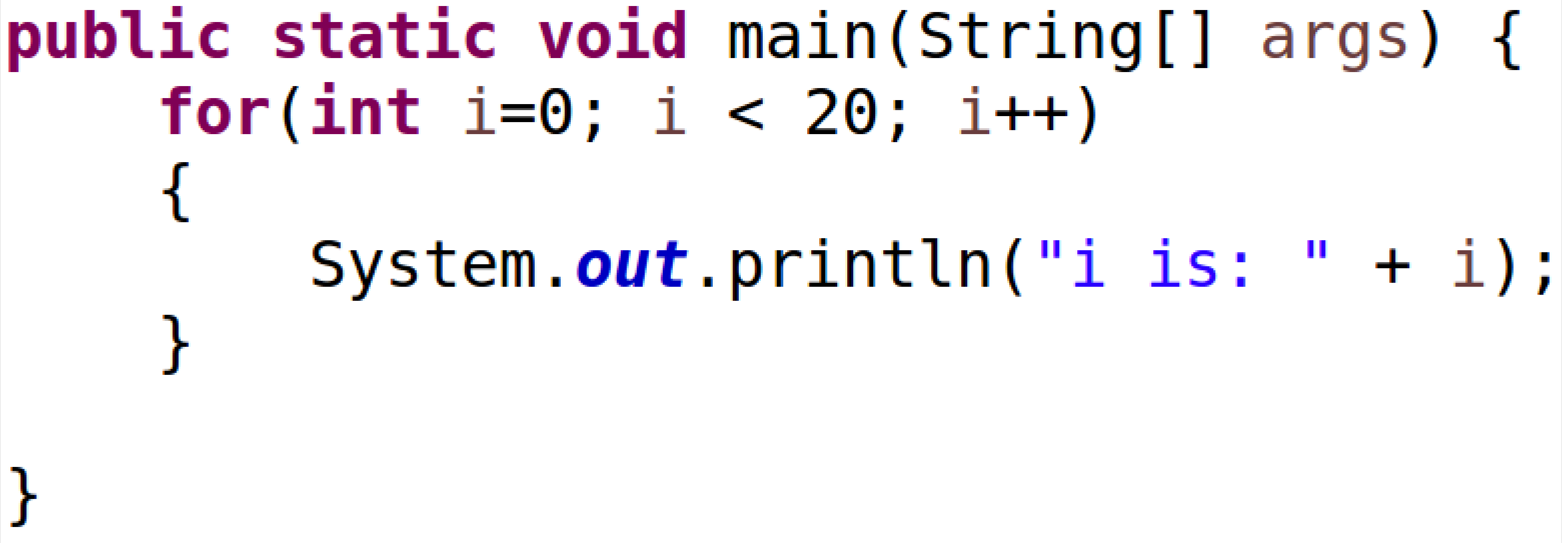
Other languages (Java)

You're a Programmer?!
Yea, I know Java Pretty well...
Wow! Can you make me a game?!
One does not simply
MAKE a game...

Highlights

- Java



```
public static void main(String[] args) {
    for(int i=0; i < 20; i++)
    {
        System.out.println("i is: " + i);
    }

}
```

Similarities

Java is similar C++ and should not be too hard to go between these languages

Java (and most programming languages) have the basic tools:

1. if/else
2. arrays
3. functions
4. classes (if object oriented)

Differences

Easier:

- Automatic memory management
- No pointers
- No pass-by-reference (the & in functions)
- No operator overloading
- Automatically initializes variables

Harder:

- Stricter types
- Stricter structure of programs

Differences

Java has automatic memory management,
so there is no delete

However, “new” still exists and you use it
the same place as C++: when you want boxes

Java also does not technically compile,
instead it run code on a virtual machine
(this makes it more portable, but also slower)

Differences

The way Java and C++ represent data is slightly different:

In C++:

- Pointers & Arrays = memory address
- Types = boxes

In Java:

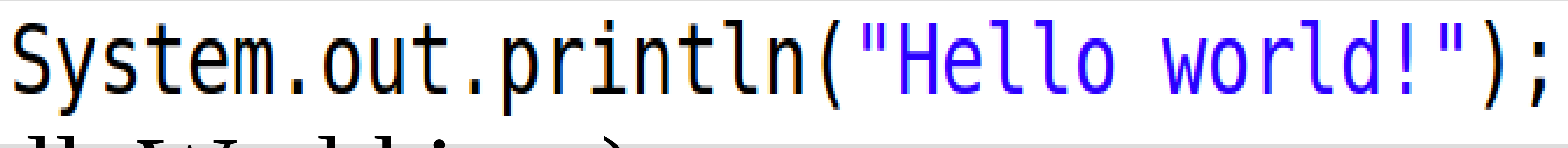
- Classes & arrays = memory address
- int/double/char/etc. = boxes

Hello world!

