# In this lecture, we will discuss…

- ✧ The “||” operator
- ✧ Class methods and class variables
- ✧ Class inheritance

# var = var || something

- || operator **evaluates the left side**
  - If **true** – **returns** it
  - **Else** – **returns** the **right side**
  - `@x = @x || 5` will return `5` the **first** time and `@x` the **next** time

- Short form
  - `@x ||= 5` – same as above

# var = var || something

```ruby
class Person
  attr_reader :age
  attr_accessor :name

  def initialize (name, age) # CONSTRUCTOR
    @name = name
    self.age = age # call the age= method
  end
  def age= (new_age)
    @age ||= 5 # default
    @age = new_age unless new_age > 120
  end
end
person1 = Person.new("Kim", 130)
puts person1.age # => 5 (default)
person1.age = 10 # change to 10
puts person1.age # => 10
person1.age = 200 # Try to change to 200
puts person1.age # => 10 (still)
```

# Class Methods and Variables

- ✧ **Invoked** on the **class** (as opposed to instance of class)
  - Similar to `static` methods in Java
- ✧ `self` **OUTSIDE** of the **method definition** refers to the `Class` object

# Class Methods and Variables

- **Three** ways to **define class methods** in Ruby
- Class variables **begin** with `@@`

# Class Methods and Variables

```ruby
class MathFunctions
  def self.double(var) # 1. Using self
    times_called; var * 2;
  end
  class << self # 2. Using << self
    def times_called
      @@times_called ||= 0; @@times_called += 1
    end
  end
end
def MathFunctions.triple(var) # 3. Outside of class
  times_called; var * 3
end

# No instance created!
puts MathFunctions.double 5 # => 10
puts MathFunctions.triple(3) # => 9
puts MathFunctions.times_called # => 3
```

`self` outside of method refers to `Class` object

# Class Inheritance

- ✧ Every class **implicitly inherits** from `Object`
  - `Object` itself inherits from `BasicObject`
- ✧ **No** multiple inheritance
  - **Mixins** are used instead

# Inheritance

```ruby
class Dog
  def to_s
    "Dog"
  end
  def bark
    "barks loudly"
  end
end
class SmallDog < Dog
  def bark # Override
    "barks quietly"
  end
end

dog = Dog.new # (btw, new is a class method)
small_dog = SmallDog.new
puts "#{dog}1 #{dog.bark}" # => Dog1 barks loudly
puts "#{small_dog}2 #{small_dog.bark}" # => Dog2 barks quietly
```

Implicitly inherits from `Object`

< Denotes inheritance

# Summary

- **Class inheritance** lets you **override** parent's behavior
- Class methods **do not** need an instance of object in order to be **called**
- Class variables begin with `@@`

**What's next?**

- Modules