**Summary of keywords for class design:**

public:

* On an instance variable, allows that variable to be accessed from inside or outside the class.
* On an instance method, allows that method to be called from inside or outside the class.
* Typical use: instance methods that are designed to be used by programmers who use your class should be public. “Helper” methods that you don’t want programmers to call who are using your class should be private.

private:

* On an instance variable, allows that variable to be accessed from inside or outside the class.
* On an instance method, allows that method to be called from inside or outside the class.
* Typical use: most instance variables should be declared private unless you have a good reason to make them public.

static:

* When a variable in a class is declared static, this states that there is only one copy of this variable *per class*, rather than *per object* (as is typical). In other words, this variable is *shared* between all the objects of that class, rather than each object having their own copy.
* These are called *static variables* or *class variables* rather than instance variables, because there is only one copy of the variable per class, rather than per instance of the class.
* Because there is only one copy of this variable per class, static variables can be accessed with NameOfClass.variable, rather than object.variable. In other words, you do not need to instantiate the class (create a new object of this class) to access the variable.
  + Of course, to access the variable from inside the class itself, you can just use the name of the variable without the dot (like an instance variable).
* This kind of variable is not as common as instance variables. One typical use is for declaring “named constants” in a class: a variable that has a value that doesn’t change while the program is running, for instance Math.PI. Another common use is for tracking something that is common to *all* objects of the class, such as counting the number of instances of the class that have been created.
* When a method in a class is declared static, the code for that method can only access static variables from that class. Instance variables from that class cannot be accessed.
* This is called a *static method* rather than an *instance method*.
* Similarly to static variables, you can access static methods with the syntax NameOfClass.method(), rather than object.method(). Of course, from inside the class, you can just call the method by using method() without the dot, like a regular instance method.
* This kind of method is typically used in a few ways:
  + As a method that does not depend on an instance of the class it lives in. For instance, most of the methods in Math do not require a Math object to be created, so they are just called as Math.sqrt(), Math.abs(), etc. Note you never say Math m = new Math(); to use these methods.
  + As part of a “driver” or “runner” or “tester” program. A typical Java program will often have one class known as the “driver” or “runner” or “tester” class where the public static void main method lives. The class where main is declared often will never be instantiated itself, and so often contains other static methods to break the program up into manageable chunks (similar to how you learned to break up a program into functions in Python).
  + The driver class is then in charge of instantiating any other objects it wants/needs, and then calling appropriate methods on those objects.
  + The driver class itself often doesn’t have any instance variables or instance methods, only static methods and probably local variables that only live inside each static method.