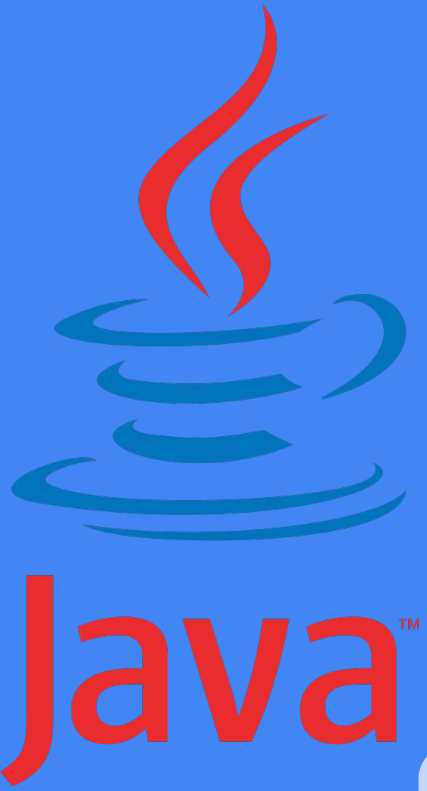
Introduction to Java Programming

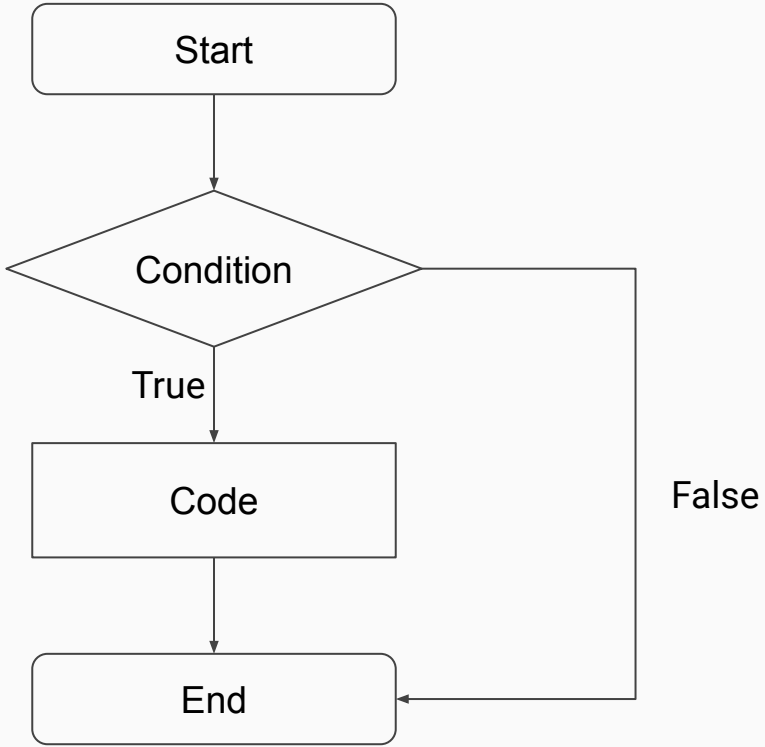
Part 4: Arrays and Object Oriented Programming



