Ruby on Rails BootCamp

Ruby Day 2

RegExp!

- Regular Expressions in Ruby

Patterns

- Find a pattern in text

/find me/

- “Hello World” =~ /World/
- pattern = /World/
- pattern.match “Hello World”

Rubular

- http://www.rubular.com/

TitleCase, snake_case

- “this is supposed to be a title”.gsub(/\b\w/) { |w| w.upcase}
- “this should be a snake case”.gsub(/\b\W/){ |b| ‘_’}

Exercise!

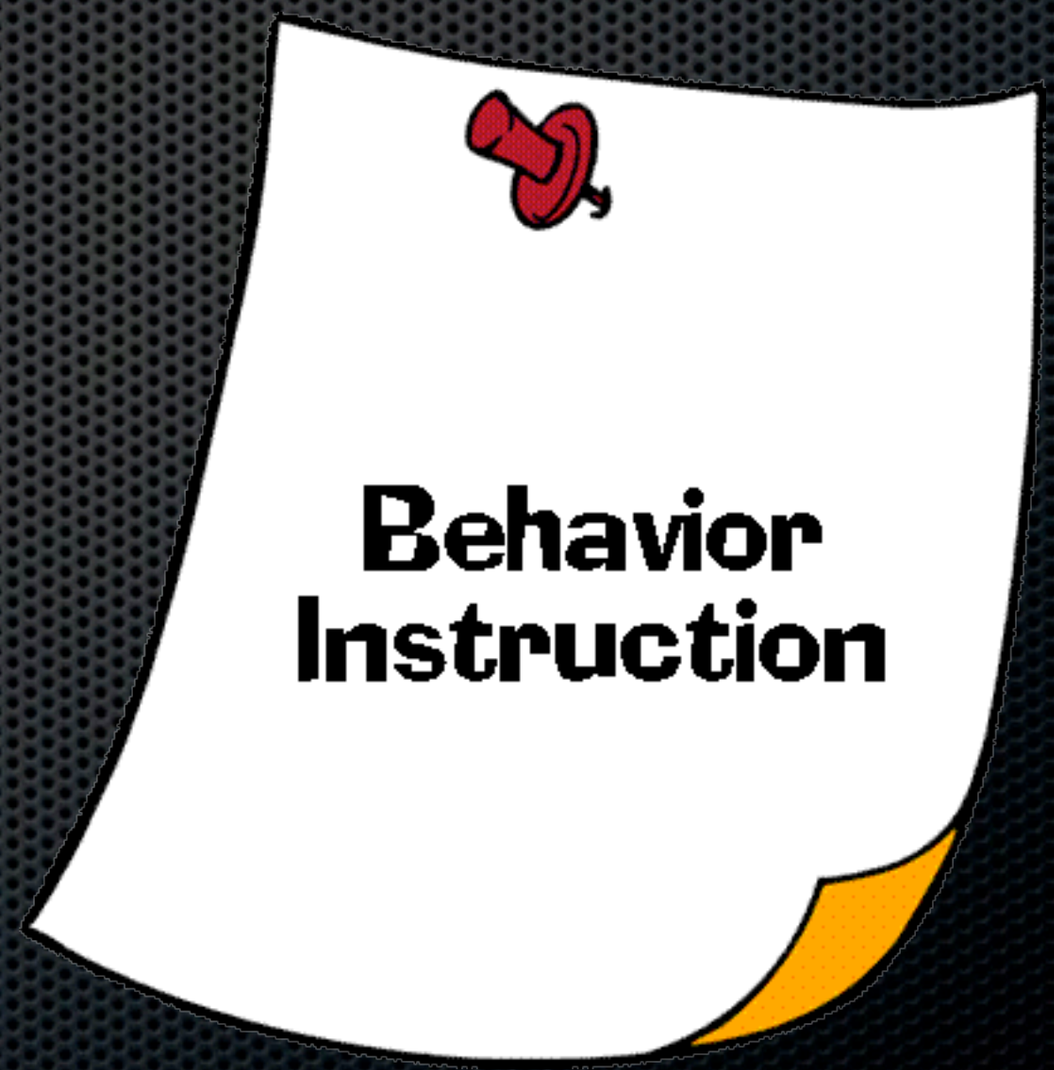
- pig_latin_spec.rb

Cucumber or Rake?

