Lesson 3

* Arrays
  + What if we wanted to create a program for teachers to track students? We’d have lots of variables like student1, student2, etc…
  + Arrays allow us to group similar variables together and refer to the values by location
  + Location, or rather Indexing, is zero based. The first value, or element in an array is located at position 0.
  + An array with 5 elements will have positions 0 through 4
  + Arrays are denoted with square brackets []
  + Declare an array of a specific type: int[] arrayName = new int[size];
  + Assign and read values just like regular variables, except the identifier is the array with the index in square brackets: arrayName[index] = 25;
  + Iterate through array with for loop – array.length
  + Iterate through array with enhanced for loop (introducing enhanced for loop here)
* Methods
  + What if I have code that I want to use over and over, but not in a loop, I mean code that I want to use at different times? For example, if I write code to send an email to someone, I don’t want to have to rewrite that code every time I want to send an email. I want to modularize it. Write the instructions, give them a name, then use them any time I need them.
  + Method signature and body
    - Access modifier (for now, use public static; we will discuss these another lesson)
    - Return type – methods, just like operators, perform operations and can return a value to use later, or they can return nothing
    - Name – we need a way to invoke our methods
    - Parameters – some methods take input and perform operations on the input received when used.
    - Body – this is where the code to execute goes. If the method returns something, you must have a return statement.
    - REMEMBER: declaring a method does NOTHING! It is equivalent to writing instructions; just because you write instructions to build a house does not mean a house is built when you finish writing them. You must invoke the method!
    - You can put anything inside your methods.
      * Conditionals, loops, etc
    - Parameters are parking spots, arguments are the values passed into the parameter spots when the method is invoked. You will hear them used interchangeably.
  + Call a method by writing its name and parenthesis
    - Put arguments inside the parentheses if method has parameters, otherwise leave them blank
    - You can assign the return (if it returns something) to a variable just like any other value, data types must match!
    - Use good method names! Should read like a recipe or story and say exactly what is happening.
    - Methods should do one thing.
* Objects
  + Primitive data types have just data, no additional properties or actions (methods)
  + String and Array are not primitive data types, they are objects
  + Access Object properties and methods using dot notation
  + Look up String and Array JavaDocs, look up Scanner too!
  + Create a new instance of an object with new keyword and parenthesis, except for String
* Equality (primitive vs objects)
  + Primitive is easy, ==
  + You may have tried this with String using user input and it didn’t work, you have to use .equals() a method on all Objects.
  + Explain heap and how each instance of an object is a different allocation in memory.
  + Draw variable identifiers on one side and new instances on other side and show switching and equality.
  + Code switching and equality