INHERITANCE

MOTIVATION

- Physically, items often share common aspects
- Example:
  - cars, semis, motorcycles are all road vehicles
  - all have wheels (different #s)
  - some have windows, ...

MOTIVATION

- Coding each separately would repeat any common aspects
- Logically different -> makes sense to code as different things
- How can we represent and capitalize on logical relationship?

INHERITANCE - BASIC IDEA

- Represent connections between classes via inheritance
- parent/super/base class
- child/sub class
- allows for code/software reuse

IS-A VS HAS-A

- "has-a" relationship represents variables in classes
- "is-a" relationship signals potential child
- Example:
  - mustang is a car
  - car is a road vehicle
  - car has windows, trunk
- Example:
  - horse is a mammal
  - dog is a mammal
- If X is derived from Y, should be able to say X is a Y

IMPLEMENTATION

- In Java, denoted with `extends` keyword

```
public class Car extends Vehicle
{
    //code here
}
```

WHAT HAPPENS

- child inherits methods and variables from parent
- parent gets nothing from child
- private methods/variables cannot be referenced
    - still exist
- constructors not inherited

CONSTRUCTORS

- What if you need/want to use parent constructor?
- `super` reference -> references parent
- could just set same variables, but better practice to let parent class handle
- call to super should be first line
- no call -> automatically calls parent with no parameters
- no explicit parent -> implicit is `Object`

SINGLE VS MULTIPLE

- Java only allows single inheritance
- One class can't inherit from 2 parents (multiple inheritance)
- But, multiple classes can inherit from same parent (siblings)

OVERRIDING METHODS

- Defining method with same name overrides parent
- Very common
- Examples:
  - `toString`
  - `equals`
- Not the same as method overloading
- `final` methods cannot be overridden