Millersville University Style Guide For Java Programs (short form) (revised February 20, 2020)

All Java programs should conform to the following stylistic rules:

1. Every program should be created as a separate project in Eclipse.
2. Class definitions should be preceded by a Javadoc comment like the following:

```
/**
 * A program that prints out information about a music collection, including the names of tracks, their artists, and the length of each track.
 *
 * @author Dr. Blaise W. Liffick
 */
```

3. Method definitions should be preceded by a Javadoc comment like the following:

```
/**
 * Create a new Statistician that behaves as if it was given all
 *   the numbers from this and another Statistician.
 *
 * @param other An existing Statistician, which may not be null.
 * @return A new Statistician that acts as if it was given all the
 *   numbers from this Statistician and the other Statistician.
 * @precondition The other parameter may not be null.
 * @postcondition Nothing about this Statistician has changed.
 * @throws NullPointerException If other is null.
 */
```

4. Programs should use meaningful identifiers, i.e. they should attempt to convey the meaning of each name. For example, variables to hold a total sum of several numbers might be called `total` or `sum`, rather than `x` or `y`. Avoid using single letter names except when there is no easy way to create a variable name that is descriptive. For example, `s` is OK for the name of a `String` if you are referring to any generic string

and not something specific such as a word or name (in which cases `word` or `name` would be preferable).

However, single letter names for loop control variables are frequently used and would be acceptable in most instances, especially when such variables are being used to index into an array.

5. Note the following special rules for identifiers:

    (a) Class names *must* begin with a capital letter. The class name *must* match the name of the file at the time it is created. (If you make a mistake on this, use Eclipse's *rename* operation to correct the problem.)

    (b) Variable and method names begin with a lower-case letter. However, class constants should be in all upper case.

    (c) In ALL cases of variables, class names, and method names, when an identifier is made up of multiple words, use an initial capital letter for all words after the beginning of the identifier. In the case of class constants, separate words with the underscore character.

    (d) Examples:

        i. a class named `ScrambleWords`;
        ii. a variable named `totalTime`;
        iii. a method named `averageTime`;
        iv. a class constant named `MIN_BALANCE`

6. All programs must use good indentation in order to enhance readability. While Eclipse assists in this by providing indentation automatically, it is easy to add code to a program that prevents Eclipse from automatically including the appropriate indenting. A good habit to get into is to use **CTRL-shift-f** when you complete a program. This causes Eclipse to reformat the indentations of the code. **All code must be properly indented when submitted for grading. This is especially a problem for programs that are submitted remotely.**

7. Visually separate the code of methods by using one blank line.

8. Use comments throughout the program to enhance the readability of the code. This is especially important for any complex code. (If it took you a while to figure out how to write the code, chances are you should explain it.) Use line comments if the note is relatively short (only a few words), and a block comment for anything larger. **Don't** use comments to state the obvious. This is an example of a poor comment:

```
total++;   // increment total
```

9. Use blank lines between blocks of code in order to enhance readability. Think of the code as if you are writing paragraphs in a paper. When your code shifts to a "new thought", put a blank line. This is certainly true when you feel the need to write a lengthy comment about the code ... precede the comment with a blank line.

10. Surround all binary operators (`+`, `-`, `*`, `=`, `<=`, etc.) with one blank on either side, as an aid to readability. Unary operators (`++`, `--`) do not need a blank between the operand and the operator.