	Java Style Guide & Scoring Rubric
	Coding Style
	Documentation / Comments

D1	__ (external) Class documentation is placed immediately above its definition.
	Format:
	`/**`
	` * @author Your First and Last Name`
	` * One or more sentences describing the purpose of the class.`
	` */`
	`public class ClassName`
	`{ ...`

D2	__ (external) Method documentation is placed immediately above its definition. This is optional for constructors, destructors, mutators, and accessors. **Exception**: Method documentation is not required above main.
	Format: (List the description followed by any @param @pre, @throw, and/or @return tags as needed.)
	`/**`
	` * Describe the purpose of the method.`
	` * @param paramName and description of parameter 1`
	` * ...`
	` * @param paramName and description of parameter n`
	` * @pre What must be true for the method to run correctly.`
	` * @throw Name and description of uncaught exception 1 thrown`
	` * ...`
	` * @throw Name and description of uncaught exception n thrown`
	` * @return Description of the return value`
	` */`
	`public void methodName(...)`
	`{ ...`

D3	__ Contain correct spelling, grammar, and punctuation.
D4	__ Accurately describe the code.
D5	__ Are concise. Often use commands beginning with action verbs. (e.g. `// Compute the total cost.`)

D6	__ A short comment describing only one line of code may be placed just to the right of the code it describes (if not violating the 80 character rule); Other comments are placed directly above the code they describe, and the beginning / is aligned with the first character of code on the next line.
	`final double LOWEST_A_GRADE = 89.5; // Scores 89.5 to 100 are A's`
	
	`// Get scores from the user.`
	`System.out.print("Please enter the score for the first player: ");`
	`player1Score = keyboard.nextInt();`
	`System.out.print("Please enter the score for the second player: ");`
	`player2Score = keyboard.nextInt();`

D7	__ Document lines of code where appropriate (don't comment the obvious or restate the code). Assume the reader is computer literate and familiar with programming and the language used, but knows **nothing** about the intent of the code.
	Identifier Naming
N1	__ Identifiers are descriptive of the data stored and/or action performed. (e.g. firstName rather than fn)
N2	__ Names for variables (i.e. local, class, and instance) and methods are lowerCamelCase. (e.g. name, firstName, getAccountBalance())
N3	__ Names for classes and enumerations are UpperCamelCase. (e.g. ThirdShiftEmployee)
N4	__ Names for constants and enumerated values (inside the braces) are UPPER_CASE with underscore separators. (e.g. DAYS_IN_YEAR)
N5	__ Single letter identifiers may only be used for counter variables (e.g. i, j, k).
	Named Constants
C1	__ Named constants are used in place of numeric literals. **Exceptions**:{0,1,2}; output formatting; numeric constants in formulas

Formatting

F1 __ Variables and named constants are declared vertically. **Exceptions**: counter variables may be declared horizontally or occasionally similar variables that make sense when grouped together.

F2 __ Code is aligned and indented 2 – 5 spaces in any block. A single statement within a loop, if, or else clause not containing braces is also indented and is on the line below the condition or else (reason: simplifies debugger tracing).

```
if (temperature <= FREEZING_PT)
  System.out.println("Cold!");
```

F3 __ Braces {} either a) align vertically or b) the left brace starting each new block is on the same line as the construct it defines, and the terminating right brace aligns with the beginning of the construct. **Exception**: Braces on the same line are allowable if there is 1 statement in the block.

Option 1	Option 2

if (condition)
{
  statement(s);
}
	if (condition) {
	  statement(s);
	}

Blocks are easily identified, and the paired beginning and ending braces are easily seen if scrolling is not necessary.	More code may be seen without scrolling.

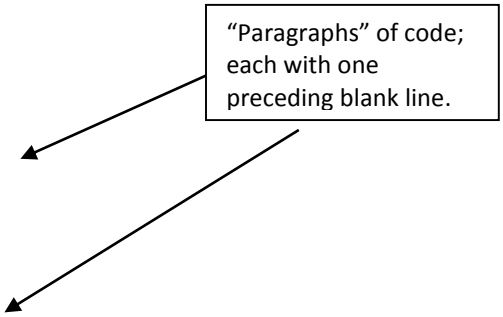
F4 __ Insert one space on each side of relational (<, >, <=, >=, !=, = =), conditional logic (&&, ||), assignment (=), and compound assignment (+=, -=, etc.) operators.

F5 __ Insert single blank lines to break code into "paragraphs" of similar code. Often times, code "paragraphs" are preceded by a comment describing its purpose.

```
int operand1 = 0,
    operand2 = 0;
int sum = 0;

// Get operands from the user.
System.out.print("Please enter an integer: ");
operand1 = keyboard.nextInt();
System.out.print("Please enter a second integer: ");
operand2 = keyboard.nextInt();

// Calculate and display the sum.
sum = operand1 + operand2;
System.out.print("The sum of " + operand1 + " + " + operand2);
System.out.println(" is " + operand3);
```



F6 __ Avoid multiple, consecutive blank lines.

F7 __ Avoid multiple statements per line of code.

F8 __ Follow the 80-25 rule: long statements (> 80 characters) are clearly broken into separate lines. Note that a space is a character. Strive to limit methods to 25 statements or less.

F9 __ Delete anything not used by the program.

Structure

S1 __ Continue statement not used.

S2 __ Break statement only used to terminate cases of a switch statement.

S3 __ Minimize the number of return statements in a method. Strive for one return statement (exit point) in a non-void method, unless doing so makes the code unnecessarily complex.

S4 __ Hide as much class information as possible.

S5 __ Declare all variables and named constants locally to a method or inside blocks within a method. Fields and properties are only used if there is a "has-a" relationship with the class (or if there is no other way to accomplish a task rather than to create a field/property)

Correctness and Efficiency

V1 __ Code works for varied types of input (test extreme and boundary values; test all methods).

V2 __ Code solves the entire problem.

V3 __ Code is efficient (e.g. Repeated code is reduced and unnecessary calculations are not performed.)

V4 __ Any input prompts and output messages are clear and professional (e.g. The user knows exactly what to input and correct spelling, grammar, punctuation, and proper language are used.)