Rails from the Ground Up!

with Jared Richardson
RoleModel Software

Ruby
Part One
An Introduction

Ruby the Language

Ruby != Rails

Lightweight language

Cross platform

Interpreted

Older than Java

Matz!

Yukihiro “matz” Matsumoto

Ruby’s first release was 1995

Blended His Favorites

Perl

Smalltalk

Eiffel

Ada

Lisp

His Goals?

Balance functional and imperative

Wanted Ruby to be natural, not simple

Didn’t Take Off Until

Ruby on Rails

by DHH (David Heinemeier Hansson)

Released in ~2005

(Ruby had been out for a decade!)

Be Aware

Ruby and Ruby on Rails are different

A Quick Aside

Ruby Version Manager

aka rvm

https://rvm.beginrescueend.com/

installs

manages

Start here...

The following is a single line:

bash -s stable < <(curl -s https://raw.github.com/wayneeseguin/rvm/master/binscripts/rvm-installer)

Put rvm in the Path

Reload your environment

```
source ~/.bash_profile
```

OR

exit Terminal and start a new session

Requirements

Check platform specific requirements

```
rvm requirements <ruby version>
```

```
rvm requirements 1.9.3
```

Install 1.9.3

```
rvm install 1.9.3
```

Now Wait...

rvm will...

-download

-compile

-install

What's Installed?

```
rvm list
```

Choose Your Ruby!

```
rvm use <version>
```

Useful Commands

```
rvm default <version>

rvm info

rvm remove
```

Something's Wrong!

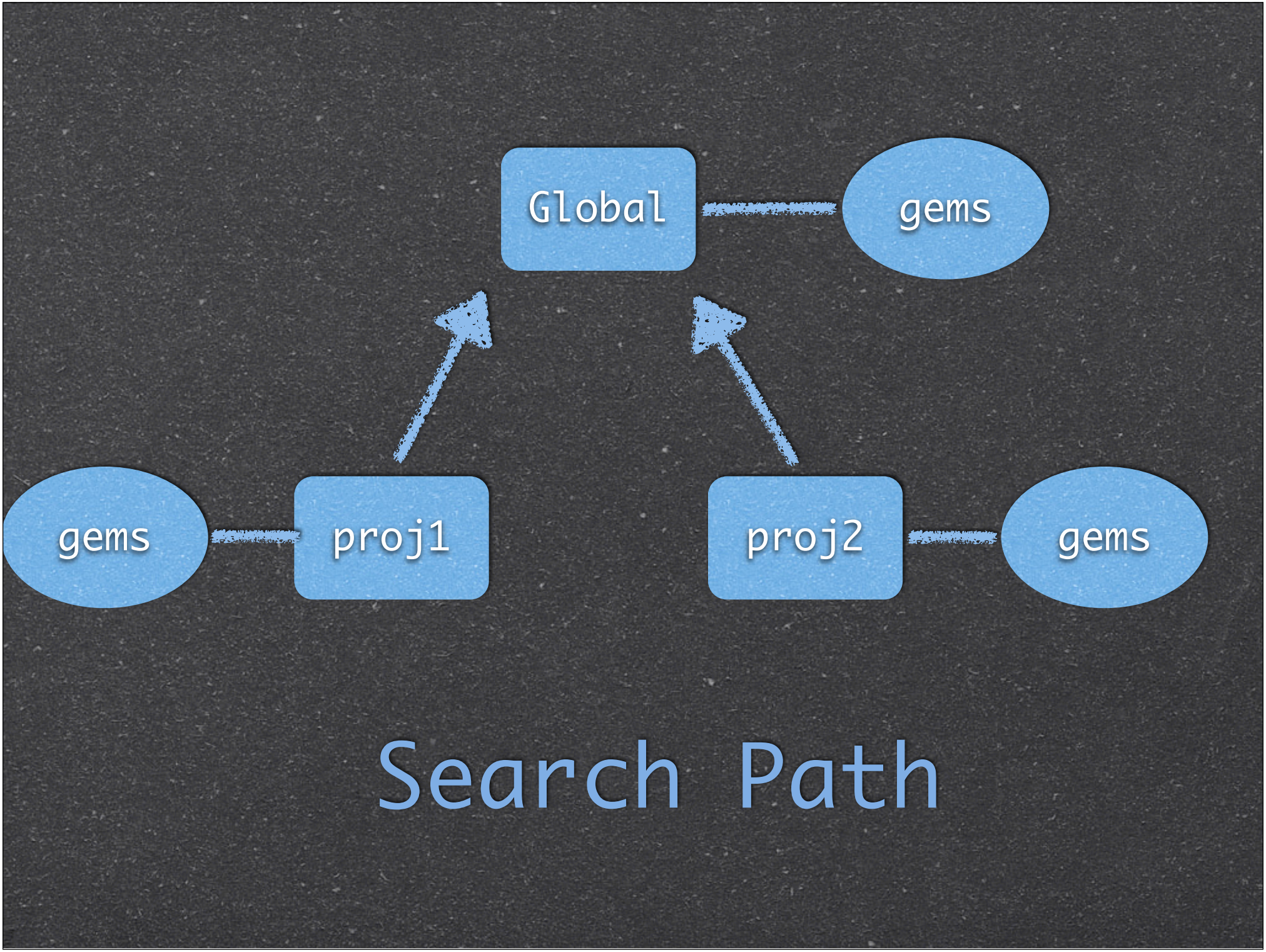
As a last resort...

