Mistakes/Reminders:

arrays use arr.length field, strings use str.length() method

Properties vs Fields (At least in C#) Properties Expose Fields; by having two variables (one property and one field) we don’t have to expose the inner workings of the class by showing the field representation of data. Also, we can change up the field name and data types or even its existence as long as the property is fine, and programs that reference the property will be fine despite the changes in fields.  **I guess you could do that with general getters and setters anyway.**   
  
Properties have a lot of other functionality in C# too, like overriding or reflection

can also make get public but set private.

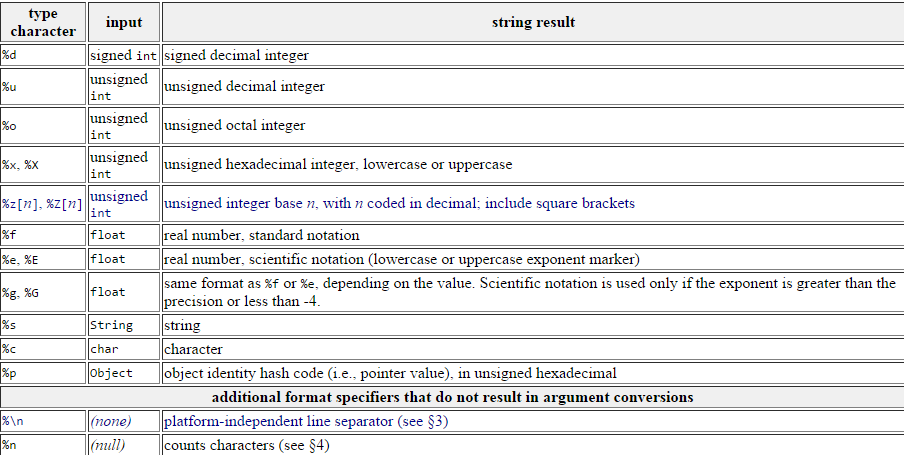
map1.equals(map2) works as long as the keys and values have valid equals methods. Does not work with arrays.

Maps use containsKey and containsValue methods, not just contains

HashMap may have parameters Integer and Character, but what comes out and gets put in can act just like the primitives. JVM can handle it.

Conditional Operator Exists in Java

you can use Conditional operator without having parentheses, but parentheses honestly makes it look better.



To turn char array back to string, do String x = new String(array);

HashMap !!!! (**containsKey, containsValue,** size, put, get)

**Note: memorize String and HashMap methods**

To remove spaces from a string, use replaceAll method.

Don’t forget about static variables: If you are going to make global variables, they need to be **static**

**Don’t forget to make your methods static**

referring to different classes in different folders is hard in java. It requires packages to mirror the folder structure and the **%CLASSPATH%** system variable to be configured.

Or you could just use Eclipse.

Use **String.format** if you want formatted Strings

HashSet = add, remove, contains, size

HashMap = put, remove, containsKey, containsValue, size

If you put variable initialization inside a loop or if statement and declaration outside and usage outside, then you have to initialize it as **something, even null** in order for your class to compile.

don’t make dumb = vs == mistakes

**Instance variables don’t actually have to be initialized for object to exist…** Example: my linkedList implementation with uninitialized integer.

**It simply tells the compiler to assign it its default value.**

Don’t forget about passing by reference and passing by value.

Java’s hashTables check key equivalence by the **equals method** associated with that object type. Hence. It checks for duplicates by **value, not reference.**