REVIEW FROM LAST TIME

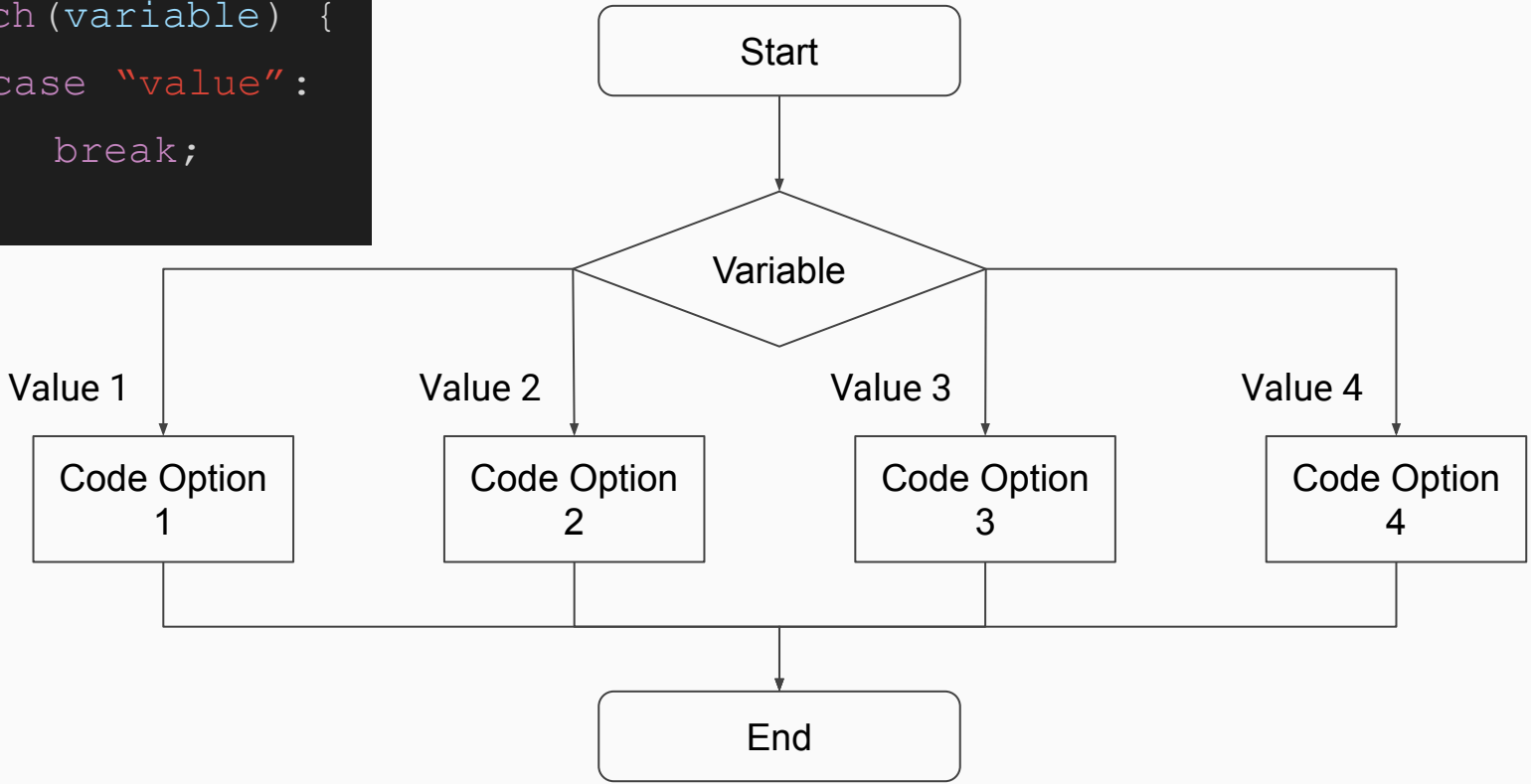
If Statement Logic

```
if(condition) {
    //code
}
```



Switch Statement Logic

```
switch(variable) {
    case "value":
        break;
}
```

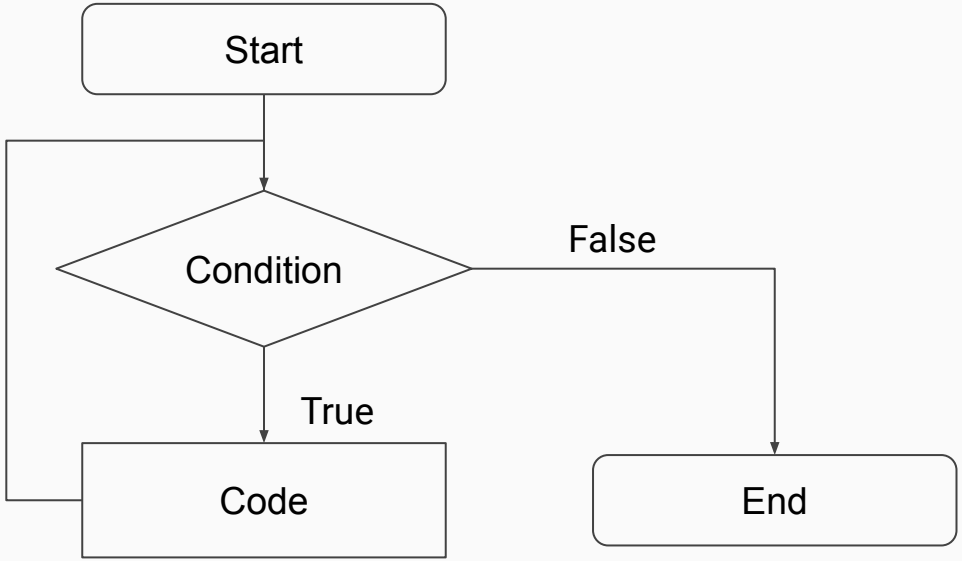


Case Switches vs. If/Else

- Both can be used to change the behavior of a program based on certain conditions
- However, case switches only work with ints, bytes, chars, shorts, Strings, and enums (we will talk about enums later). They only work with specific values - not ranges.
- Case switches are only useful when you have a range of behaviors corresponding to specific values of a variable
- On the other hand, if statements are much more broadly useful. They execute based on whether their condition evaluates to true or false, and therefore can be used to change program behavior based on a range of values
- Remember, each is only a tool that can get you to an end goal. Every task can be done in many ways, and it is in your interest to have access to as many tools as possible.