Poll the class.

Behavior Driven Development

- User Stories
- Business Value
- Communication
- Shared Tools

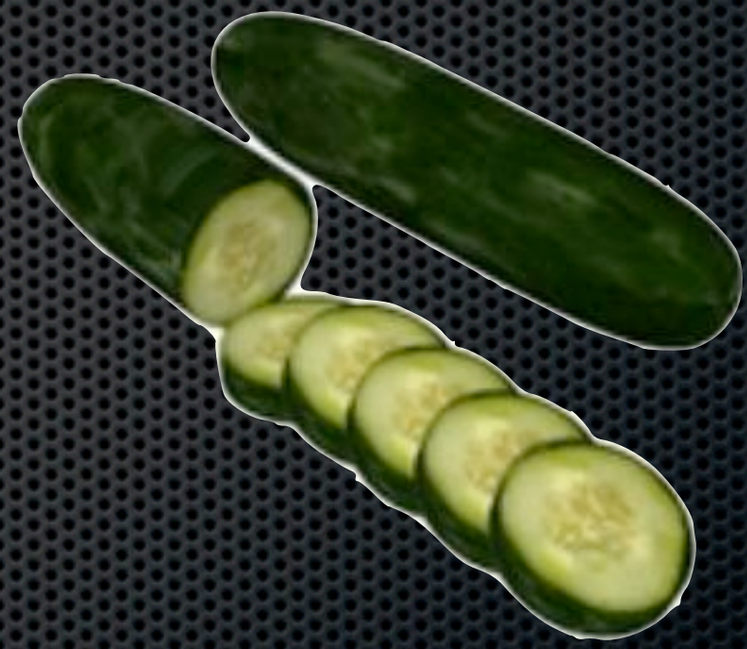


Behavior Specs

- Title
- Narrative
- Acceptance Criteria / Scenarios / Outcomes

Cucumber

behaviour driven development with elegance and joy



- gem install cucumber
- cucumber --help

Features

- Human Read/Writable description of behavior
- Business Users write the features, Programmers develop the features to spec
- https://github.com/cucumber/cucumber/wiki/Feature-Introduction

Given-When-Then

- Describing your feature
- Preconditions
- Something happens
- Expected outcome

Step Definitions

- REGEXP!!
- Translation between the human narrative and the code
- This type of sentence, matches to this type of code

Demo!

- I want a Celsius to Fahrenheit converter
- I tell my story
- I copy the generated regex
- I write my tests
- TDD!

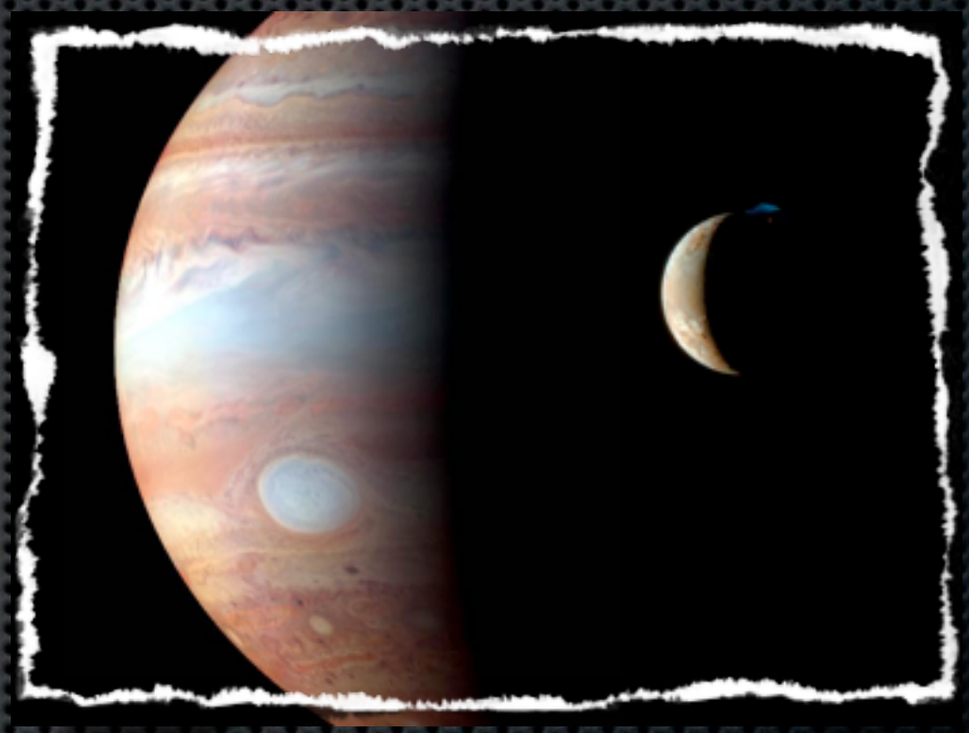
Example!

