Introduction

This document contains **rules** and **requests** for you to follow in all your Java coding assignments for COMP 3760. Following the **rules** is **required**. The **requests** are optional, but kind and helpful to your instructors.

Coding RULES (required)

You **MUST** follow all the rules in this section. Typically 20-25% of an assignment mark is for compliance with these requirements. *This corresponds to about 3 to 4% of your overall course grade!*

Rule #1

Put your name in *every* source code file you wrote or modified.

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

If you are using a file that originally came from me (this may happen a time or two), and if 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 very near the top of the file.

Alternatively, you can put this with the `main()` method (if there is one), but in that case the comment needs to be more substantial than just "main driver for the program" (which is obvious); it should be something *about what the program itself does*.

You should think to yourself: "If I found this Java code lying around somewhere in 10 years, what will I need to know so that I would understand what it is?"

This might be as short as a single sentence, or it might be more. However, it *must 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 *must be something*.

Exceptions: occasionally 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 acceptable 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* am the one who is supposed to write a description for those methods!

Rule #4

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

This might include class names, public method signatures, etc. I often test your code by incorporating it into some of my own. Following specifications is also how coding in the real world works. It is *absolutely critical* that everything lines up correctly, or else I might not be able to run your code at all.

A few other things

Here are a few more stylistic things that I care about, and I might penalize if you violate them repeatedly or flagrantly, but in practice they probably won't be a problem. In past terms they have not been problems. *Do not undertake this as a challenge!*

Never display unidentified/unlabeled output.

Usually N/A because usually your programs won't do any console output at all.

Write comments on significant blocks of code.

Usually N/A because usually your programs won't have blocks of code large enough to make this a serious problem.

Use meaningful and descriptive variable names.

Probably N/A because you already do this very well.

Coding REQUESTS (optional)

The items in this section are requests, not rules. You will not be penalized for breaking them. It's mostly about ways you can save us time during marking. If it's easy/convenient for you to follow them, those few seconds here and there (multiplied by 100+ submissions) can add up.

- PLEASE do not use console/user-typed input unless the assignment specifically requires it.
  - And if you *do* use console input for some reason, make sure the user prompt is *really good* so that the user (that's me) knows exactly what to type. Your user (me!) did not write your program and does not know what to put in.
- Don't ever use `args[]` for anything. As a marker, dealing with this is a terrible time waster.
- DO 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 low importance, but helps me a little because my testing project doesn't use one. If you do decide to use a package, that's good for you! It takes only a fraction of a second for me to ignore or delete your `package` statement, and there are good reasons for you to use packages. But if you don't care about them, and if you're wondering whether I care, the answer is: I don't, so leave it out.