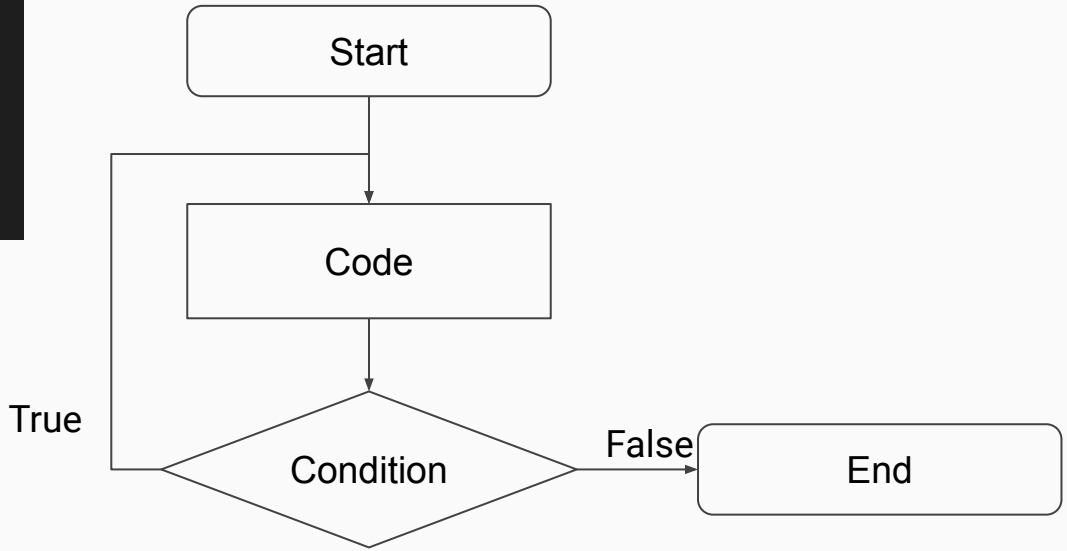
While Loop Syntax and Logic

```
while(condition) {
    //code
}
```



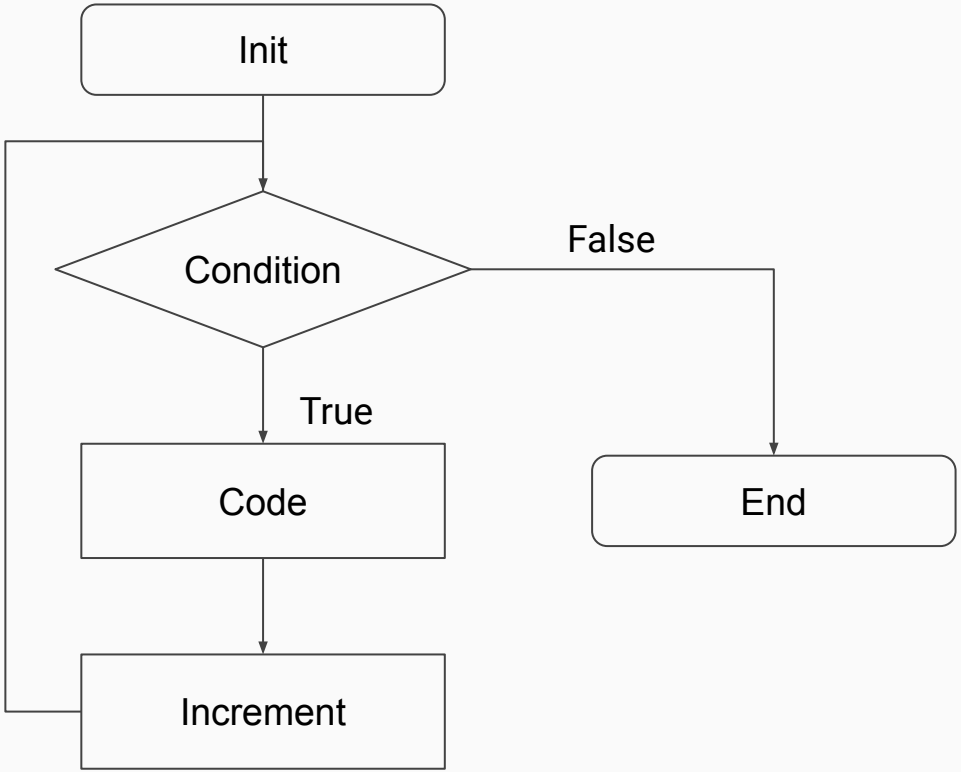
Do … While Loop Syntax and Logic

```
do {
    //code
} while(condition);
```



For Loop Syntax and Logic

```
for(init; condition; increment)
{
      //Code
}
```



When To Use Each?

