Java While Loops

Adapted from Nick Parlante @ Coding Bat.

Java Loops

With loops, you get to leverage the power in the computer. Working with computers, you quickly learn that they lack any sort of insight to solve problems on their own. On the other hand, the computer is happy to execute the code we specify a million times over, which is its own sort of power. Loops are the way to do this. They are like if statements but have the possibility to iterate (loop) more than once.

While Loops

A while loop has a boolean test and a body containing statements, like this:

```java
int counter = 1;
while (counter <= 10)
{
    System.out.println("Current Count: " + counter);
    counter++;
}
```

An if-statement looks at the test one time and then maybe executes the body once. The while-loop extends this idea, executing the body again and again, checking the test each time. The while-loop follows these steps:

1. Check if the test is true or false. If false, the loop "exits" and does not execute the body. If the test is true, continue with step 2.
2. Execute the body statements, starting at the top and proceeding down through them all.
3. Go back to step 1 and check the test again. If it is true, run the body again. Each time the body finishes, "loop around" and check the test again. Each run of the body is called an "iteration" of the loop.

Eventually the test is false, the loop exits, and the program continues with the whatever, if anything, comes after the while statement.

Notice the counter++. This statement prevents the loop from executing indefinitely by incrementing the counter variable; it will eventually reach the endpoint (sentinel), which we defined as 10 in the loop header.
Infinite Loops

The most famous sort of bug you can get with loops is an "infinite loop", where through some mis-arrangement, running the loop body fails to get any closer to making the test false. Typically, this comes down to some sort of logical disconnect between the body and the test. For example, suppose the body fails to make the count bigger by accident:

```java
int count = 0;
while (count < 100) {
    System.out.println("count:" + count);
    count = count * 1; // OOPS, does not change count .... 0 times anything is always 0!!
}
```

Or suppose we forget to put in the count statement entirely, so the loop just spins in place with count stuck at zero:

```java
int count = 0;
while (count < 100) {
    System.out.println("count:" + count);
    // OOPS, forget to put in a line that changes count at all
}
```

Prints when run:
```
count:0
count:0
count:0
count:0
...  
--never stops--
```

Note the difference between < and <=. < is just less than, and <= is less than or equal to. You should remember this from our discussion/review on conditional statements. Please make sure to understand what the difference is and how it can affect the flow of your program.

For more information on while loops, check out this video tutorial: