Static

So far we’ve seen

Classes:

- ”Types” of things (Car, Nim, Student)

Objects:

- Specific instances of class types
- My white Honda Civic, “game” (instance of Nim), s1 (instance of Student)

Objects store information and methods

- Each instance variable is attached to a particular object
- Each method is attached to a particular object

- I must instantiate an object before I can store any data, or access any method, of a class!

Example

Static keyword

- Static variables and methods are attached to the *class* (the type), not the *object* (not a specific instance)

- Why is this useful?
    - Can call methods (and access information) without instantiating an object
    - Can keep track of variables and methods that apply to the class as a whole

Using static (example)

Rules of thumb

When to use static for variables?

- Ask yourself: is this variable going to change for different instances of the class?

- Example: Rectangle class. Store `numSides`, and `sideLength`. Which should be static?

When to use static for variables?

- Ask yourself: is this variable going to change for different instances of the object?
- Example: Rectangle class. Store `numSides`, and `sideLength`. Which should be static?
- Also remember: a variable must be static if it’s accessed by a static method

When to use static for methods?

- Does it need to access any non-static variables? If so, it can’t be static!
- Is it called by any static methods? If so, it *must* be static!
- Do we want to use it before instantiating an object of the class?

Quick note: we have already seen static!

```
public static void main
```

- Makes sense---we call this function without instantiating an object

- Above rules apply! (Can only call static methods from main; compiler will complain otherwise)