While Loops have the broadest application case, as they simply loop a section of code until told to stop. This doesn't mean that they are always the right choice, however, but any situation that requires repeating code can be reasonably handled with a while loop. Best used when the number of repeats is uncertain.

Do … While Loops are basically restructured while loops with one key difference: because their code execution comes before the conditional, they will always run at least once. This can be useful for certain applications and is handy to keep in mind, but these are the loops I find myself using the least.

For Loops are a good choice when you want to repeat a section of code based on a numerical iterative variable. If you want a piece of code to repeat a certain number of times, this is your best bet. Basically, if numeric control is important, for loops can accomplish it. The structure of a for loop makes the control of variables easy.

Break and Continue Statements

We have seen break statements in both loop contexts and case switch contexts. However, what do they actually do?

A break statement will exit from the current while/do/for/switch block and continue execution of the code from the end of that block. A continue statement, however, will simply jump to the next execution of a while/do/for loop, bypassing the code remaining in the execution of that loop.

What if you want to stop the execution of a loop from inside a nested loop? Java has a solution to this. You can label a loop using a label and then a colon like `label: for ...` and then call a break or continue statement such as `break label;`

In general, one should usually program a loop so it exits based on its condition and doesn't exhibit unexpected behavior requiring a break or continue. However, all tools have times when it makes sense to use them.

Reminder: Java Operators and Math Functions

```
Math.pow(a,b);
Math.min(a,b);
Math.max(a,b);
Math.abs(a);
Math.PI;
Math.sin(a);
Math.floor(d);
Math.ceil(d);
```

```
boolean a,b;
        a&&b;
        a||b;
        !a;
int x,y;
        x>y;
        x>=y;
        x!=y;
```

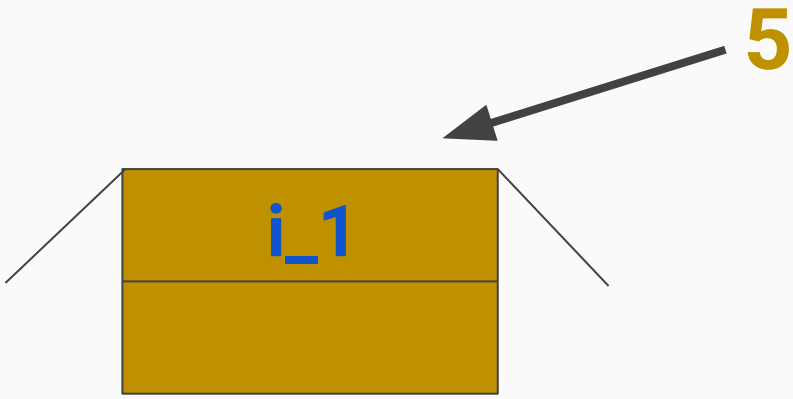
```
int x = 5, y = 3;

        x+y;
        x-y;
        x*y;
        x/y;
        x%y;
        x++;
        x--;
        x+=y;
        +x;
        -x;
```

