CS 61BL LAB 2

Ryan Purpura

Slides up at rpurp.com

ANNOUNCEMENTS

- Project 0 finally released!
- Lab 1 has been given a 24 hour extension! Please get checked off today if you haven't gotten checked off yesterday (just call one of us over)
- There is a worksheet today! It is due at the end of the lab and is graded on correctness!

JAVA TYPES

- In Java, every variable has a type, and the type must be explicitly *declared* when you first use it
  - Syntax: `<type> <variable_name> = <value>;`
  - For example: `int x = 5;`
  - Note that you don't need to initialize immediately, so you could just do `int x;`

PRIMITIVES

- Primitives are a kind of data type in Java, which represent a basic unit of data.
- Examples of primitives
  - `int` is an integer between -2,147,483,648 and 2,147,483,647
  - `double` is a double-precision floating point number (in other words, can store decimal values like 1.61803)
  - `char` represents an ASCII character like `'A'`, `'%'`, `'+'`
  - `boolean` is either `true` or `false`

CLASSES

- In Java, classes represent a blueprint for data types called "objects".
- Objects can *do* things and *remember* things
  - Doing things is handled by *methods*
  - Remembering things is handled by *instance variables*
  - An object's methods and instance variables are defined by its class
- For now, think one class = one file and the filename must be the same as the class contained within

TYPES DEMO

DECLARING CLASSES

```
public class Car {
    private String myMake;
    private int myMilage;

    public Car(String make,
               int milage) {
        myMake = make;
        myMilage = milage;
    }

    public void drive(int distance) {
        myMilage += distance;
    }
    public int getMilage() {
        return myMilage;
    }
    public String getMake() {
        return myMake;
    }
}
```

It is recommended to keep your instance variables private.

The constructor is where you initialize the instance variables. The name is the same as the class name and has no return type.

Nothing is returned in a void method.

The `int` means that the method returns a `int`

The `String` means that the method returns a `String`

USING CLASSES

```
public class Main {

    public static void main(String[] args) {
        Car myCar = new Car("Tesla", 0);

        System.out.println(myCar.getMilage());
        System.out.println(myCar.getMake());

        myCar.drive(200);

        System.out.println(myCar.getMilage());
    }
}
```

