

Thing	What it does
<code>;</code>	Notes the end of a line of code
<code>System.out.println();</code>	Prints out a string (text)
<code>//</code> <code>/**/</code>	Creates a comment that the computer ignores. This is where you tell the user what is going on!
Data Types: <code>int</code> (numeric non decimal) <code>double</code> (numeric decimal) <code>boolean</code> (<i>true</i> or <i>false</i>) <code>String</code> (text)	Usage: <code>int variableNameHere;</code> //makes a numeric variable named //variableNameHere <code>String textVariable;</code> //makes a text-holding variable
Declaring and Using Variables: <code>int number = 0;</code> <code>String word = "lol";</code> <code>int numberPlusTwo = number + 2;</code> <code>number = 5</code>	//makes an integer variable and sets //the value of it to 0 //Makes a string variable and sets it //to the text lol //makes a new integer variable that //is two bigger than the value of //number //sets the value of number to 5
Quick blurb on scope + indenting Scope "is the part of a program where variables exist" You can't have variables with the same name in the same scope Trying to use a variable outside the scope it was declared will not work (since you can have multiple variables by the same name the compiler will not know which to use). Whenever a new block is created, we increase the indent by one so that it is clear to the user when things	/*There are 3 main levels of scope Variables declared at the Global scope are available everywhere Variables declared at the Method scope cannot be accessed outside of the method Variables declared at the Block scope can only be accessed within the block (a block is anything closed by {}). These variables belong to the biggest block they are a part of. */ /*which is easier to understand? if (x){ if (y){ if (z){ if (a){

begin and end.	<pre> thing(); }else{ bing(); } }else if (d){ ming(); } }else{ ping(); } } if (x){ if (y){ if (z){ if (a){ thing(); }else{ bing(); } }else if (d){ ming(); } }else{ ping(); } } </pre>
Using if statements: <pre> boolean yes = true; if (yes == true) System.out.println("Yes!"); if (yes != true) System.out.println("Yes!"); else System.out.println("No!"); //Note these two do the same thing! if (yes != true) { System.out.println("Yes!"); }else { System.out.println("No!"); </pre>	<pre> /*This will print out Yes! because == is the compare operator. It will evaluate to true if what is on the left is equal to the right. since true == true will be true and do the print statement*/ /*This will print No! because != is the not equal symbol so because true != true is false, it will go to the else statement and print No!*/ </pre>

}	
Using do...while loops <pre>do{ System.out.println("f"); }while (true); boolean yes = false; do{ System.out.println("f"); } while(yes); while (yes){ System.out.println("f"); } int loopCounter = 0; while (loopCounter <= 10){ System.out.println(loopCounter++); } while (loopCounter <= 10){ int loopCounter = 0; System.out.println(loopCounter++); }</pre>	<pre>//Will print f forever //will print f , then it will not loop because yes == true is false //will never print f because this checks if (yes) before it enters the loop and therefore never enters it // //Here, this will not run since loopCounter does not have a large enough scope. When in doubt, make a variable at the start of the program (outside the loop).</pre>
Using for loops: <pre>for (int i = 0; i <= 10 ; i++){ System.out.println(i); }</pre>	<pre>// int i = 0 is executed once before the block (notice how i is initialized) // i <= 10 is the condition for executing the block. When this is true, it will loop again // i++ executes after the block. ++ adds 1 to the variable i (it stands to reason that -- will subtract 1 from the variable i) // This will print out the numbers 0 through 10, each on their own line</pre>

```

import java.util.Random;
import java.util.Scanner;
//For now, ignore these, they let us use Random number generation and
//get user input

//This must match the file name, ie RandomNumberGuessing.java is the
//host file
class RandomNumberGuessing {
    //This is the method that gets run at the start
    public static void main(String args[]){
        //Here we create variables
        int guesses = 0; //The amount of guesses that the user made
        int guess;
        //Don't mind this, it is how we get our random number.
        //This will make sense later but for now don't focus on this
        Random rand = new Random();
        int randomNumber = rand.nextInt(100) + 1;
        Scanner keyboard = new Scanner(System.in);
        //Here we are going to start the game
        //We want to keep going until the user gets the number right
        //hence why we are looping
        do{
            //The user guesses so ++ makes the variable guesses go up
            //by one.
            //Alternate form: guesses += 1; guesses = guesses + 1
            guesses++;
            //Ask user for input
            System.out.print("Enter a number between 1-100: ");
            //Get input
            guess = keyboard.nextInt();
            if (guess < randomNumber)
                System.out.println("Higher!");
            else if (guess > randomNumber)
                System.out.println("Lower!");
            //if the guess isn't higher, and it isn't lower, then it
            //must be correct
            else
                break; //This exits the loop
            //The only way to exit is to find the breakpoint because
            //true will never be false
        }while (true);
        //Tell the user their greatness
        System.out.println("Yay! You got it in " +
            guesses + " guesses");
    }
}

```

