Computer Science Code Standards

Code Conventions

1. Class names are capitalized. Example: `Date`, not `date`.
2. All variable names are lower case. If a variable name is made up of more than one word, the subsequent words in the name after the first are capitalized. Example: `dateAsDays`, not `date_as_days`.
3. Names of all constants are upper case. If more than one word is used in the name of a constant, separate words with underscores. Example: `FIRST_YEAR`

Code Style

1. Absolutely no code beyond 80 columns so that it creates word wrap. Break the line if needed. This is BAD:

```
   String  myVariable = "Here is a long string and then" + "let's concatenate
another really long string to it" + "and just to make sure, here is another long
string";
```

   This is GOOD:

```
   String  myVariable = "Here is a long string and then" +
                        "let's concatenate another really long string to it" +
                        "and just to make sure, here is another long string";
```

2. Reuse loop variables in loops that are sequential. This is BAD:

```
    for (int i = 0; i ...) {

    }
    for (int j = 0; j ...) {
    }
```

   Why do you need `j`? These are not nested loops.

   This is GOOD:

```
    for (int i = 0; i ...) {

    }
    for (int i = 0; i ...) {

    }
```

3. Meaningful variable names are **always** a must.
   This is BAD:

```
public void print() {
    //varies the spaces on the left of the diagram based on
    //whether or not the right hand number is 1 or 2 digits
    int x = 1;
    if (this.top > 21 && this.top < 31)
        x = 0;
    else if (this.top < 10)
        System.out.print(" ");
    ...
```

   `x` is being used as some sort of displacement, of 0 or 1, but the name tells me nothing.
4. In your runnable file, the `main()` static method should be the first method.
5. Please use the following bracketing style for blocks of code be it a method body, if statement, loop, or class or interface definition. DO NOT DO this:

```
while (something)
{
statements
}
```

   DO this:

```
while (something) {
statements
}
```

6. Don't explicitly name objects we will NEVER NEED to refer to. DON'T DO THIS:

```
//opening an input stream to read the data
File inFile = new File(fileName);
FileInputStream inFileStream = new FileInputStream(inFile);
DataInputStream inDataStream = new DataInputStream(inFileStream);
```

   We will never ever refer to `inFile` or `inFileStream` so let them remain anonymous and DO THIS:

```
//opening an input stream to read the data
DataInputStream inDataStream =
              new DataInputStream(
                 new FileInputStream(new File(fileName)));
```

7. Declare variables as locally as you can and at the time that you need them. DO NOT DO THIS:

```
// Sort using Selection sort
int i,  minPosition, temp, j;
for(i = 0; i <= length - 2; i++){
    minPosition = i;
    for(j = i + 1; j <= length-1; j++){
        if (fileNums[j] < fileNums[minPosition])
            minPosition = j;
    }
```

```
            temp = fileNums[i];
            fileNums[i] = fileNums[minPosition];
            fileNums[minPosition] = temp;
        }
```

DO THIS:

```
        // Sort using Selection sort
        for(int i = 0; i <= length - 2; i++){
            int minPosition = i;
            for(int j = i + 1; j <= length-1; j++){
                if (fileNums[j] < fileNums[minPosition])
                    minPosition = j;
            }
            int temp = fileNums[i];
            fileNums[i] = fileNums[minPosition];
            fileNums[minPosition] = temp;
        }
```

The variable `i` should be local to the outer for loop as should `minPosition` and `temp`. They never ever will be used outside block of code in the for loop and if we do need something like them again, we can recreate them. The variable `j` should be even more local, it need only exist in the inner for loop.

