**T02 Coding Standards**

Version 1.0

Revised February 1, 2021

Tanner Evans (manager)

Thomas Bowidowicz (documents)

Robin Acosta

Marcos Lopez

Jared Bock

Jacob Varela

**CS 460 - Software Engineering**

Abstraction:

* Abstraction should be utilized to separate utilization from implementation. The goal should be to build such that one could redesign the entire implementation without changing the public utilization.
* This sometimes means creating wrapper classes for objects with obnoxious standard declarations and simple utilizations. (I have wrapped a synchronized queue which expected a data type of a pair consisting of an Integer and a Message (another custom object) in a simple MessageQueue class before, and even though I had a class file with three lines, this made my other modules far cleaner, without sacrificing understandability.) Do not be afraid to extend data types into sensibly and more fittingly named classes for your use.

Classes to Ease Integration:

* Each module should be built with a TestDriver class which simulates the range of inputs and checks the outputs. This is meant less to check the full range of data and more to provide an early means to run code and correct missteps.
* Dependencies upon incomplete modules should be mocked up with a dummy class which simulates the still-inaccessible class functions.
* All dummy classes should be commented with a todo comment or some other comment which is designed to not go unnoticed.

Naming:

* Mutable variables should be written in camelCase.
* Finals should written in CAPS.
* Any non-local variables that do not change should be declared final.
* Elements should be named simply and descriptively. Abbreviations and understandable shortenings are acceptable.
* The letters I and O should never be used as variable names.
* Method names should usually be descriptive verbs.
* Classes and interfaces should be descriptive nouns geared towards simplicity.

Indenting:

* Each code block level should be indented [4 spaces without the use of tabs.
* Statement beginnings should be indented consistently within a block.

Bracing:

* The opening brace of a code block should usually follow the statement that it applies to on the same line.
* The closing brace of a code block should be aligned with the statement that it closes.
* Empty or single-statement code blocks may be collapsed to one line.
* Curly braces should be used for all multi-line blocks.

Commenting:

* Every class file should contain a JavaDoc header comment which details:
  + The author(s) of the code.
  + The completion level.
  + Any bugs or incomplete functionality.
  + A list of all public elements
* Every public element with non-trivial meaning should be preceded by a JavaDoc comment which details:
  + Its expected use
  + Supported inputs
  + Intended outputs
* Inline commenting should be included where useful.

Line Discipline:

* Lines should be limited to 120 characters, including comments.
* Long statements should be broken down as necessary, and aligned logically.

Access Principles:

* A principle of least privilege should be maintained. Objects should generally provide methods for modification of their members if and only if such functions are necessary.
* Access modifiers should be as constrained as possible.
* Package-private access is acceptable in small and useful amounts.

Object Modification and Initialization:

* Getters and setters should only exist when necessary.
* Fluent methodologies are acceptable.

Debug Printing:

* Debug printing should be designed to print only if some debug flag is set.
* Debug information can print to standard out, but if persistent data is desired, it should print to a debug file.
* All catch blocks which intentionally circumvent exceptions should print error messages to stdout or an error file when they do so.

Organization:

* All non-method elements of a class should be placed at the top of the class, followed by groups of all public, protected, and private methods.
* Local variables should generally be declared at the beginning of a method. This is clearly infeasible for GUI methods etc.