**Key Points**

* Classes allow us to group *state* (instance variables) and *behavior* (methods) in one convenient package. This allows us to DRY (Don't Repeat Yourself) up our code so we can easily make new instances of an object by writing something like:  hana = Dog.new("hana", "shiba inu", 11, "woof", ["carrot", "avocado"])

**Instances**

* The initialize method (a common bug is misspelling it!) defines how to create an *instance* of our class (what attributes will instances have, any setup logic).
  + If we are writing a Dog class, we might want to keep track of a name, breed, age, bark sound, and list of favorite foods for each instance of a dog. These will be stored as *instance variables*. They are defined by using an @ symbol (@name = "hana" will mean the name of this instance of a Dog is "hana").
  + Note we do not call initialize directly; instead we use the new class method: Dog.new(arguments)
* *Instance methods* refer to methods that can be called on instances of the class.
  + Dog#favorite\_food? is an instance method; so we could write hana.favorite\_food?("carrot") and we expect to get true as the return value.
    - MyClass#my\_method is how we express that my\_method is an instance method for MyClass.
* *Getters* and *setters* are common instance methods that allow us to access instance variables and update instance variables.
  + Naming convention is name\_of\_instance\_variable for the getter, and name\_of\_instance\_variable= for the setter
  + These are so common that you will learn a quick way to do this in the near future :)

**Classes**

* *Class methods* refer to methods that are called on the class directly, not the instances.
  + Dog::new is an example of a class method; we call it on the class itself. It won't work for instances of a dog (hana.new will raise an error)
    - MyClass::my\_method is how we express that my\_method is a class method of MyClass.
  + To define a class method, use def **self**.my\_method
* Class variables are variables that are defined in the class, that all instances can access. (An instance will only have access to its own instance variables, but it can access any class variables).
  + To define a class variable, use @@
* *Class constants* are constants that are defined in the class; all instances of the class can access it. Note you cannot reassign class constants.
  + To define a class constant, define it in in the class in ALL\_CAPS, but outside of any method (common); or in a class method.

**Monkey Patching**

* Monkey patching allows us to add methods to existing classes, like Integer, String, Array, Hash, etc!
* self refers to the *receiver* of a method, or the object that comes before the dot (.) when a method is called. So when we say [1, 2, 3].average, in our average instance method definition, self will refer to array instance [1, 2, 3] (the receiver of the call to the average method).