Let's start with the basics, printing to terminal

This might seem weird, but in Java everything must be in a class (EVERYTHING)

So to do simple printing, we must first make a class, put main inside it (in a special way), then we can finally print:



```
System.out.println("Hello world!");
```

(see: helloWorld.java)

Types

Types are much more strict in Java, and you cannot convert between them as easily

It still allows int -> double conversion, but you cannot go double -> int without type casting

Conversions between char and int are still fine (though less useful in Java)

(see: strictTypes.java)

Static

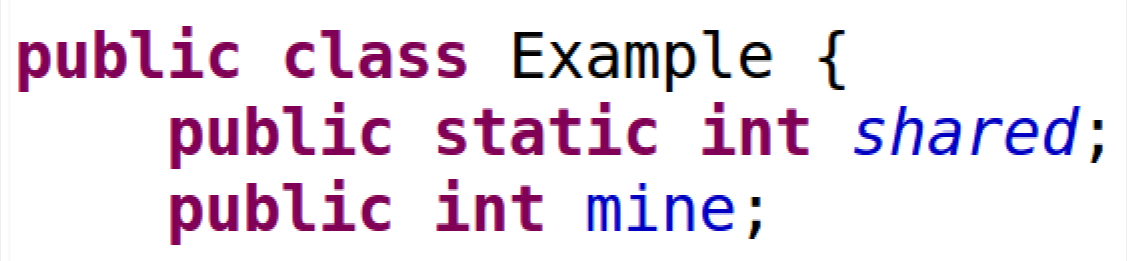
Other than forcing things to live inside a class, Java works in a very similar manner to C++

Coming from C++, the concept of “static” might seem a bit odd

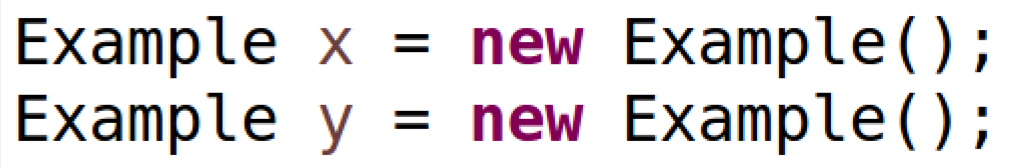
This is a property of the class, shared by all objects/instances of the class
(thus there is only “one” of these shared by everyone of this type)

Static

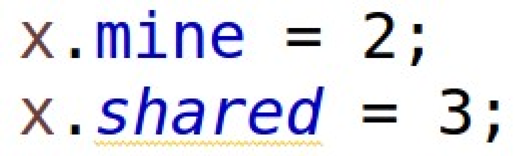
Consider this:



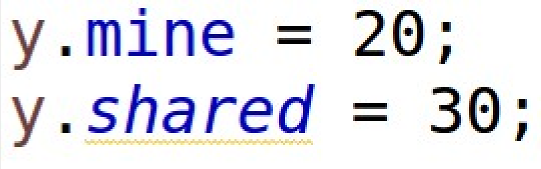
```java
public class Example {
    public static int shared;
    public int mine;
```



```java
Example x = new Example();
Example y = new Example();
```



```java
x.mine = 2;
x.shared = 3;
```



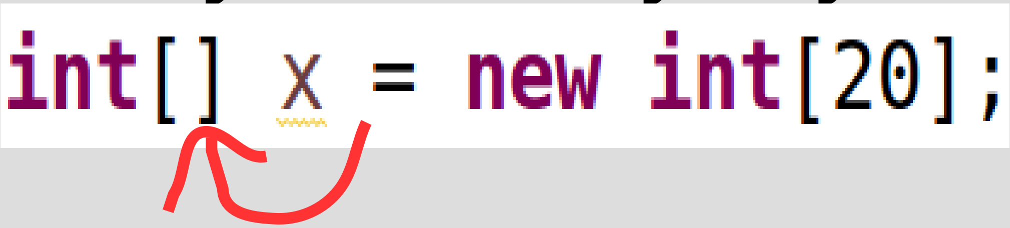
```java
y.mine = 20;
y.shared = 30;
```

(see: Example.java)

Standard tools

Other than this, Java has the same normal tools C++ does

Array notation is slightly different, where the array is denoted on the type and not name: (and they are always dynamic)



```
int[] x = new int[20];
```

