Java 4: Loops

while(heartOfComputing){}

Types of Loops

There are 2 main types of loops, a variation on one of them, and a complex system known as Recursion. The two main loops are “While Loops” and “For Loops”. “While Loops” work where “While a condition is true, keep doing this”. This is good for when you want something to keep looping until a specific condition is met. “For Loops” work where “For 0 to 10 do this”. It’s hard to explain in words, but the syntax looks like this: for(int i=0;i<10;i++) which means we declare a variable i, and every time this loop occurs, i is incremented by 1, and will stop looping when i equals 10. So to put it all abstractly

while(condition is true)

for(variable; condition with variable is true; increment variable)

Next we will talk about the variation

Loop variations

The only important loop variation is known as “Do While”. It essentially does something before checking the condition. This image will help explain.

Recursion in Brief

We will cover recursion later, but to give you the most surface level overview, it is when a function calls itself until some condition is met. Don’t worry about this now, we can cover it when we talk about functions in depth.

Loop Example

```
public class HelloWorld{
     public static void main(String args[]){
          int x = 0;
          while(x < 10){
               System.out.println(x);
               x++;
          }
          for(int i=0;i<10;i++){
               System.out.println(i);
          }
     }
}
```

Code Analysis

You may notice when you run this program, you get the same output twice. These loops do the same thing, but the implementation and effects are different. For one, the while loop is changing our variable x, meaning that after the loop ends, our variable will be 9, rather than 0. In comparison, in the For Loop, we are only changing i, which once the loop ends is forgotten.

With this knowledge, you can determine when to use each. While Loops are good for when you need to test an existing variable. For Loops are good for when you need to test a value a specific number of times.

The next slide is something critical that you cannot forget

Infinite Loops

It will happen, you will make a program that fails to exit due to an infinite loop. On a console you can press CTRL + C to terminate the program. This happens mostly when working with while loops, but can also happen in any loop. It's important when making loops to always verify that they can end.

Of course, there are some things that you might want to be infinite, until something happens, which then you may want to *break* away from it, but how?

The Break Statement

Yes, we simply break! Calling the word break in a loop will cause it to exit the loop.

Give it a try in the code from before, add “break;” after the println, and you’ll notice it only prints one value before exiting.

Break statements are commonly used when you have a program that is slightly random, and the condition where a loop should end isn’t always obvious. Of course, if you put your break in an If statement that is never satisfied, it will still infinite loop.

Experiment with these concepts, and try using everything you know about loops, variables, and conditions to make something… in fact, here’s a little project.

Project 1: Shapes

You know how to print things to console, so how about printing some shapes with *’s.

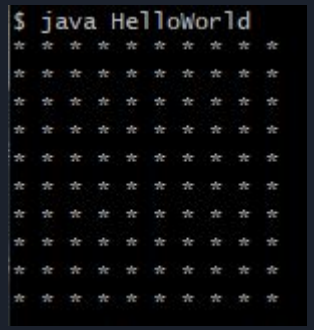
Try making a right angle triangle, a square, and a pyramid in code using loops and ifs.

Remember that you can nest loops (meaning put them within each other) to produce some nice effects.

On the next slide I will show how to draw a square, use this as a base for your triangles

Some Code to Get Started

```
public class HelloWorld{
	public static void main(String args[]){
		for(int i=0;i<10;i++){
			for(int j=0;j<10;j++){
				System.out.print("* ");
			}
			System.out.print("\n");
		}
	}
}
```



Tips

Remember everything about for loops, especially how they are created

for(variable;condition;increment)

Think about how you would start a right angle triangle… first 1 * then on the next line 2 *'s

Remember if you're ever stuck to contact me for tips.

Think about how you are controlling the for loops, since that's what drives your output.