Will Pittman's One-Page Guide to Stylisticness

This sheet is intended to provide rough guidelines to help you build good coding and commenting style. Adapted from Michael Schmitz's student style primer.

Header
Always include a header at the top of your assignment. The header should include your first and last name, the date that you turned the assignment in, your section id, your TA's name, and the assignment name and number.

Variable and Method Names.
When naming variables or methods, be descriptive yet succinct. The names need to be understandable.

- Don't call it "x" when you mean "index" or "result."
- Conversely, you do not want a variable called "theNewArrayThatIAmCopyingDataInto."
- Also, avoid using many variables with similar names, such as "temp1" "temp2" "temp3" etc.
- In some situations, such as a for loop, a single-letter variable, such as "i," is appropriate.

Data Fields

- A variable should only be a data field if the object needs to remember it between two or more public methods.
- If it can be a local variable, it should be a local variable
- Mark your data fields private
- Initialize your data fields in your constructors

```
public class Something{
      //Don't initialize your data fields out here
      private int x;
      private File f;

      public Something(){
            //This is the correct place to initialize
            x = 5;
            f = new File("blah.txt");
      }
...
```

Boolean Zen
"Boolean Zen" is the term we use to describe a series of programming issues that demonstrate a lack of understanding of Boolean variables.

```
DON'T:      if (bool == true)
DO:   if (bool)

DON'T:      return bool == false;
DO:   return !bool;

DON'T:      if (bool)
           return true;
      else
           return false;
DO:   return bool;
```

Other Tips

- Keep formatting, comments, etc consistent throughout your program.
- Don't forget to properly label data fields and methods as public or private.
- Make sure to remove any debugging code from your program before you turn it in.
- And lastly, be careful to follow the specification. Straying from it will likely hurt your score. Always check to make sure that any output **exactly** matches the specs.

Commenting

Adapted from Alyssa "The Saint of Code" Harding's commenting guidelines.

When you write comments for classes or methods, your audience is the client who will be using the class, not for the TA who (hopefully) understands the code or for you who wrote it. So imagine a random person you don't know who wants to use your code but doesn't care how it works – that's who you write for. A good way to get the hang of writing comments is to check out the Java API documentation and see how they write. Whenever we use their methods, we're their client, but we don't know how their code works.

Bad example:

```
// This is the method that prints the values in the ArrayList
// using a for loop.  Should throw exception if the list is null
public static void print(ArrayList list) { ...
```

What's wrong with this?

- *Language*. We already know that it's a method, so just say what it does instead of "this method does." Also, don't say the method "should" do something – if you programmed it correctly, it "will" do it.
- *Implementation details*. Maybe later you learn how to use an iterator and decide to change how the method works. But since you've documented that the code uses a for loop, you can't go back and change it. It is therefore better to tell *what* the method does rather than *how* it does it.
- *Output or return value*. The user probably wants to know how the results look - does it print each value on separate lines or print it in reverse order? That makes a big difference.
- *Exceptions*. The user also wants to know how to avoid breaking the method, so it's good to tell them precisely what will cause an exception. In case they do break the method, they also want to know what type of exception it is in order to handle the error in their program.

Good example:

```
// Prints the contents of list as comma-separated values.
// Throws an IllegalArgumentException if list is null.
public static void print(ArrayList list) { ...
```

Particular commenting style (i.e. JavaDoc, pre/post, freehand) doesn't matter, as long as the client understands everything they need to.