Objects Notes

* Two things that programs are made of: Data Structures and Algorithms
* Object Oriented Programming
  + Involves balancing data with algorithms
* Some methods and data logically go together i.e.:
  + A list of ingredient amounts and a method to convert a metric recipe to English units
  + A list of student grades and a method to calculate the student GPA
  + A list of students and a method to calculate class rank
* Those methods make sense together as they all have to do with each other
  + ↑ This is where Objects come in!!
* **Objects combine methods and data that BELONG TOGETHER**
  + For different parts of a program different objects have methods to help the whole program function properly
  + Each object has methods that are their “core” methods
* Blueprint of an object is a **class**
  + Outlines each instance
    - Similar to a skeleton
  + **ONE** blueprint can create **MANY** instances of an object
  + Ex: Class called Tomato with methods that return what color the tomato is an its size
    - To create an instance of tomato   
      Tomato **(class)** tomato **(instance name)** = new Tomato(); **(declaring new instance)**
  + The keyword: *new* means that a new instance is being created
* Four parts of an object
  + 1. Data
    - Instance Variables (each instance ^ like before, has its own set)
    - Initialized before the constructor
    - Each instance of the object will have these!
  + 2. Data Getters
    - Called *accessor methods*
    - **Access** information
    - Get data so the program can function properly usually called something self explanatory (getA, getB, getC)
  + 3. Instance Creators
    - Called *constructors*
    - Construct new instances!
    - Necessary for making new instances because they construct them!
  + 4. Data Setters
    - Called *mutators*
    - Can change things (usually called setA, setB, setC)
    - Sets different variables to certain values based on functionality of program
      * Ex: a program on the characteristics of a person so to set the hair to brown if hairColor was a variable a mutator method could be called   
        public void setHairColor{  
         hairColor = “brown”;   
        }
    - Typically return void because they only change the instance variables
* Classes are usually public which means that anyone can create an instance of a class
* Instance variables are declared at he very top of the class
  + Keep in mind: *they don’t have to be given values here but they can be given default values*
  + Usually always private
  + Private keeps each variable within the class that it is initiated in
* Private keyword: belongs to class and keeps things private
* Final keyword: makes variables not changeable and then public them so they can be accessed from other classes as well
* **VARIABLES**
  + **Local**
    - Everything that you have done thus far
    - Created inside method or loop (the usual)
    - No public/private declarations
    - Garbage collected once loop ends
  + **Instance**
    - Belong to each instance (remember we have instance of objects!)
    - They are usually declared as private so that they can only be accessed within the class itself
      * Garbage collected too but with the instance
    - Can be access with “this” even if there s a local variable with the same exact name
* Overloading methods
  + Unique methods with the same name
  + What differentiates these methods are the arguments passed into them
  + Common with classes that have multiple constructors
  + Hair(String color) and Hair(String color, int length)
    - Overloading methods ^
  + Different *method signatures*