How to draw a boxes and pointers diagrams.

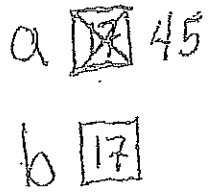
Look at the code. For every variable in the code, draw a little box next to it. If the variable is a primitive type, that box is going to contain a value. If the variable is a non-primitive type, that box is going to contain an arrow.

Then follow the code line by line. What you diagram depends on what the code says:

An assignment of a primitive type

If you see an assignment of a primitive type, that *copies* the value of the primitive type. So the variable being assigned gets whatever value is in the existing primitive type.

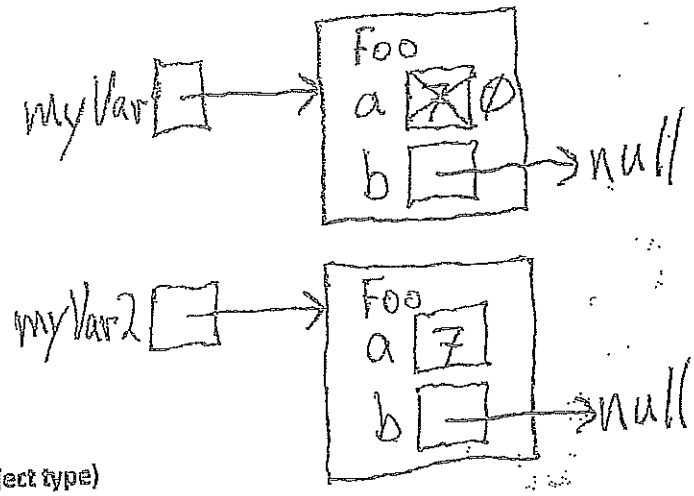
```
int a = 17;
int b = a;
a = 45;
```



A *new* for a class

A *new* for a class creates a new instance of a class. You should make a new rectangle for that class, label it with that class's name, and fill in all the fields for the class (according to the constructor of the class). Note that fields follow all the same rules as normal variables. Make the variable being assigned point to that newly drawn rectangle. *Note that without a "new", no new instances of a class (rectangles) can be created.*

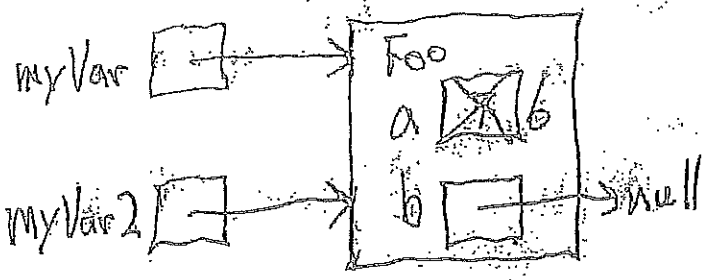
```
class Foo {
   public int a;
   public OtherClass b;
   public Foo() {
     a = 7;
     b = null;
   }
}
Foo myVar = new Foo();
Foo myVar2 = new Foo();
myVar.a = 0;
```



An assignment of a non-primitive type (object type)

If you see an assignment of a non-primitive type, that copies the reference (i.e., that makes the variables point to the same object). So the arrow of the assigned object points to whatever the original object pointed to.

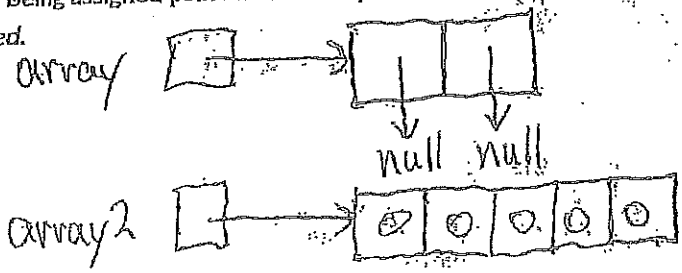
```
Foo myVar = new Foo();
Foo myVar2 = myVar;
myVar.a = 6;
```



A *new* for an array

An array is an object that contains a bunch of other objects. You should draw a new grid which is the values of an array, then make the variable being assigned point to that array. *Note that without a "new", no new arrays (grids) can be created.*

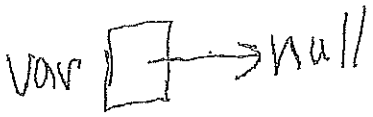
```
//note that arrays default
//to null and 0
Foo[] array = new Foo[2];
int[] array2 = new int[5];
```



Null

I draw null like this. It's not technically correct, but it'll help remind you that object variables should always be arrows.

```
Foo var = null;
```



Strings

Strings are objects but because they're implemented by Java we don't have visibility to know what instance variables are in there. But otherwise they act like other objects. I draw them like this:

```
String a = "hello"
String b = a;
```