**Coding Standards**

**General Coding Principles**

* Use proper indentions for every block of code.
* Do not rely on beautifying plug-ins available on your compiler as its format is not the same on other compilers.
* Separate commented out code from the block that is actually being used.
* When making a class, group together the attributes/variables, function calls, if/else statements, loop functions, and return statements, separately with a new line/space.

**Variable/Attribute and Function Names**

* Only use single letter variable names for indices.
* Use descriptive, self-explanatory names (e.g. bookTitle, maxCount, countBooks();).
* Always use the full word when naming a variable for easy identification.
* Only capitalize the second word of a variable/function/attribute name, or the first letter of every word.
* Feel free to reuse variable/attribute names for as long as they are in separate classes. Although it is preferred to be more specific when there are instances of Inheritance, Abstraction, etc. present in the code (e.g. member, goldMember, platinumMember).
* Do not use ASCII characters as variable names for easy typing.
* Use arbitrary keywords instead of Proper nouns or overly specific names (e.g. use *file* instead of *grades.txt*).
* Match the names of the labels displayed on the screen with their corresponding variable names (e.g. onscreen = bookCountLabel, oncode = bookCount).

**Documentation**

* Always type in comments to describe the functionality/purpose of every function.
* Do not document the obvious (e.g. i++ /\*increments i\*/).
* Document about how the block of code works, not why it was used.
* Never comment on variable declaration, variables should already be self-explanatory and are not required to be specified when and where it will be used in the block of code itself.
* Do not add unnecessary comments unrelated to the system itself (e.g. /\*kktmd na magcode\*/).

**Testing**

* Focus on testing the major features of the system rather than the trivial functions needed for basic operations.
* To avoid frequent debugging and retesting in the long run, first test if a function works or not before merging it to the system’s functionalities.
* After testing out functions, document any debugging cases used.
* Test out error handling functions only when necessary. Do not intentionally sabotage the regular operations/functionality of the system just to test out certain error handlers.