Use the new keyword to construct an object by invoking a constructor

Use dot notation to access an object's methods

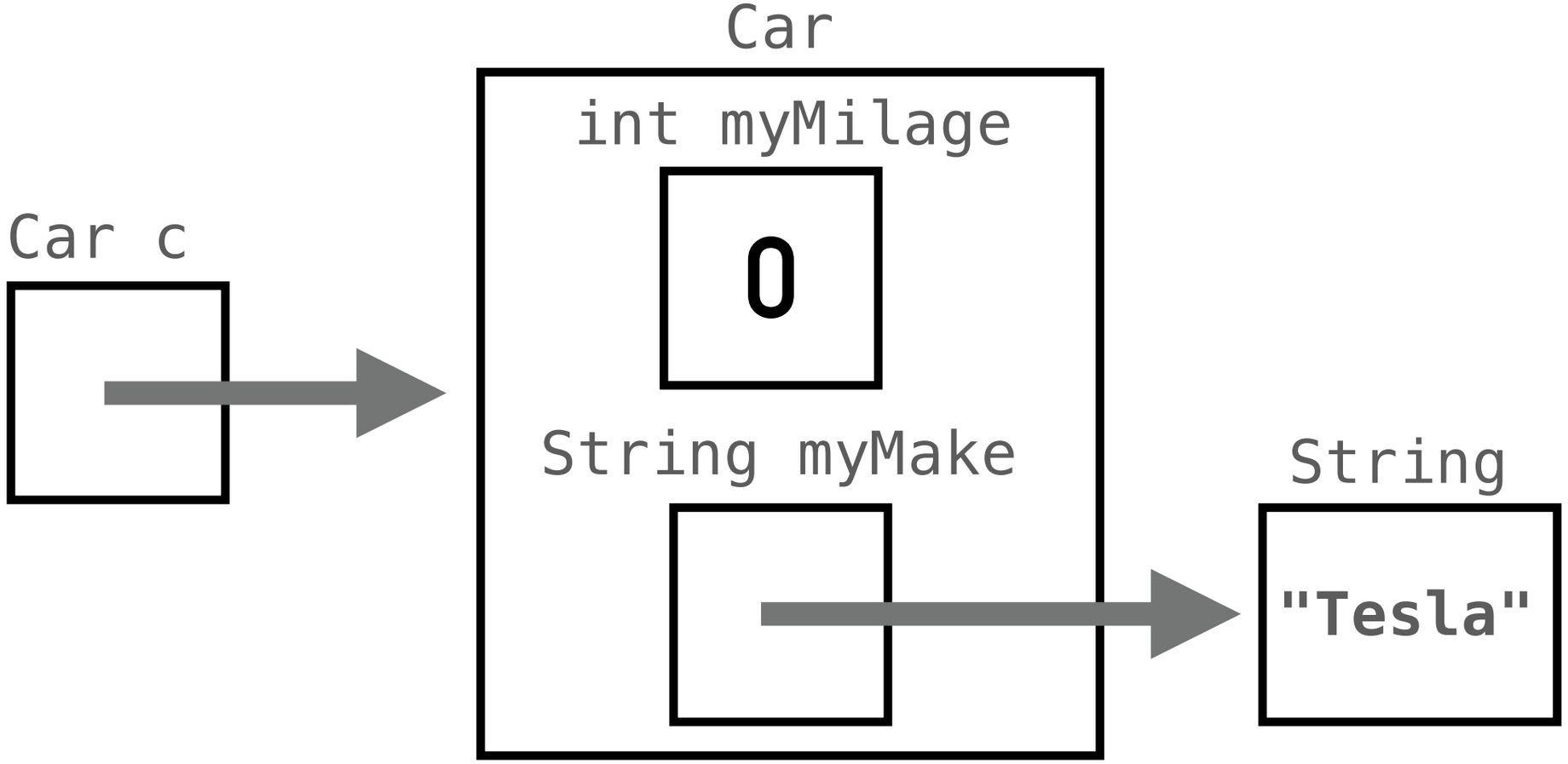
CONSTRUCTORS

- A class can have multiple constructors
- If you don't define a constructor, Java will make one for you that takes no arguments
- But if you define your own constructor, Java will *not* make you a no-argument constructor (but you can make your own!)