- New Feature!
- I’m in London and I can’t share bar stories without being able to convert Fahrenheit to Celsius
- Write the Steps
- Make Them Pass!

IO, Files, and Rake

Streams and Tasks

IO



- Input and Output
- Reading in, Writing out
- Working with the external (real!) world

Files

- file = File.open(“test_file”, “r+”)
- "r" Read-only, starts at beginning of file (default)
- "r+" Read-write, starts at beginning of file
- "w" Write-only, truncates existing file to zero length or creates a new file for writing
- "w+" Read-write, truncates existing file to zero length or creates a new file for reading and writing

Files with Blocks

- File.open(“test_file”, “r+”) do |r|
  - r << “Add some text”
- end
- f = File.open(“roster.txt”)
- f.each{ |line| puts line}
- f.map{|line| line.split(‘,’).first}

Files

- f = File.open(“roster.txt”)
- f.gets (get a line)
- f.pos (character position)
- f.rewind (pos = 0)
- f.read (file as a string)
- f.readlines (array of each line as a string)

Directories

- Dir.pwd
- Dir.chdir(“..”)
- All files: Dir[“*”] All hidden files: Dir[“.*”]
- d = Dir.new(“.”)
- Dir.mkdir(“test”)
- d.entries
- d.count
- d.each{|f| puts f}

Rakefile

- Ruby Make
- Ruby scripting
- Ordered Tasks (configuration!)

Rake tasks

- task :name => [:dependency] do
  - <ruby code>
- end

Exercise

- Create a task that reads all the lines in names and outputs them
- Create a task dependent on the class directory task that makes a directory in the class directory for each name in names

Advanced Ruby

With Great Power...

Blocks

- A method with no name!
- Dynamic method definition

Block Syntax

- {}
- do
  - <code>
- end

Blocks: Calling and Checking

- yield
- call
- block_given?

Blocks with Parameters

- yield(x)
- call(x)
- {|x|}
- do |x|
  - <code>
- end

Blocks Demo

- Time My Code!
- timer_spec.rb

Blocks Exercise

- Do My Work!
- worker_spec.rb

Magical Ruby

AKA MetaProgramming

MetaProgramming

- Code That Writes Code!!
- DSLs
- DRY

MetaProgramming

- send
- instance_eval
- class_eval
- instance_variables
- instance_variable_set /instance_variable_get
- define_method /undef_method