(see: Tools.java)

Classes & inheritance

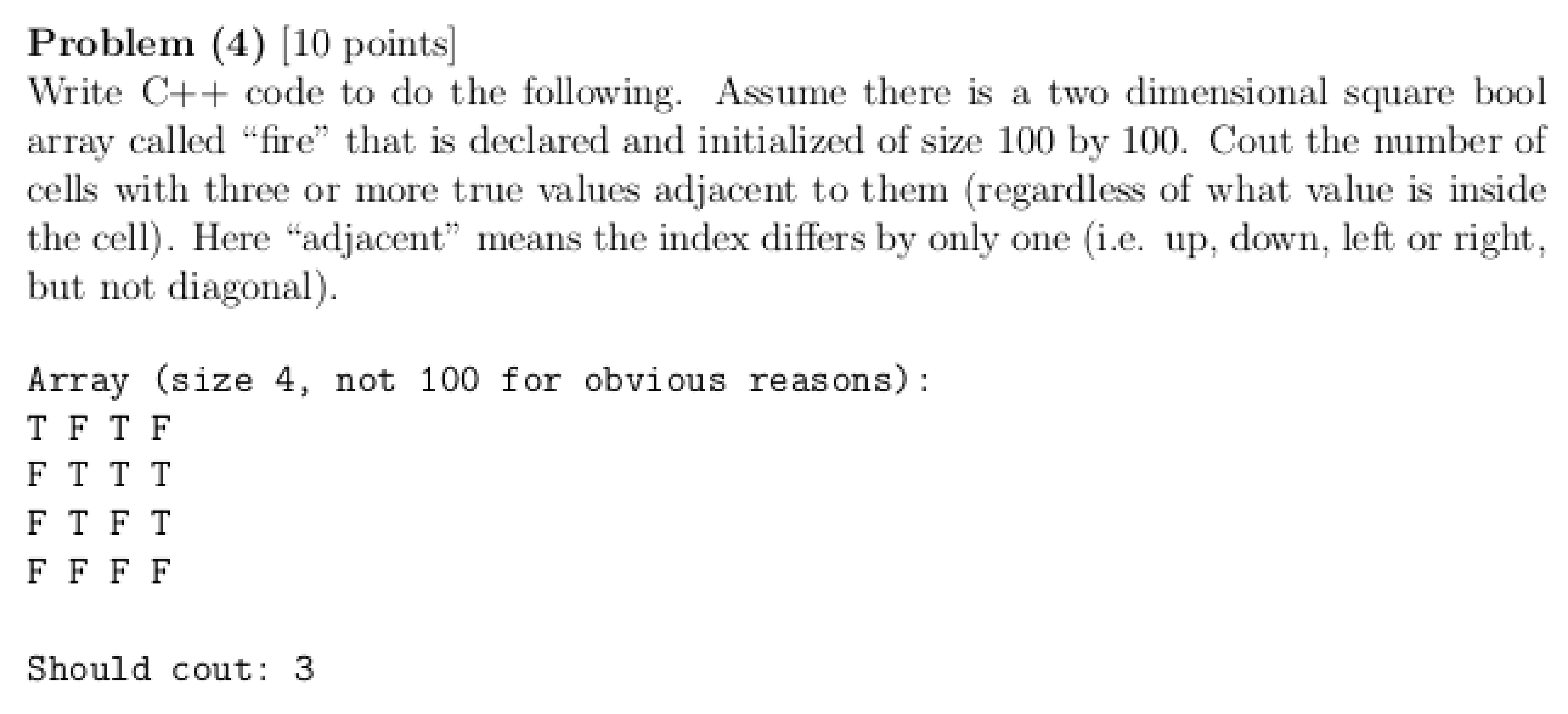
Inheritance works pretty much the same way as we talked about it in C++ (though there are subtle differences of more advanced stuff)

Java by default has every function virtual, along with always being a memory address makes inheritance use a lot easier

(see: ClassesAndInheritance.java)

Mini-problem

For the rest of the time, I will solve this midterm 2 sample problem in Java:



Problem (4) [10 points]
Write C++ code to do the following. Assume there is a two dimensional square bool array called “fire” that is declared and initialized of size 100 by 100. Cout the number of cells with three or more true values adjacent to them (regardless of what value is inside the cell). Here “adjacent” means the index differs by only one (i.e. up, down, left or right, but not diagonal).

```
Array (size 4, not 100 for obvious reasons):
T F T F
F T T T
F T F T
F F F F

Should cout: 3
```

(see: BoolFind.java)