END OF REVIEW

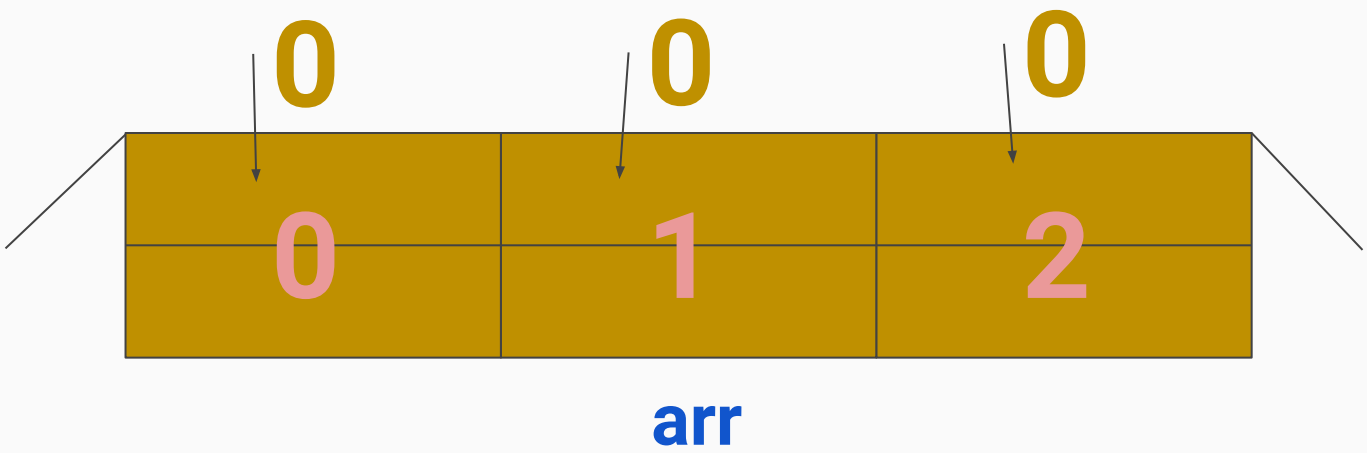
Variable Syntax

```
int i_1 = 5;
```



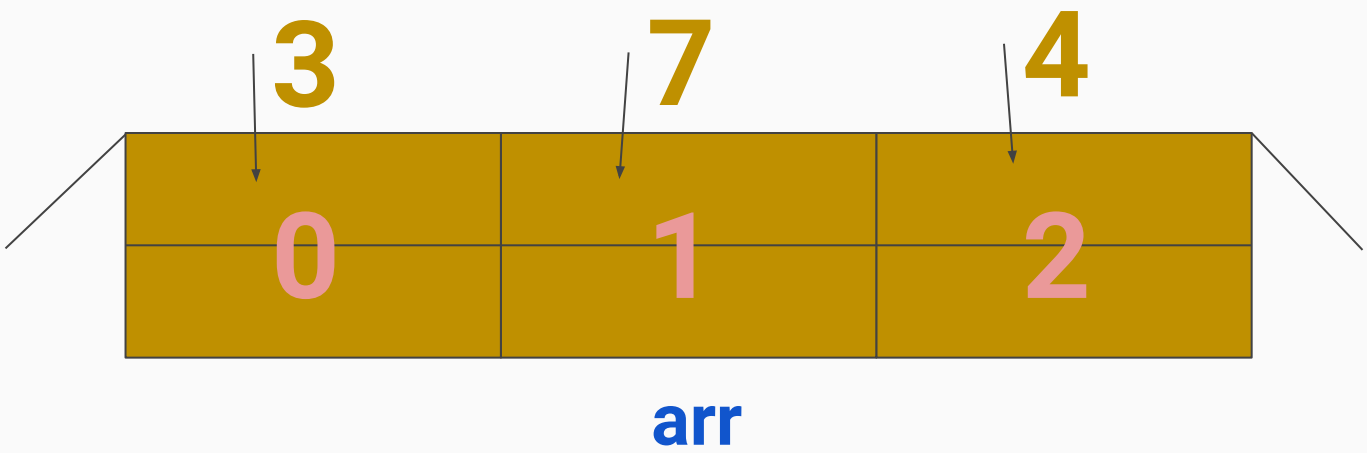
Array Syntax

```
int[] arr = new int[3];
```



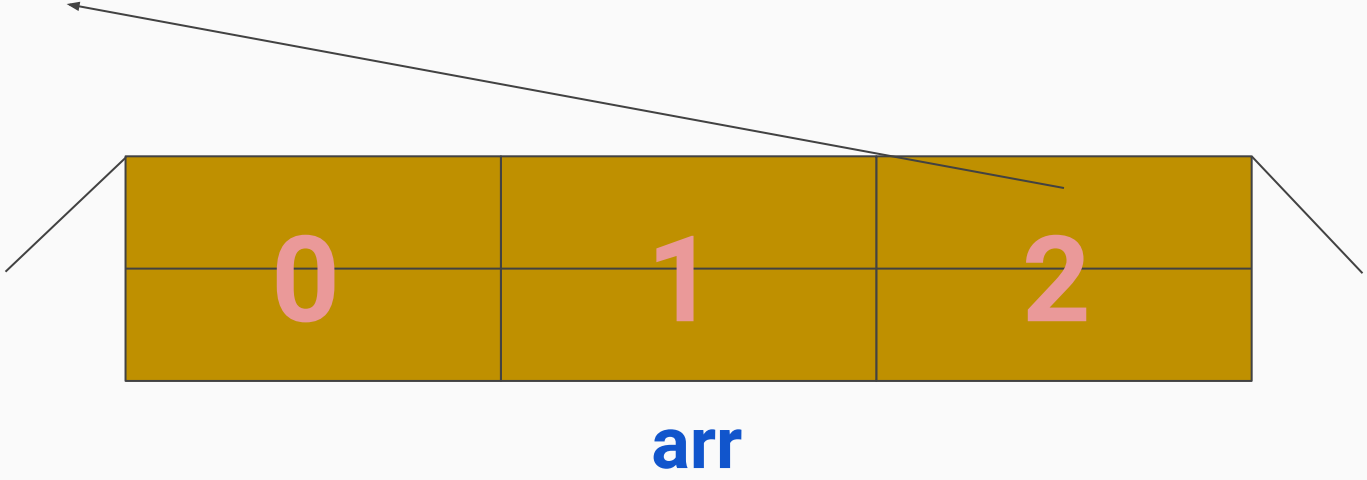
Array Syntax

```
int[] arr = {3,7,4};
```



Accessing Array Elements

```
arr[2];
```



Project: Using For Loops to Iterate Over Arrays

An Aside: The For, Each Loop:

A “for, each” loop (syntax below) makes iterating over an array (or any array-type structure, we will talk about alternatives later) much simpler. This is a common use of a for loop, so it is useful to know this syntax. Note that this doesn’t give you an index variable to work with, however.

