CIS 254 Programming Guidelines

Programs will be graded in several areas: **Style, Design**, **Program Correctness**, and **Documentation**

Style Rules:

1) Use **meaningful** variable names: **sum, NOT s; area, NOT a; value, NOT x**. Use one-character variables **ONLY** for array indexes. Variable names begin with lowercase.

2) Use blank lines to separate different parts of the program.

3) Use proper indentation and spaces for readability. Binary operators should have a space before and after. Unary operators should not have a space between the variable and operator.
   total = amount + tax; **index++;**

4) Use indentation within classes, methods, in conditional and repetition statements.

5) **Capitalize** the class name: **class SimpleInterest**. Use **lowercase** for method names: **average()**. If your variable or method name consists of two or more words, use **camelCase**: **gradeCounter, interestRate, setHeight(), convertToCelsius().**

6) Break up long lines. Never allow code to extend into 1 inch margins.

Design Rules:

1) Organize your classes so that separate tasks are performed in **methods**. Do **NOT** code all tasks in **main() or a single method**. (**Once we have covered chapter 6**)

2) Use **meaningful** method names. Meaningful method names reduce the need for comments to explain what is happening: **convertToFahrenheit(), not convert()**

3) Do **NOT** use **break**, **continue** or **return** statements inside of a loop. Your loop **MUST** be designed so that the loop continuation condition will become false at some point.

4) Try to have only **ONE** return statement in a method.

Correctness Rules:

1) Your program **MUST** compile. **NO** points are given for any file or program that does not compile.

2) Test your output for correctness. You must **ALWAYS** test with values that are outside the range of the assignment to verify your code's correctness.

Documentation Rules:

1) Clearly describe the purpose of the program in a comment at the beginning of the program.

2) Comments should describe what the code is for: the purpose of any method or class, the meaning of inputs and outputs. Best comments are JavaDoc comments.

3) Do not comment every line. Comments should not repeat the code.

4) Use /* and */ for multi-line comments. Use // for comments on one line.