1. A loop, is made of four components; a variable, initialization of the variable, a test, and an update or change to the variable. The variable in a loop is what the loop is dependant on, I say this because this determines how long the loop can run. This can be crucial to the functionality of the loop but is really the easiest part of the loop. The test within the loop determines if the loop proceeds through another iteration, also proving to be essential to the functionality of the loop. Most bugs caused by loops are because of the incorrectness of the test in the loop. The last part of a loop, the updating of the loop, determines how long or how many times the loop runs through; this usually affects the test statement in some way.

It is very important to be familiar with these components because loops are constantly used in java programming and a large amount of bugs can originate from loops. Loops are without a doubt essential to java programming and recursion is also a common necessity across a wide variety of programming languages. If you do not feel confident in your ability of loops or feel as if they are not necessary, you must understand that they are crucial to the effectiveness of your code and can make the difference from a good coder and bad coder.

The three types of errors within java are syntax errors, runtime errors, and logic errors. Syntax errors are grammar errors within the programming language that are very common errors. In a loop, they can be seen in the use of initialization, testing, and updating the variable by calling or mistyping the wrong name for a variable or method. Runtime errors are errors that call on the computer to do something that it cannot do. For example, if you were initializing a double to 1 divided by an int, if the int was ever equal to zero, a runtime error would occur. This exact situation could happen in a test within the loop or even with the update of the loop. Lastly, logic errors are errors that don’t necessarily directly cause an error, but rather is a placement of a wrong operator or method call. This can be seen in defining the variable by naming it the wrong type, in initialization by multiply when you should be dividing, if its a numerical datatype of course, in testing by flipping a relational operator, and even in updating the variable, if you were subtracting from a count variable rather than adding.

1. When using an instance variable, you should be using it within an object, for example using it just to take in a Scanner input. The purpose of a parameter is to have a variable passed into the method and then used within the scope of the method that it was passed through. Therefore it should only be used when it is within the method; the further use of this variable can then be transferred using a return statement. Local variables are to be used for “temporary” storage that are within a method. For example, they could be used to change something within a method or a return value.

A common mistake that can be easily made by misusing one type of variable for another, is using local variable outside of its scope. I commonly made this mistake when I was initially learning java by trying to use an initialized data type within a method, in an alternate method. This is a very simple mistake to fix as it can be avoided by making a return statement or just defining this variable within the other method.