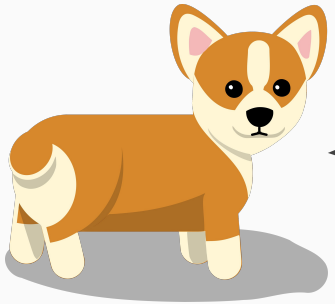
```
for(int o : arr) {
    //code
}
```

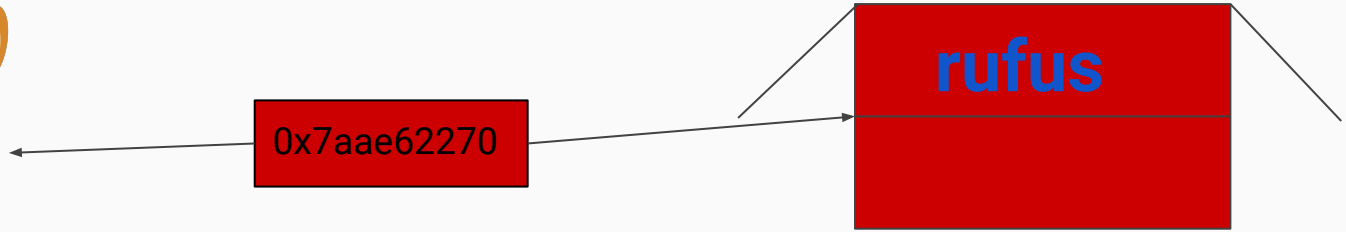
Object Oriented Programming: What Do We Know?

- A **class** is a file ending in the ".java" extension. All code in Java is contained inside of a **class**. **Classes** provide the code blueprint needed to construct **instances** of themselves, which are called **objects**.
- **Objects** have behaviors and states unique to them. **Classes** are blueprints for objects that define their possible behaviors and states, and allow them to be **constructed**.
- In order to access an object's states (called **fields**) and behaviors (called **methods**), we use a dot (.) after the object's identifier. We have seen this before many times, with Math.max(), or System.out.println()

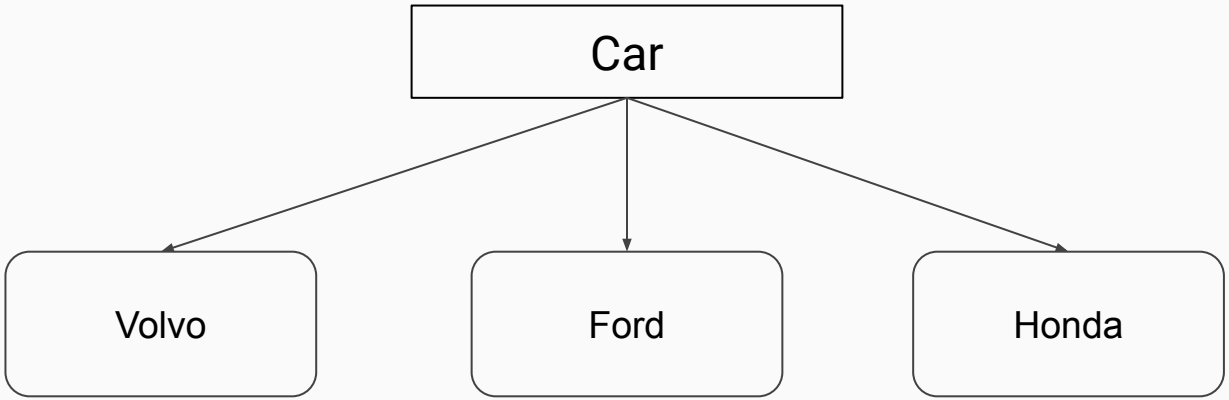
Variable Syntax

```
Dog rufus =
new Dog(7);
```

