Coding Standards, ITEC 3150, Spring, 2015

Coding standards are a set of mandatory formatting rules that govern the production of quality, legible code. Coding in a uniform style makes it easier for you to read code from your instructor and classmates. If you already have programming experience, you may have gotten used to different standards. Like any new development shop you work in, please comply with these standards for all code. Your suggestions for improvement are welcome but until the coding standards are updated, code will be graded for compliance with the existing standards.

1. Variable names should be descriptive. Simple variables such as a, t or x should be avoided. Variable names should begin with a lower case letter and the first letter of each word should be capitalized. For example, firstPerson. The use of fp here would not be a descriptive variable name.

2. Class names should begin with an Uppercase letter and the first letter of each word should be capitalized. For example, GoodClassName.

3. Constant names should be all upper case. Words should be separated by and underscore. For example, CONSTANT_NUMBER.

4. Indentation should provide the reader with a context for their location.

5. Comments will be used. Poorly commented code will receive a lower grade. Comments should include descriptive text that will ensure a reader understands the code. All variables should be commented to ensure there is no confusion about the variables purpose.

6. Blank lines should be used after comments, variable declaration blocks, constructors and after each method.

7. Javadoc comments should be used to describe the class in the opening comment block
    a. Example class JavaDoc comment

Example class comment
```
/**Class: ClassName
* @author Your Name
* @version 1.0
* Course : ITEC XXXX Spring 2012
* Written: January 18, 2012
*
*
* This class – now describe what the class does
*
* Purpose: – Describe the purpose of this class
*/
```

- `ClassName should be the name of the Java class.  Not ITEC 3150 or some course number.  If it is a public class, it will be the file name without the .java extension`
- `Author is your name.  Programs submitted without an author will lose points.`
- `Version should start with 1.0.  Minor changes should receive a point upgrade (1.1) and major changes increment the first number (2.0).`
- `Course ITEC 3150, Spring, 2015`
- `Written:  What date did you create this program?  Please don't make it the day it is due`
- `Class Description- what does this class do at a high level`
- `Purpose – why is this class created?  What should a programmer expect this class to do`

b. Example method JavaDoc comment

```
/**  Method: Method Name
  *  Convert calendar date into Julian day.
  *  Note: This algorithm is from Press et al., Numerical Recipes
  *  in C, 2nd ed., Cambridge University Press, 1992
  *  @param day day of the date to be converted
  *  @param month month of the date to be converted
  * @param year year of the date to be converted
  * @return the Julian day number that begins at noon of the
  *  given calendar date.
 **/
```

- `Method –must be a descriptive name of the method.  Helps programmers reading the code understand the purpose of the method.  methodName() is not a good name.  printResults() is a better name.`
- `@param – this is a Javadoc comment describing each argument to a method.  Notice you can have multiple param lines, one for each argument.`
- `@return – describes the results of the method and the type of the return value.`

8. In an *if* statement, use brackets ({}) to delimit the code that you want to execute even if you only have one statement in the if or else block.

9. Use blank space to improve readability. For example when writing an expression with binary operators, use a blank around the operators. Good – (-b + 3) Not good – (-b+3)

10. Main methods should appear at the end of a source file.