SE1011 Coding Standard

This coding standard is adopted from the Java Style Guide published by Google and other sources.

1. Source Files - Filename matches Class name

- File name - the source file name consists of the case-sensitive name of the top-level class the file contains.
- File extension - the source file extension is ".java".

2. Source File Structure

- A source file maintains the following order:
    a. File header
    b. Package statement
    c. Import statements
    d. Exactly one top-level class
    e. Attribute
    f. Getters and Setters
    g. Constructors
    h. Other methods
- File headers are a comment block consisting of the following:
    a. Course
    b. Section
    c. Quarter
    d. Academic Year
    e. Assignment or lab name or number
    f. Student name
    g. The date of the last modification of the file.
- An example of a file header is:

```
/**
 * SE1011 – 021
 * Fall 2015
 * Lab 2 - My First Program
 * Name: Brad Dennis
 * Created: 10/13/2015
 */
```

- There should be 2 lines of space between methods other than getters/setters which have 1 line
- There is a line space above comments in code that are not inline

3. Special Characters - Never type a literal tab

- To include a tab in a string, use "\t" rather than a literal tab character.
- Use four spaces instead of a tab character for indentation.

4. Comments

- Class comments – each class is commented using Javadoc style. The comment contains a brief description of the class (not required for a class that is the main or driver).
- Method comments – public methods are commented using Javadoc style, with the exception of getters and setters which are not commented.

```
/**
 * This method prints out "Hello" to the person given and
 * returns the number of letters in the person's name.
 *
 * @param  name   The person to say hello to.
 * @return The number of characters in the person's name.
 */
```

- Inline comments – use comments when necessary. These should be used when the meaning of the code is not obvious from reading the method heading text or to document key design decisions, algorithms, or tasks.
- Self-documenting code:
  - Choose meaningful names for all variables, parameters, classes, and methods. Use complete words were practical.
  - Use named constants instead of numeric literals in your code.
  - Make loop exit/continue conditions clear in the loop's condition statement. For example, instead of writing

```
while(!isDone){
    …
    if(str.length() > 0 && str.charAt(0) == 'A') {
        isDone = true;
    }
}
```

    write

```
while(str.length() == 0 || str.charAt(0)) != 'A') {
    …
}
```

- Flower-boxes or other designs – comments are **not** enclosed in boxes drawn with asterisks or other characters.

5. Formatting

- Variable and Method names – the first letter is lowercase, the first letter of each subsequent word in the name is capitalized, e.g., lowerCamelCase.

- Class names – Each word is capitalized, e.g., UpperCamelCase.
- Constants – the entire word is capitalized; multiple words are separated by underscores, e.g., DEFAULT_LINE_LENGTH.
- Braces – the open brace goes at the end of the line before the start of the code block. The close brace goes on its own line, indented to match the beginning of the line containing the open brace. (Egyptian Braces)
- Code inside braces – indent four spaces for each level of braces.
- Continued lines – indent each line of a continuation an equal number of spaces past the start of the first line.
- An example of a properly formatted code snippet is below:

```
public void someMethod(int argumentOne) {
   if(argumentOne <= 0) {
      doSomething();
   } else {
      doSomethingElse("Perhaps the best way to deal with "
         + argumentOne + " pieces of pizza is to eat them.");
   }
}
```

- Column Limit – no line of code (including comments) contains more than 100 characters.