

Java Program Coding Standards

4002-217-9

Programming for Information Technology

Coding Standards:

You are expected to follow the standards listed in this document when producing code for this class. Whether you agree with the use of standards or not, most companies are starting to apply them, so you might as well get used to dealing with them.

Naming Conventions:

1. Names used for classes, methods and variables are expected to be meaningful.
2. Do not use names that differ only in case. (Example: *SQLStatement* vs. *SqlStatement*)
3. We are following the Sun naming conventions used in their Java development:
 - Class names are always capitalized. (Example: public class *Ticket*)
 - Method names always are lowercase for the first word, but all other words are capitalized. (Example: *mortgageEscrowCalculation()* would be a valid method name.)
 - Underscores are not used to separate words in a name. (Exception: *constant_names*)
 - Only the first character of an embedded acronym is capitalized. (Example: *selectSqlStatement*) Exception: when the acronym is the first word in a name, it's all lowercase. (Example: *xmlData*)
 - Constant names should be all uppercase. Constant names are the only place in Java where underscores are used as part of the name. (Example: static final float *MINIMUM_RADIUS* = 6.7f;)

- Names for exceptions should have each word capitalized. The exception name must end with the word "Exception". (Example: InvalidDataException)

Comments:

1. You are expected to comment your code. Some percentage of each project's grade will be based on the documentation provided.
2. Each file should have a header associated with it. This header will contain the following information: (1) Author and course section, (2) Purpose of the file, (3) Caveats: any restrictions on the classes use or known errors. The name of the author and the course section should appear as the first item in the header. (Exception: It is not necessary to put a class header with an inner class. However, there should be a comment about the purpose of the inner class, unless it's blindingly obvious.)
3. Each class should have a header associated with it. This header should describe the purpose of the class. The header should appear ahead of the class definition.
4. If command line arguments are used, this information should appear in a comment that appears ahead of the main method in the class.
5. Sections of code where you are doing something clever, unusual, or where you are making use of some obscure feature need to be commented.
6. Don't comment the obvious. (Example: `j++; // increment j`)

Miscellaneous:

1. Use indentation to indicate the scope of methods and various control structure, like loops, and if statements.
2. I expect that any work submitted for grading has been tested fully. You will lose a significant number of points if your program fails to compile or fails with a stack trace during a run.
3. You are expected to use the SDK1.6 and an editor as your development environment.

4. Labs are required to be individual projects. Group efforts are absolutely forbidden.
5. The book "The Elements of Java Style" by Allan Vermeulen, et al. Cambridge Press 2000, has a very good discussion of coding and documentation standards. It's only 128 pages long and gives different rules along with examples.

Coding Example:

This program shows the way code should be written for this course:

```
// Author:      A.T. Hun      4002-217-01
// Date:       June 24, 2008
// Description: This program accepts one number as a command line argument and prints
//             the Fibonacci sequence that contains the requested number of elements.
//
// Caveats:    Invalid values will be ignored. (i.e. negative numbers or text strings.)
//             The program will not accept a value greater than 45.
//             If more than 1 argument is entered, all arguments after the first are ignored
```

```
// This class generates a Fibonacci sequence with the requested number of terms
class FibSequence {
```

```
    private int maxIterations;
```

```
    // Constructor
```

```
    public FibSequence(String arg){
```

```
        // convert the value from a String to a number if possible
```

```
        int iterations = 0;
```

```
        try {
```

```
            Integer argValue = new Integer(arg);
```

```
            iterations = argValue.intValue();
```

```
        }
```

```
        catch(NumberFormatException nfe) { // not a number - exit
```

```
            System.exit(2);
```

```
        }
```

```
        // see if the number is <= 45 and > 0 - exit otherwise
```

```
        if(iterations > 45 || iterations <= 0) {
```

```
            System.exit(3);
```

```
        }
```

```
        maxIterations = iterations;
```

```
    }
```

```
    // This method is called recursively to calculate
```

```
    // and print out the Fibonacci sequence
```

```
    public void calc(int firstValue, int secondValue, int counter) {
```

```
        if(counter > maxIterations) return; // end of recursion
```

```
        // calculate and print the next term in sequence
```

```
        int nextTerm = firstValue + secondValue;
```

```
        System.out.println(nextTerm);
```

```
        // call fibo to get next term
```

```
        counter++;
```

```
        calc(secondValue, nextTerm, counter);
```

```
    }
```

```
}
```

Header
comments –
expected for
each class

Method
comments –
expected for
each method in
the class

```
// This class tests the Sequence class by accepting a command line argument of the  
// number of terms from the sequence and then calls the calc method of the Sequence class to  
// generate the required list of terms.
```

```
public class SequenceTest {  
  
    // Command line usage: java SequenceTest numberOfElements  
    // Example: java SequenceTest 12  
  
    public static void main(String[] args) {  
  
        // test for at least 1 argument entered  
        if(args.length <= 0) { // no arguments entered - exit  
            System.exit(1);  
        }  
  
        // create the sequence object and initialize  
        FibSequence seq = new FibSequence(args[0]);  
  
        // call the recursive routine to calculate the values and  
        // print them out  
        System.out.println("1\n1"); // print first two terms  
        int count = 3; // first two terms are fixed  
        seq.calc(1,1,count);  
    }  
}
```