**Java Programming Standards**

**Computer Science 220**

1. Introduction

Why do we need programming standards? From the official Java website:

Code conventions are important to programmers for a number of reasons:

* 80% of the lifetime cost of a piece of software goes to maintenance.
* Hardly any software is maintained for its whole life by the original author.
* Code conventions improve the readability of the software, allowing engineers to understand new code more quickly and thoroughly.
* If you ship your source code as a product, you need to make sure it is as well packaged and clean as any other product you create.

Further:

* You gain experience using programming standards representative of those in the operational Air Force and other professional environments.
* Since your work will be graded by someone other than yourself, it is imperative that you do all that you can to ensure your code is understandable. If your instructor cannot understand it, it will not be graded as correct.
* You avoid penalties on each programming exercise for not adhering to these standards!

1. Class Structure
   1. The elements of your Java files should be arranged in the following order:
      1. Class header
      2. Package directive (if included)
      3. Import statements
      4. Class definition  
         Within a class definition, items should be arranged in the following order:
         * 1. Constants
           2. Variables
           3. Constructors
           4. Methods  
              If the class has a public static void main( String[] args ) method,  
              it should be first
2. File/Class Header
   1. Each file in a Java project must contain exactly one top-level class.
      1. Each file will contain a header comment written according to Javadoc guidelines.
      2. Each file header comment will contain at least the following information:

/\*\*

\* Description: *Description of Class Here*

\*

\* @author *Your Name Here*

\*/

* + 1. The main file header comment will also contain your detailed documentation statement:

/\*\*

\* Description: *Description of Class Here*

\*

\* Documentation Statement: *Documentation Statement Here*

\*

\* @author *Your Name Here*

\*/

1. Method Header
   1. Each method header will have a header comment written according to Javadoc guidelines.
   2. Each method header comment will contain at least the following information:

/\*\*

\* *Description of Method Here*

\*

\* @param a *Description of Parameter Here*

\* @param b *Description of Parameter Here*

\* @return *Description of Return Value Here*

\* @throws *Description of Exception Here*

\*/

public int method( int a, double b )  
 {  
 ...   
 }

1. Class, Interface, Constant, Variable, and Method Names
   1. Class and interface names will begin with an upper-case letter and contain upper and multiple words in a class name will be delimited by an upper case letter.
      1. Good Examples: Tractor, TractorTrailer
      2. Bad Examples: tractor, tractorTrailer, Tractor\_Trailer
   2. Constant names will be all upper-case and multiple words in a constant name will be delimited by an underscore.
      1. Good Examples: MAXIMUM, NUM\_TRACTORS
      2. Bad Examples: Maximum, numTractors, NUMTRACTORSINIOWAANDMISSOURI
   3. Variable names will begin with a lower-case letter and multiple words in a variable name will be delimited by an upper-case letter. In general, variable names will be nouns.
      1. Good Examples: count, largestTractor, countOfTractors
      2. Bad Examples: Tractor, TRACTOR, Tractor\_Count
      3. Exceptions to this rule can be made for standard acronyms such as GPA.
   4. Method names will begin with a lower-case letter and multiple words in a variable name will be delimited by an upper-case letter. In general, variable names will be verbs.
      1. Good Examples: loadTractor, isThisATractor, whereIsMyTractor
      2. Bad Examples: load\_tractor, LoadTractor, IWISHIHADATRACTOR
2. General
   1. Indentation and Braces
      1. All code will be properly indented to clearly the logical structure of the code.
      2. Opening braces can be on the same line as the block being opened or on a new line:

for( int i = 0; i < N; i++ ) {  
 System.out.println( i );  
}

or

for( int i = 0; i < N; i++ )  
{  
 System.out.println( i );  
}

* 1. Commenting and Whitespace
     1. In addition to class and method header comments, inline comments should be included in your code where necessary.
        1. Variable declarations should have a truly descriptive name or a comment explaining the purpose of the variable.
           1. Good Examples:  
               private int countTractors, tractorWeight;  
               private int count; // Count of tractors processed so far.  
               private int weight; // Weight of current tractor.
           2. Bad Examples:  
               private int count; // Without comment, what is being counted?  
               private int weight; // Without comment, weight of what?
        2. Particularly complex code segments should have comments. These comments may need to explain both what is being done and why it is being done at this point in the code.
     2. Whitespace (spaces and blank lines) should be used consistently to enhance the readability of your code.