CS-101: Computing and Algorithms I
Programming Style Guide

Programming assignments should be written in a way that is easy to read, write, modify and maintain. Correctness of the program is important, however the style in which it is written is just as important. I will be looking for codes that are "excellent" in terms of design and style. The grade for style and design depends on how the appearance of the output is and on how easy it is to use.

You must check to ensure that the following items are met before you turn in your code.

1. ***Class names:*** All class names should be meaningful. All class names must begin with a capital letter and if two or more words are combined to form a class name, then the first letter of every name should be a capital letter. `Examples: Passenger, CheckingAccount, SavingsAccount etc` Every class should be a separate file in the software. The class containing the main method must be named MainClass.java

2. ***Header for the Class:*** Each class must be preceded by the following class header comments

```
/**************************************************
Class Name:
Author's Name:
Date:
Description of the class:
****************************************/
```

2. ***Variable Names:*** All variable names must be meaningful. Only exceptions are for loop indices and they can be i, j, k etc. Note that every variable name must begin with a small letter and if two or more words are joined together, then all except the first word must begin with capital letter. `Examples: count, sum, grandTotal, lengthOfPassengerName etc.`

3. ***Method names:*** All method names must obey the above rule 2, however in addition, the method names are expected to indicate an action being performed, or a job that is being accomplished. `Examples: initialize, countAll, checkIsleOrWindow, etc. No method should be more than a page of printout.  Similar headers must be generated for each class and interface.`

3. ***Method Header Comments:*** Every method must contain a header as below.

```
/**************************************************
Method Name:
Input to the method:
Output(Return value):
Brief description of the task:
Author: (especially important for group projects)
*************************************************/
```

4. ***Constants:*** User defined constants in Java are obtained by using the keyword "final". Use all capital letters in defining constants. Also use the "_" to combine words in constants. `Examples: PI, MAX_SIZE, MIN_SIZE etc.`

5. ***White Space:*** Separate the code into blocks by using white space. That is, use blank line between blocks of code. Note that following code is seperated into 4 blocks, each block is preceded by a line of comment.
   The code should look clean and should be readable, and it should easy to find the beginning and end of methods, classes, loops etc.

```
// Declaring variables a and b
         int a, b;

// Welcoming the user
        System.out.println("Hello, Welcome to my addition program");

// Assigning Values to a and b
       a = 5;
        b= 9;

// Printing the sum to the user
       System.out.println("The sum is"+ (a+b));
```

6. Indentation: Use proper indentation to separate code into blocks. This means that lines inside loops, branching statements and methods and classes should be properly indented. If the body of the loop, if statement of else statement is a single statement, (not a block) indent the statement three spaces on its own line. Put the left brace, {, starting each new block on the same line as the construct that defines it, such as a class, method, loop, if statement or else clause. The terminating right brace (}) should line up with the construct. Example,

```
while (total < 9) {
          total += 5;
}
System.out.println("The total is " + total);
```

7. In-line Comments: Comments must be accurate. Keep comments updated when the code changes. Every related block of code (5 to 8 lines) must be preceded a line of comment. Do not comment every line. There cannot be pages and pages of code no single line of comment. Do not use java language while writing comments, instead use English sentences.

<u>**Note: Here is way to check if your comments are appropriate: Assume that a person with no knowledge of your assignment and no knowledge of Java reads only the comments of your program. The person should understand "what" task the program does and "how" it accomplishes that task.**</u>

<u>8.</u> Messages, Prompts : Do not condescend. Do not attempt to be humorous. Be informative, but succinct. Define specific input options in prompts, when appropriate. Specify default selections in prompts when appropriate. For every input you are requesting from the user, you must describe the format you want the information in.

9. Output: Label all output clearly. Present the information in a consistent manner. Ensure that output is readable.

Design Guidelines

A. No Magic numbers: There may not any numbers in your program other than 0 and 1. Every other number must be used via a named constant (using final modifier)

B. Structured Programming: Do not use "continue" statement. Only use "break" statement to terminate cases in the "switch" statement". Have only one "return" in each method unless it complicates the code too much.

C. Classes and Methods: Do not have additional methods in a class that contains "main" method. Define the class that contains the "main" as the first class in the file. If only one class is imported from a package, import only that class. If two are more are imported, import all classes.

D. Modifiers: Do not declare class variables with "public" visibility. Always use appropriate modifiers for each situation. If a variable is never modified in a program make it "final". Use "final" instead of literals.

E. Other design tips will be taught in class as we go through the lectures.