output.

Unacceptable: `42`

Tolerable, but just barely: `x is 42`

What would be even better depends on the context. If you can truly guarantee the reader knows what "x" is, the "barely tolerable" option might be good enough. But if x represents some concept that has an actual meaning (which of course it does), then label the output to signify that. For example: `Average number of apples eaten per year: 42`

Rule #6

Write comments on significant blocks of code.

This is even more subjective. Here are a couple of suggestions to help you think the right way.

One simple rule of thumb: the more uninterrupted lines of code you have in a row, the more likely it is that you should have some kind of comment that mentions what it's all about.

Another guiding principle: if you have chunks of code that represent logical sub-parts of the broader code that you are writing—then each chunk should have a comment. "First we have to sort the data" ... "Next we have to count all the odd numbers in the list" ... "Next we have to count all the even numbers" ... and so on.

Rule #7

Adhere strictly to any class-definition requirements in the handout.

This might include class names, public method signatures, etc. I may be testing your code by incorporating it into some of my own. It is *absolutely critical* that everything lines up correctly, or else I might not be able to run your code at all.

Coding preferences (optional)

The items in this section are requests, not rules. You will not be penalized for them. It's only a list of ways you can save me time when I'm marking. If it's easy for you *not* to do them, those few seconds here and there (multiplied by possibly 100+ submissions) can add up.

- Don't do console input. Instead, hard-code whatever input values your program needs so that it will run fully without intervention.
    - If you *do* use console input for something, make sure the user prompt is *really good* so that I know exactly what/how to enter. Your user is *definitely* not as smart as you in this case!

- Don't use `args[]`, either. This is 5x worse than console input!
- Put highly-visible (and well-named or commented) constants near the top of your file for changeable things, such as input file names.
- Use the default package. This is certainly the least important request on the list. It takes only a fraction of a second for me to delete your `package` statement. And there are good reasons for using packages. But just in case you don't care about them, and if you're wondering whether to bother, then I thought I might as well mention it.