Lesson 2,

* Boolean data are data that can either be true or false.
  + Some examples of Boolean data in our every day life include answers to yes or no questions, such as
    - Is it hot outside?
    - Is the dinner finished?
    - Are we there yet?
  + These answers to these questions are either yes or no, similar to Boolean values of true or false. The answer to any one of these questions, or any question like it, is typically used to determine a further action. For example, if it is hot outside then we won’t wear a snow coat. If dinner is finished, we will wash our hands and get to the table. And if the answer to are we there yet is no, then our children will entertain themselves for a minute and ask again.
  + In order to identify and evaluate Boolean data in our code, we can use Boolean operators to check things such as if a value is equal to another value, whether a value is greater than or less than another value, or if multiple conditions are true together. Any of these comparisons will result in true or false, and that is called Boolean logic.
  + To perform Boolean logic, we use Boolean operators, and when we combine Boolean values and at least one Boolean operator we get a Boolean expression, and that expression will evaluate to another Boolean value, either true or false.
    - == equality operator
    - Greater than, less than
    - && and ||
    - Multiple &&s and ||s
  + Why are Boolean values important? (they help us make decisions)
* Boolean values and expressions are important, but we need more than just the those to enable the programs we write to make decisions. We need a way in our code to look at a Boolean value and do one thing if the value is true, and something else if it is false. In Java we have conditional statements that allow us to do exactly that.
  + The first conditional statement we can us is the IF statement. The If statement allows us to put a Boolean value or expression inside a pair of parentheses following the word IF. Then, we can follow the parentheses by a pair of curly braces. Any code we place in between the opening and closing curly braces will be executed only if the expression in the parentheses evaluates to true. //examples if name is sam say hi
  + Sometimes we have something else we want to happen if the Boolean expression is false instead. Java provides us with an ELSE statement that does just that. We can place an else statement right after the closing curly brace of an IF statement, then, after the word else we put another pair of curly braces. Any code we place in between these braces will be executed only if the Boolean expression in the if statement evaluates to false. That means the first block of code, following the IF will not execute, but the next block of code, following the else, will execute. //example if age is > than 16 you can get a driver’s license.
  + In these examples we only have two paths that could happen, the path for true and another for false. However, sometimes we may have multiple different paths that can happen depending on the value of something. For example, you may go to the store and buy a single gallon of milk if the price of the milk is less than 3 dollars, and you may even buy 2 gallons if the price is less than 2 dollars per gallon, but if the price is greater than 3 dollars, you may buy none altogether. There are three different options in this decision. To accomplish this, we can use an ELSE IF statement. Else if statements can follow an IF statement and they contain another set of parentheses with an additional Boolean expression to evaluate. If the expression is false, then it won’t run, if it is true, then the following code block will run. This may seem identical to an if statement, but there is one very important difference, the Boolean expression in the else if statement only gets evaluated if the Boolean expression in the preceding if statement evaluates to false. This makes using an if, followed by an else if, more efficient than using multiple if statements for the same decision flow, because all the if statements would be evaluated. //example milk
  + If we have a logical decision flow that has many paths, we could use a bunch of else if statements, with a single else statement at the very end that defines the default code to execute if all of the previous Boolean expressions in the if and else if statements evaluate to false. There is also another programming construct we can use to create logical paths with multiple options, in a similar fashion. This construct is called a SWITCH statement, and is used to evaluate a variable and then provide multiple different code blocks that could be executed based on the value of the variable. //example grades
    - Case
    - Break
    - Default
  + Nested if statements and flattening them
* Conditional statements such as if, else if, else, and switch are extremely useful for making decisions and are at the core of why computing is so valuable. However, these statements only execute once and then they are done. Sometimes we need to perform an action over and over again until something happens. For example, if we are baking, and a recipe calls for four cups of flour, then we need to scoop a cup of flour into a bowl until the bowl has four cups of flour in it. When writing a recipe, we wouldn’t write the line “scoop cup of flour into bowl” four times, that would be ridiculous and unnecessary, we would write something like “scoop 4 cups of flour into the bowl”. Programming is the same, we don’t want to repeat ourselves, when we need to repeat any action we can use something called a LOOP.
  + Loops allow us to repeat some code until a condition is met. Like our baking example, we would scoop flour into the bowl until the total amount of flour in the bowl was equal to 4 cups. There are different types of loops in Java, and they can each be used for similar things, but it’s better to know the strengths of each so that we can use them when they are the right tool for the job.
  + For loops
    - Good for when you know how many times you need to iterate.
    - Different pieces of For loop
      * Variable initialization
      * Condition
      * Increment, post-iteration
    - Printing 1 – 10
    - Printing even numbers (I += 2) and (I % 2 == 0)
      * Notice how there are multiple ways to do things.
  + While loops
    - Good when you need to do something over and over until a condition is met (same as for loop, but usually less knowledge of how many iterations are needed).
    - Introduce condition
    - Talk about infinite loops
  + Do While loop
    - Like a while loop, but iterates at least once – checks condition at the end
  + Enhanced For loop
    - Performs an iteration for each item in a collection
      * We will talk about collections later, but an example is a grocery list
        + Find each item, put item in cart, and purchase each item, etc…
        + Bill each customer
        + Wash each dish
        + Any time you use the word ‘each’ in English, that would allude to an enhanced for loop.
* Reading and responding to console input
  + Scanner sc = new Scanner(System.in);
    - Sc.nextLine();
    - Sc.nextInt();
  + System.out.print vs System.out.println
  + Making decisions based on integers ==
  + Making decisions based on Strings .equals()
    - We will talk about == vs .equals later, just know for now that since String is not a primitive data type, == doesn’t work and we have to use .equals()
  + Make a menu driven console application
    - Select from different products and keep track of the sum of product prices to display total at the end.