Style Guide for CS 139 – Fall 2006

Style Guides (or programming standards) provide an organization with a consistent way of formatting all programs such that one programmer knows what to expect from another. It helps to debug programs and to maintain programs written by another. For this class, the guidelines that we use will follow the primary conventions of most java programmers.

NOTE: Failure to follow these guidelines will result in points taken off on your programming assignments.

Format:

1. Each class must be in its own file and that file name must match the class name exactly. For example, the `Sort` class would have a corresponding file Sort.java.
2. Class identifiers use title case. Ex: `Money, Shape, TextEditor`
3. Variable, method, and function identifiers will use mixed case and must start with a lower case letter. Ex: `myGrade, amount`
4. Named constant identifiers must be in all upper case. Ex: PI, PROGRAM_COUNT
5. All names used in a program must be descriptive. Only temporary index variables and counters may use generic names. Ex: `studentName, average, currentTemperature`.
6. Separate words in names with an underscore or a capital letter. Ex: `myGrade` (preferred) or `my_grade`.
7. Methods and functions should be listed in the program in alphabetic order. An exception is `main` (which should be first in any program with main) and constructors. If both main and constructors are present in the same file, constructors should come first, followed by main, followed by all other methods and functions in alphabetic order.
8. All variables must be declared at the start of their class, function, method or block in which they are used. All variables should be grouped by type. Only variables that are used in the program should be declared; remove unused variables before submission.
9. Variables should be described if their use is not obvious from their name as an inline comment.
10. Initialization should not be done at the point of declaration, but should be done immediately before first use and only if required.
11. All code must be appropriately indented. Indenting must be consistent. Braces for statement blocks must line up.
12. All binary operators should be separated from their operands by one blank space. `a + b` not `a+b`
13. Methods returning a value must have only one return statement.

14. The `break` statement must not be used except in conjunction with a `switch` statement.

Heading:

1. All program source files must contain a header, containing the following information in the following order in the top left hand corner of the page or file as a javadoc comment of the following form:
    a. Brief description of the purpose of the class followed by a blank line
    b. @author tag followed by the name of the programmer
    c. @version tag followed by the version and date of this program.
2. All programming assignment files must contain an Acknowledgements and References section, which contains the sources of information that you used to complete the lab. You do not need to include instructor assistance, but must indicate any other help that you received, including TA help. In addition to citing who provided the help, you must also state in which part of the code or other part of the assignment the assistance occurred.
3. Minimal heading for lab assignments should include the header without the References and Acknowledgements.

Example of the header for a programming assignment:

```
/*******************************************************************************
 * Overall description of the class goes here
 *
 *   @author            Nancy Harris
 *   @version V1 - September 16, 2006
 */
//******************************************************************************
//   References and Acknowledgements:  I received no outside help with this
//        programming assignment
//
// ******************************************************************************
```

Documentation:

1. Block comments should use the javadoc format (/** to begin the block and */ to end). Use block comments to describe the purpose of each method. See the program template for an example of javadoc commenting. Note the use of inline comments for the References and Acknowledgements.
2. Within the program, use inline comments. (//)
3. Every class should have a brief description of its purpose. Every method including main should have a brief comment describing its purpose, its inputs and its outputs. Treat these comments like an English paper; use

correct spelling and grammar. Input will use @param tags and have one tag per formal parameter; Output will use @return tags and will describe the result of the method.

4. You should also provide helpful inline comments using the // comment designator. If an inline comment is describing a block of activity, it should be on its own line, separated from the preceding section by one line of white space. Comments that describe a particular line of code or a particular variable may go on the same line as that variable or line of code as long as they fit concisely and do not wrap in submit.

   Example:

   ```
   // Declare variables used in the calculation
   method.
   int counter;   // holds the number of items
   ```

5. You should comment each major task within each method. Each major task should also be separated from other tasks by white space. Consider a task to be any structure (if, while, method) or any code that together does a more major task. (Ex: initialization followed by a calculation and assignment.).