MetaProgramming

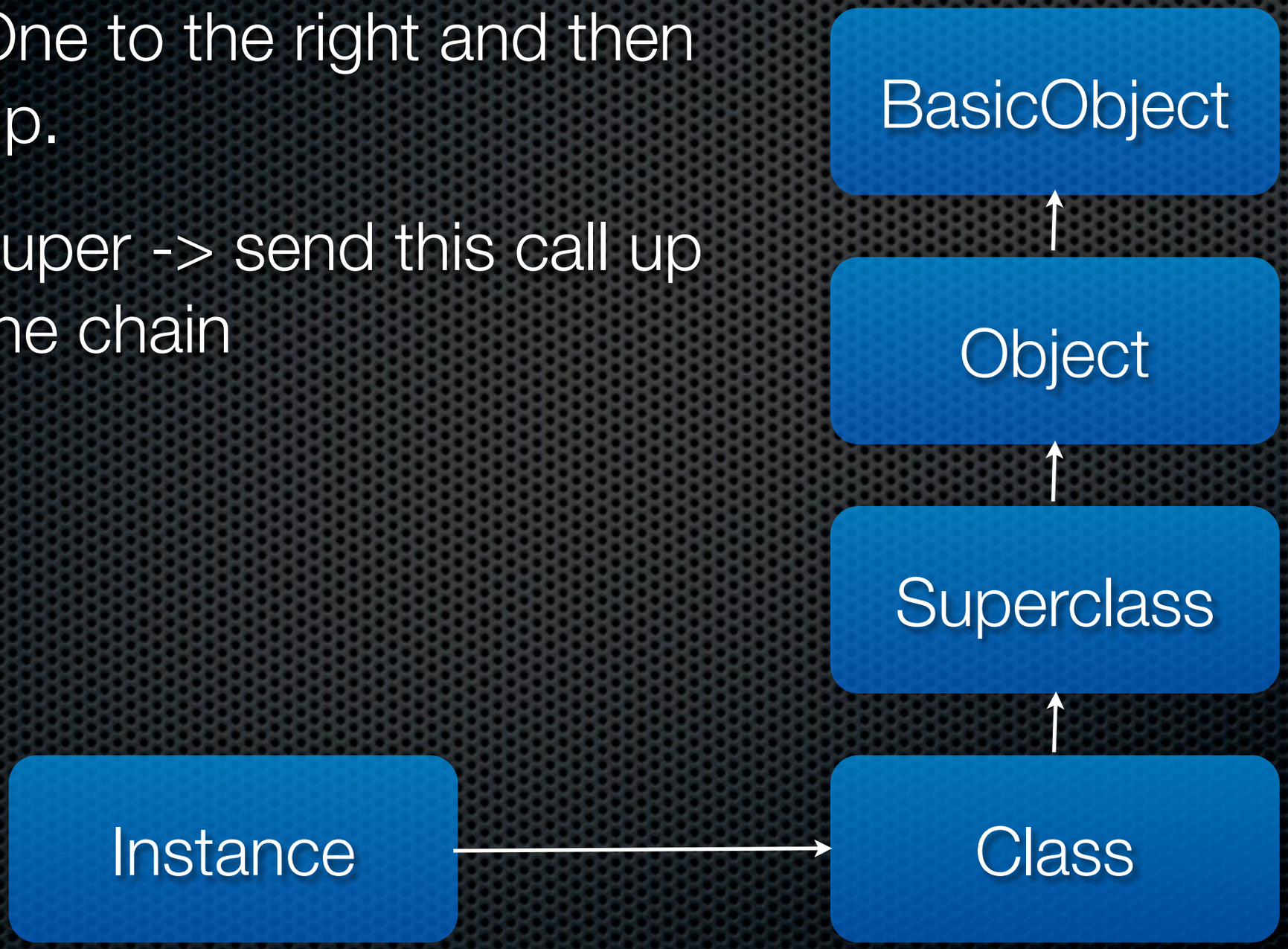
- Send: pass a message to an object, same as a method call, but can use a string.
- eval: Remember everything is just evaluated as ruby code. We are telling Ruby to run some code within a certain context (self)

Where do Methods Live?

- Klass.class_eval => Gives you an instance method for all objects instantiated from Klass.new
- Klass.instance_eval => Gives you a class method for class Klass.
- Klass.new.instance_eval => Gives you an instance method for that instance alone.