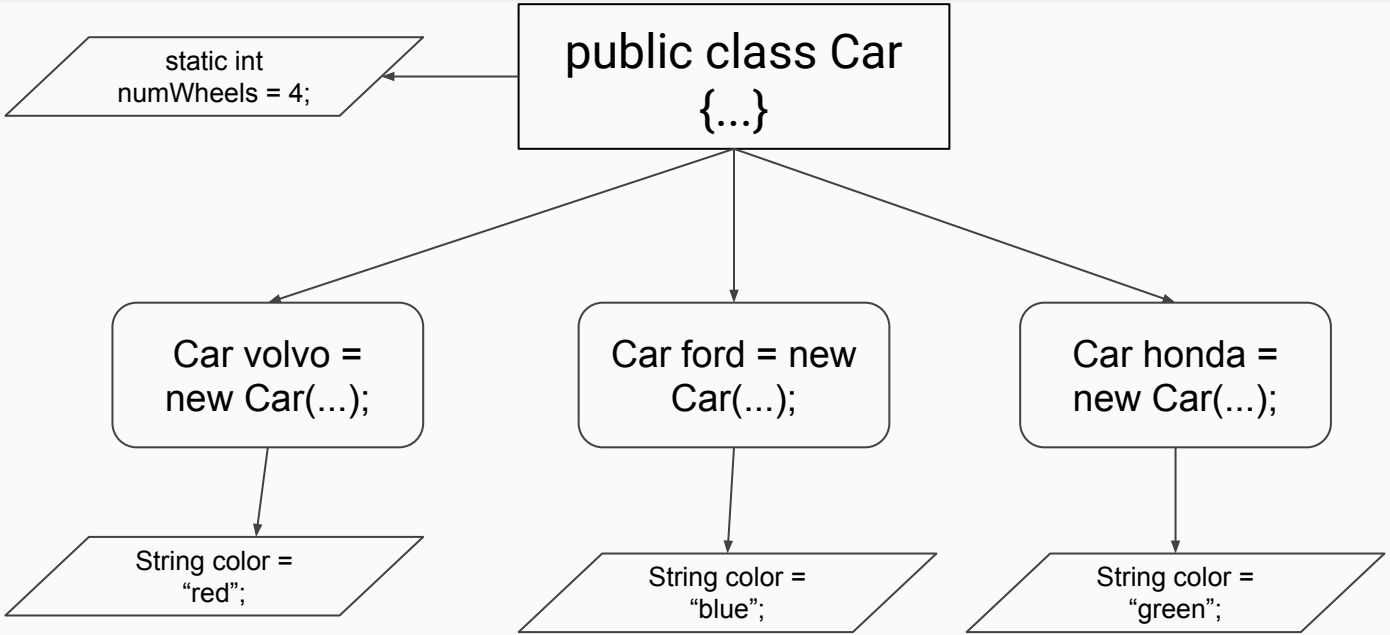




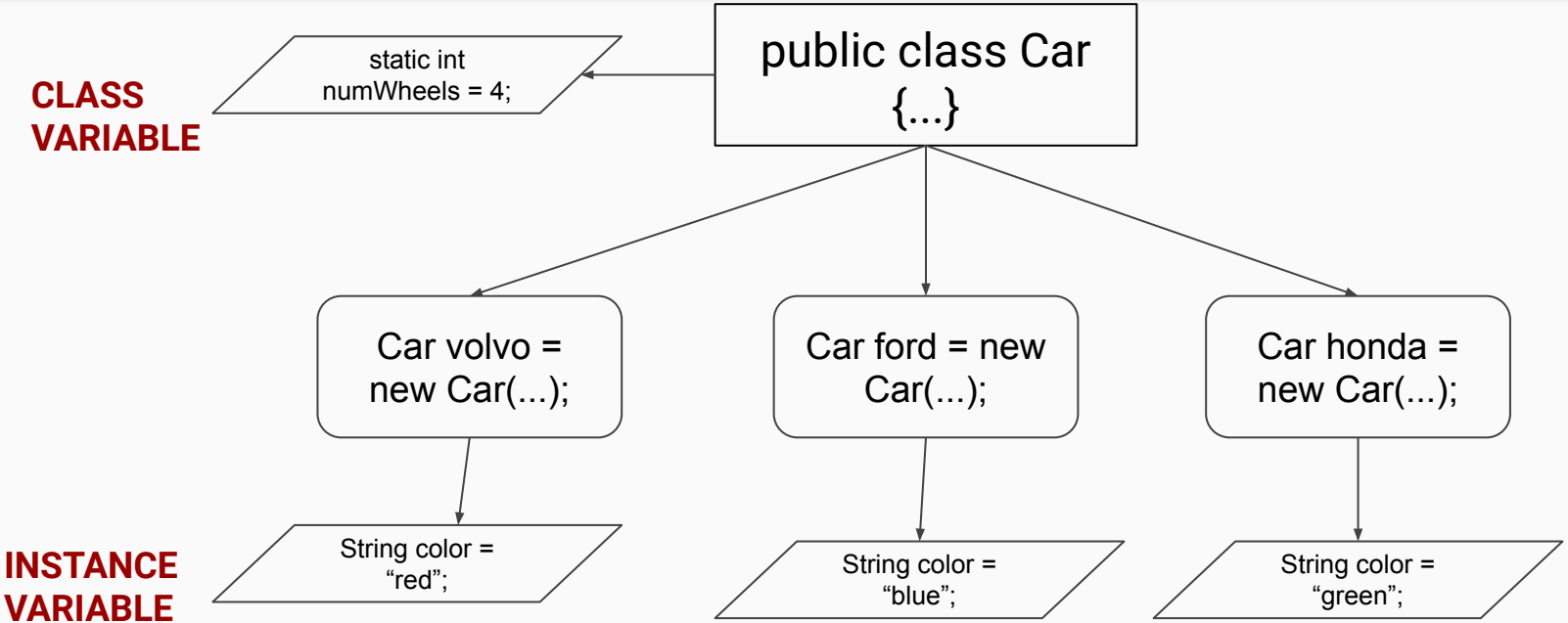
Let’s Take Another Look At This Diagram



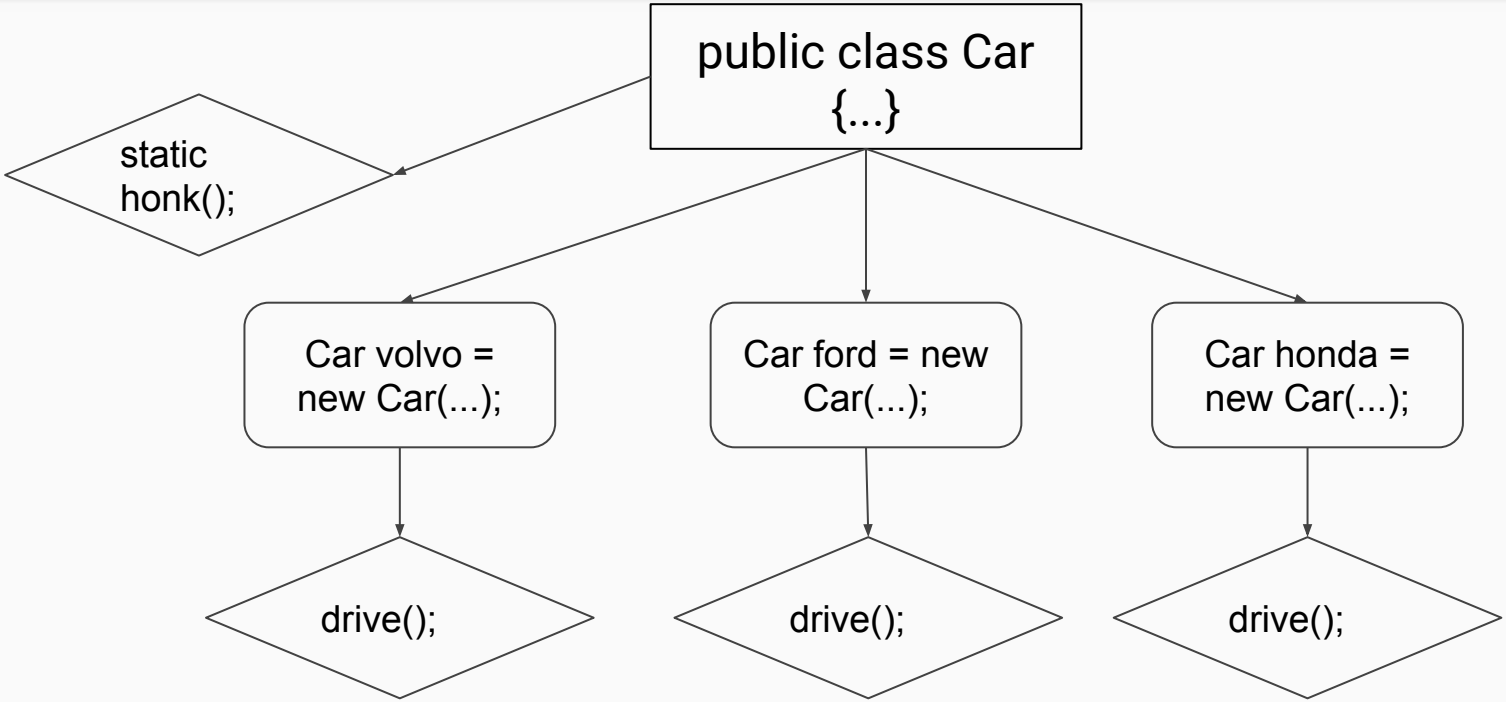
Object Oriented Programming in Java Syntax



Class Variables vs. Instance Variables



Object Oriented Programming in Java Syntax



Project: Hello World With Classes

Let’s take a look at classes, objects, methods, and encapsulation