8. Do not initialize variables unless they need to be initialized because you are accumulating a value in them. DO NOT DO THIS:

```
            double  sum = 0;
            for (int i = 0; i < size; i++)
              sum += data[i];
    ****    double mean = 0;
            mean = sum/count;
```

The variable `sum` needs to be initialized because we are accumulating the sum of the array elements in it. However, `mean` does NOT need to be initialized, and it is misleading to do so. Why does it need to be 0 before assigning it the value of `sum/count`? Answer: it doesn't. DO THIS:

```
            double  sum = 0;
            for (int i = 0; i < size; i++)
              sum += data[i];
            double mean = sum/count;
```

9. Factor your code! This is BAD:

```
           if (age > 50) {
              System.out.println("Your age is: " + age);
              age++;
           } else {
              System.out.println("Your age is: " + age);
              age--;
           }
```

This is GOOD:

```
System.out.println("Your age is: " + age);
if (age > 50) {
   age++;
} else {
   age--;
}
```

Class Structure

When defining your own class, your code should be structured as follows. Note that rarely will your classes have all these parts, but if they do, this is the order to present them in.

```
public class NameOfClass {

    // class constants and variables , if any
    private static ...

    // instance variables , if any
    private ...

    // class constructors
    ...

    // public instance methods
    public ...

    // public class methods
    public static ...

    // private helper instance methods

    // private helper class methods


}
```

Comments

1. You MUST include an id box header on each file that you submit. For example

```
/*
  Name: John Doe
  Course: CIS 203 - Computer Science II
  Assignment: 1
  Due:  9/7/07
*/
```

2. Absolutely NO comments beyond 80 columns so that there is word wrap when it is printed.
3. As a rule, comments should not be placed to the right of code. Put it above. NOT THIS:

```
sum += data[i]; // add value to the sum
```

but THIS:

```
// add value to the sum
sum += data[i];
```

There are two exceptions. When describing the instance variables in a class (also called data members), this is acceptable. For example:

```
public class BankAccount {

    // instance variables
    private String name;     // the name on the account
    private int acctNum:     // the account number
    private double balance:  // the amount of money in the account
```

Another exception is in `if-else` statements to comment the condition on the "else" part. Example:

```
if (value >= 0) {
    ....
}
else { // value < 0
    ...
}
```

To comment a method in a class that your writing, describe parameters, pre- and post- conditions, return value, exceptions thrown in that order. Many of your methods will NOT have all of these attributes. Here are some examples.

```
// Parameter: n - value to take square root of
// Precondition: n > 0
// Returns: square root of n
public double getSquareRoot(double n) {
     return Math.sqrt(n);
}
```

Example:

```
// Parameter: a - an array of doubles
// Postcondition: sum of the array is printed to the console
//                if the array has 0 length, the sum is reported as 0
public void printSum (double [] a ) {
   double sum = 0;
   for (int i = 0; i < a.length)
       sum += a[i];
   System.out.println(sum);
}
```

Example:

```
// Parameter: a - an array of doubles
// Postcondition: sum of the array is printed to the console
//                if the array has 0 length, the sum is reported as 0
// Returns: sum of the array
public double printSum (double [] a ) {
   double sum = 0;
   for (int i = 0; i < a.length)
```

```
            sum += a[i];
        System.out.println(sum);
        return sum;
    }
```

Example:

```
    // Parameter: a - an array of doubles
    // Precondition: a.length > 0
    // Postcondition: sum of the array is printed to the console
    // Throws: IllegalArgumentException
    public void printSum (double [] a ) {
        if (a.length == 0)
            throw new IllegalArgumentException("array length is 0");
        double sum = 0;
        for (int i = 0; i < a.length)
            sum += a[i];
        System.out.println(sum);
    }
```

Example:

```
    // Parameter: a - an array of doubles
    // Precondition: a.length > 0
    // Postcondition: sum of the array is printed to the console
    // Returns: sum of the array
    // Throws: IllegalArgumentException
    public double printSum (double [] a ) {
        if (a.length == 0)
            throw new IllegalArgumentException("array length is 0");
        double sum = 0;
        for (int i = 0; i < a.length)
            sum += a[i];
        System.out.println(sum);
        return sum;
    }
```