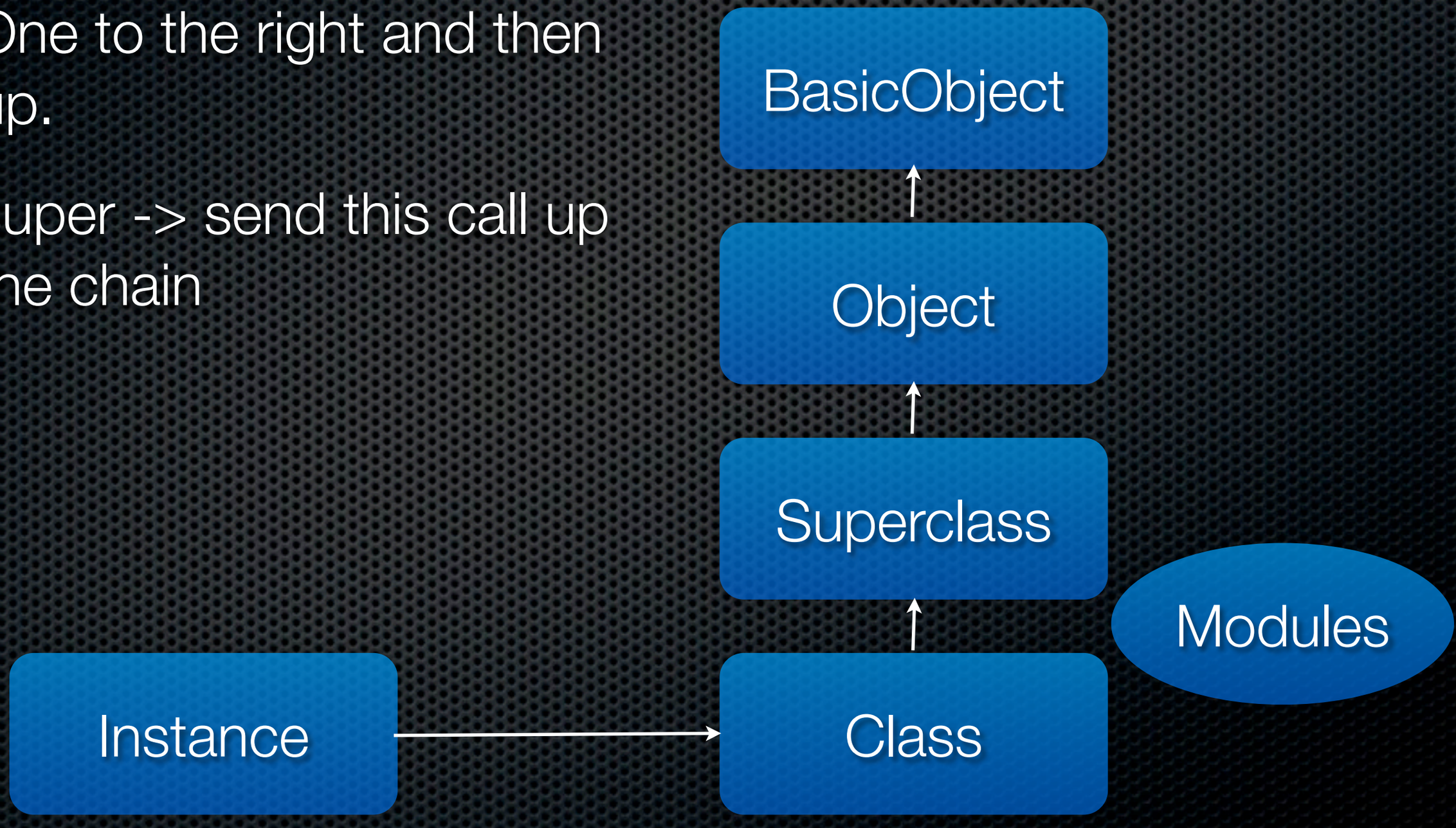
Ruby Call Chain

- One to the right and then up.
- super -> send this call up the chain



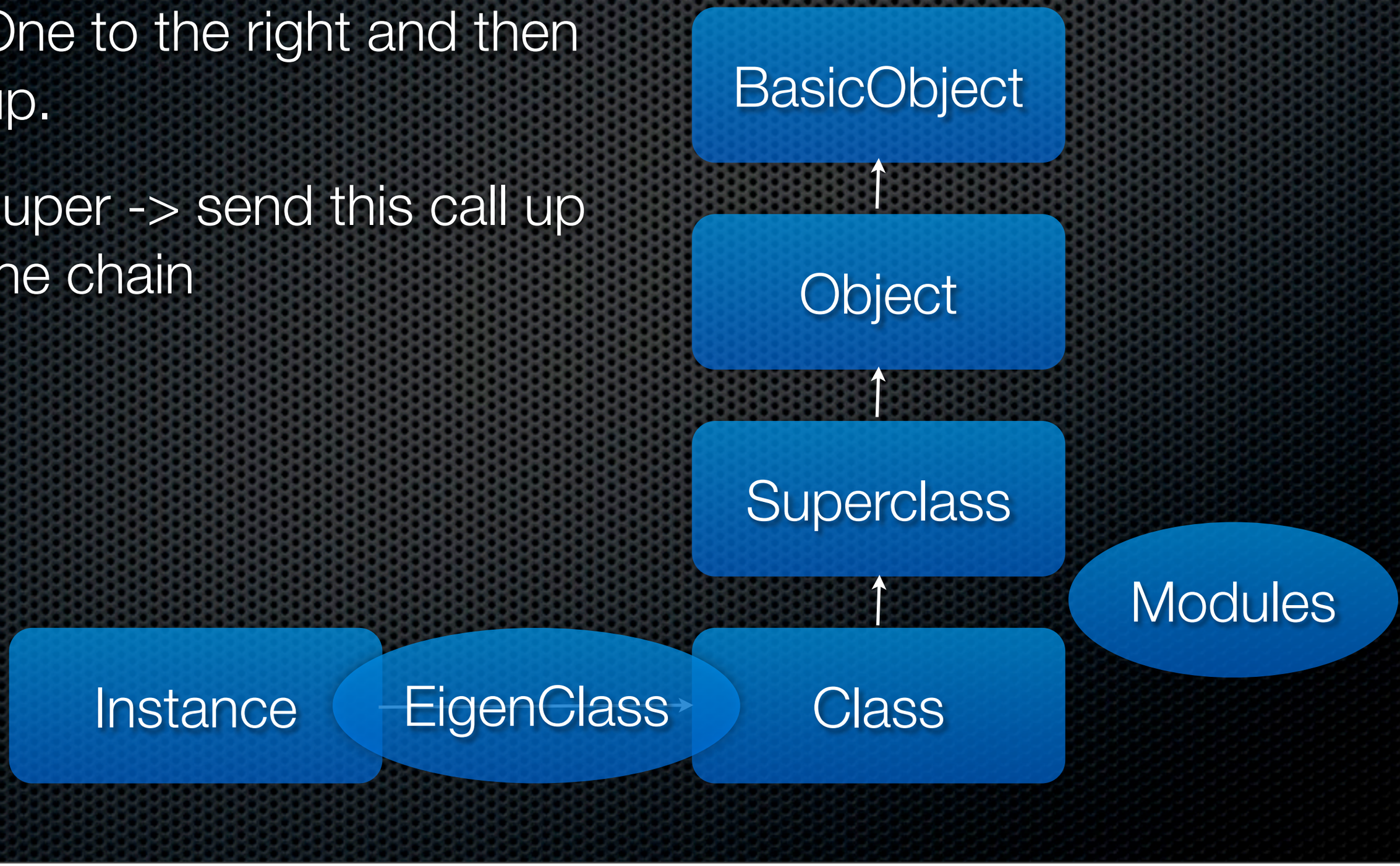
Ruby Call Chain

- One to the right and then up.
- super -> send this call up the chain



Ruby Call Chain

- One to the right and then up.
- super -> send this call up the chain



Experiment!

```
def call_chain
        "#{self}.#{super}"
end
```

Object, Animal, Speaker, Person, NamedThing, Renee

Demo!

- DRY-up some code!
- couch.rb

Exercise!

- You try!
- exercises/couch.rb

Method Missing!

- Up the chain, then back!
- The final resting place of method calls (most of the time!)
- It’s Magic, and you can too! :)

Method Missing

```
def method_missing(sym, *args, &block)
        puts "You asked for #{sym} with #{args.join(" ")}"
        super
end
```

**def respond_to?

DuckTypeing

- If it quacks like a duck
  - respond_to?(:quack) => true
- Who cares if it’s a duck?

Code Like a Duck

BAD:

```
case book.class
        when FictionBook
                puts “This book is Fiction!”
        when TextBook
                puts “This book is for School!”
        end
```

DuckTypeing

```
if book.respond_to? :print_out
    puts book.print_out
end
```

Monkey Patching!

- Open a class and add, extend, fix, or change (break) functionality!
- With great power.... ;)