CSc. 110 Code Format and Style Guidelines for Java Programs

Apply the following guidelines to all your Java source code. It is best to apply these style guidelines as you code. There are several areas where you are allowed to choose style variations. In any event be consistent. Do not mix alternate style choices within the same program. When making changes, deletions and additions maintain the style and format as you go.

Identification: Every submitted assignment should identify it's owner:

```
/*
* For assignments:  ID and Date
* Always: Program Name
*         Program Description
*/
```

Identifier names

- Names for most variables should be descriptive, unless they are merely loop counters i, j, k, or other housekeeping variables, such as temp for temporary, n for a number or x, y for coordinates.
- In Java names containing multiple words concatenated should use capitalization for the second and successive words rather than underscores
    - (e.g., "goodVariableName" rather than "bad_variable_name").
- The exception is constants (static final fields), which are uppercase and use underscores to separate the words (e.g. "public static final MAX_LENGTH = 10").
- Class identifiers must begin with a capital. (e.g. "BoxCar" or Calculator")

Coding Style

White space: In general, both horizontal and vertical *White space* should be minimized, with the following exceptions:

- Add single blank lines to offset sections of code such as variable declarations that are often set apart.
    - before the first line of a class definition or method definition and after the closing brace for each, and
    - when separating functions as well as major code blocks within functions.
- Insert spaces to assist visual parsing of syntax
    - surrounding assignment, relational and logical operators (and, optionally, around other operators).
- Use parenthesis in expressions not only to ensure proper precedence but also to modularize portions of the expression, especially complex expressions.
    - surrounding assignment, relational and logical operators (and, optionally, around other operators).

Generally, *one statement* per line

- Opening braces, '{', go at the end of the line of code that opens that block (i.e., at the end of the line containing the `if`, `else`, `switch`, `for`, `while`, `class`, etc.). Opening braces on a line by themselves is also acceptable. Each closing brace, '}', goes on a line by itself. It is to be vertically aligned with the start of the line that opens the block.

```
public static void main( ... ) {

    if( ... ) {

        ...

    } else {

        ...

    }

}
```

```
public static void main ( ... )
{

    if ( ... )
    {
        ...
    }
    else
    {
        ...
    }

}
```

- Braces used to create a list of initial values may share the same line to save space. (Used for hard coding values in a array for example.)
- Else statements may be put on the same line as the closing brace for the if statement that they are associated with.

- *Indent* either one tab usually 4 spaces, or two or more spaces, within functions, `if/else`, `for`, `while`, and similar structures. Select either tabs or spaces. Be consistent. An exception is when a line is continued for example a long println should be indented at least an extra tab to visually indicate that it is continued, you may use more if it helps with the code readability.

```
public static void main ( ... )
{
    System.out.println ("Please choose from the following options\n" +
                        "1. Input Values\n" +
                        "2. View Previous Values\n" +
                        "3. Reset Current Session\n" +
                        "4. Save Session\n" +
                        "5. Exit Session");
    // Get input
}
```

- Comments:
  - Avoid commenting beside the code. Once in a while to clarify a difficult expression. Beside the code comments are usually obvious and annoying if over used. ie:

    ```
    int x = 1; // 1 is assigned to x
    ```

  - Block comments should have a blank line before and after. A brace alone on a line can substitute for a blank line.
  - Block comments should have a blank line before and after. races alone on a line can substitute for a blank line
    Every class and every method must have a block comment describing the functionality.

These few guidelines are sufficient for this level. For a detailed examination of style guidelines and the reasons for using them you can refer to Sun's Code Conventions for the Java Programming Language document.

Most importantly be consistent within a program. Indentation, naming conventions and documentation are the most important aspects of style since these help immensely with the correctness of the program. Poor style looses marks. The appearance and readability are appreciated. Early instances of extra lines (code on every second line is a recent violation) or alternatively cramped code (no white space at all) may net a negative comment the first time but not usually lost marks.

Even if submitting electronically, print preview your code to see that it presents within a page 8.5 inches wide, by 11 inches high (Standard page) without unintended line wraps.

If you wish to appeal style infractions show us how you were consistent with published code from your textbook or Sun's official guideline sites.