BOX AND POINTER NOTATION

```
➤ int x = 5;
```

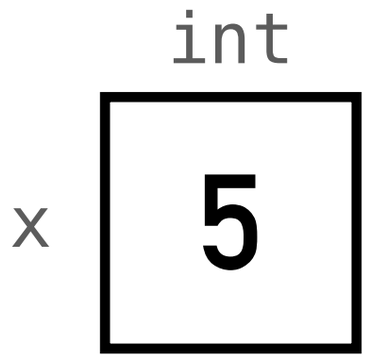

```
➤ Car c = new Car("Tesla", 0);
```



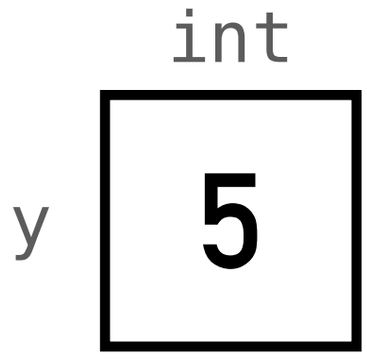
COPYING VARIABLES

- When you copy variables, you copy what's in the box

- **`int x = 5;`**

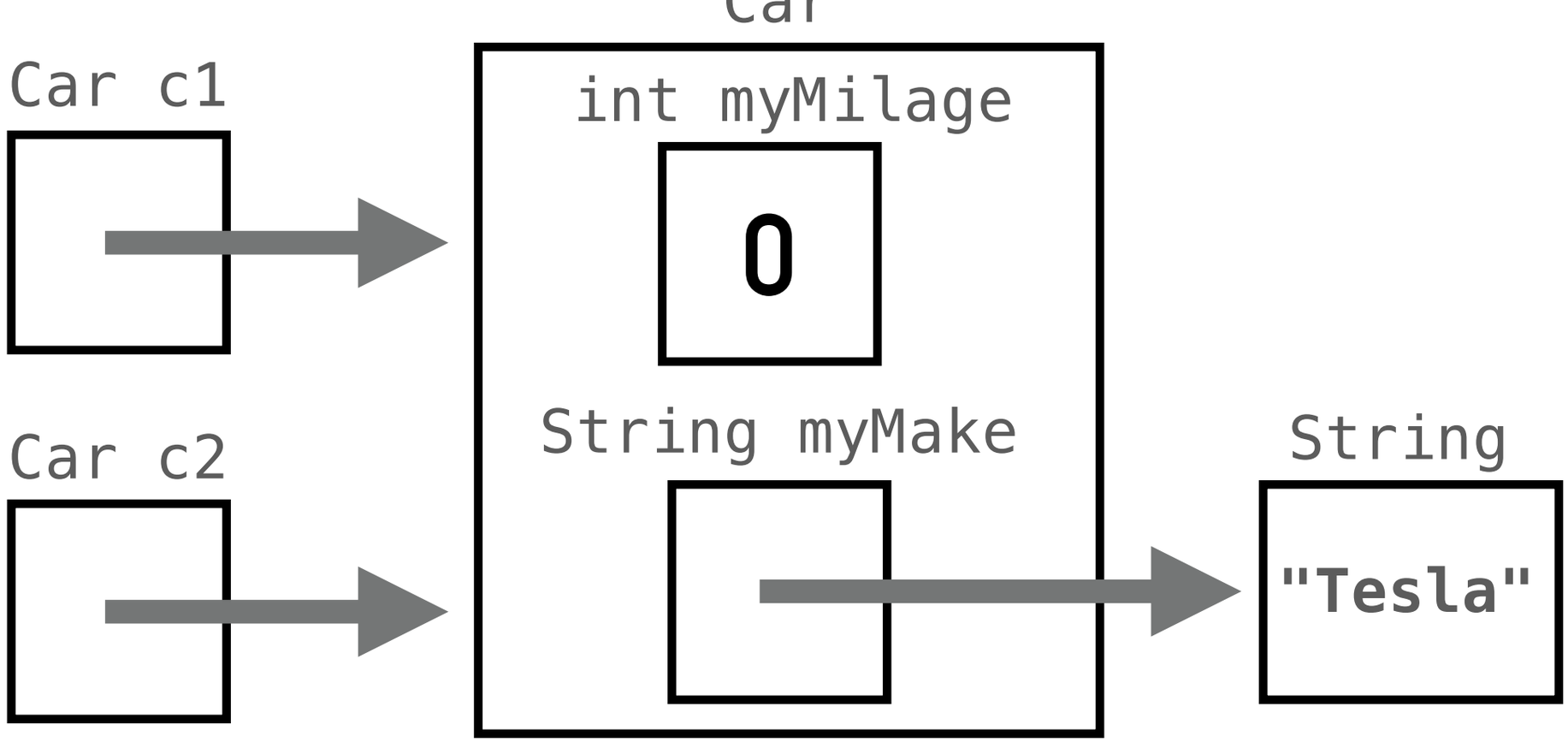


- **`int y = x;`**



COPYING VARIABLES 2

- For objects: the "thing in the box" is a pointer AKA a reference (arrow in this representation) to the data
    - `Car c1 = new Car("Tesla", 0);`
    - `Car c2 = c1;`



STATIC

- Static methods and static variables are associated with the class instead of just with one object.
- To use, prefix the method/variable declaration with "static"
- Reasoning: sometimes you want to be able to have a way to modify all the instances of a class or want all instances of a class to access the same variable.
- You can access static attributes as Class.staticVariable or Class.staticMethod()
    - If you have an object that is an instance of the class, you can also use object.staticVariable or object.staticMethod() [not recommended]