rvm implode

http://www.flickr.com/photos/22320444@N08/5269558283/

Name Spaces

```
rvm gemset

rvm gemset create development

rvm gemset create test
```



Search Path

.rvmrc

```
rvm use 1.9.3

rvm use 1.9.3@development

rvm use 1.9.3@project_two --create
```

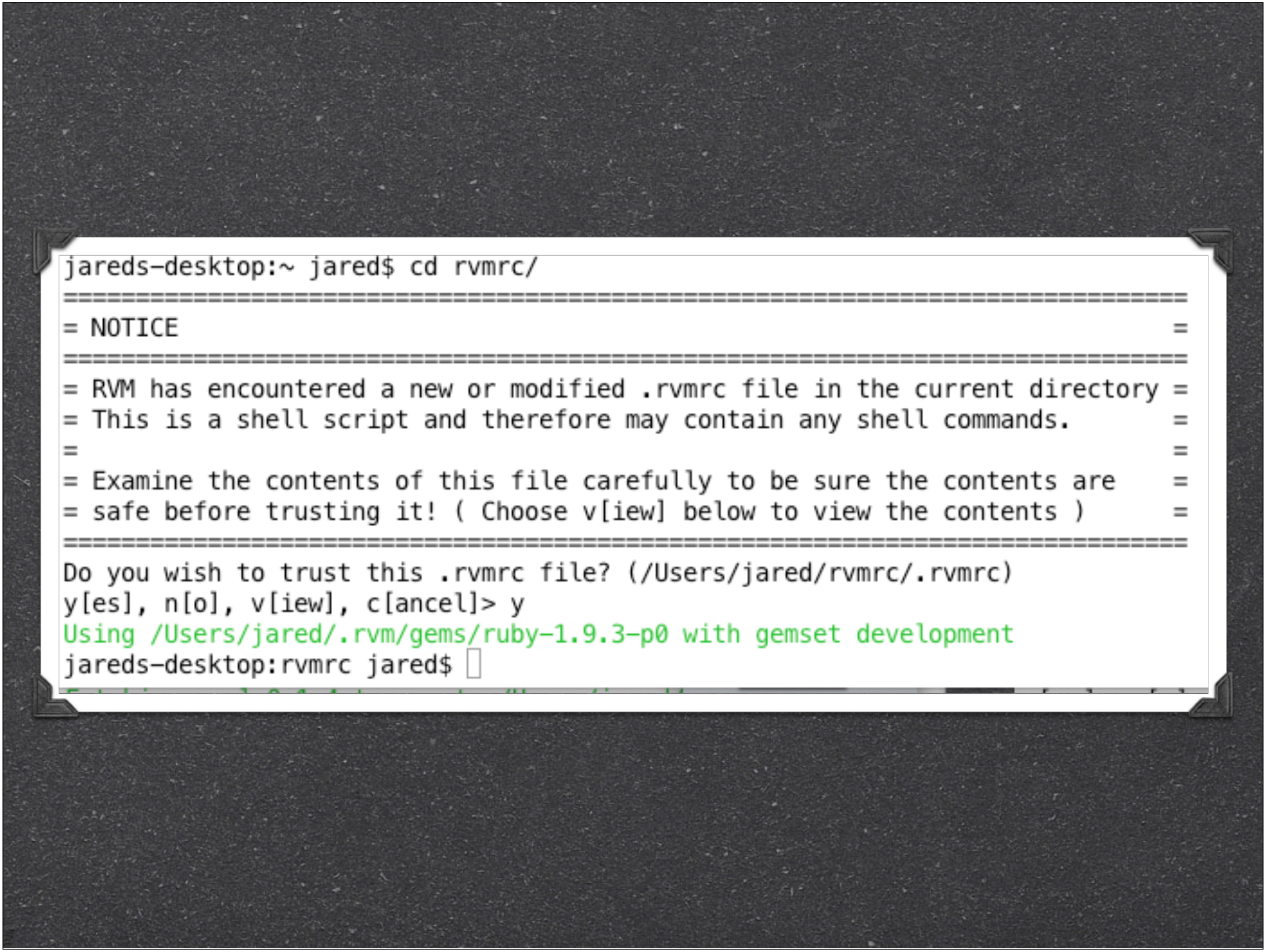
.rvmrc

Ensures Ruby version by project

cd into a directory

rvm switches to the correct Ruby

Libraries (gems) are Ruby specific


jareds-desktop:~ jared$ cd rvmrc/
= NOTICE =
= RVM has encountered a new or modified .rvmrc file in the current directory =
= This is a shell script and therefore may contain any shell commands. =
= Examine the contents of this file carefully to be sure the contents are =
= safe before trusting it! (Choose v[iew] below to view the contents) =
Do you wish to trust this .rvmrc file? (/Users/jared/rvmrc/.rvmrc)
y[es], n[o], v[iew], c[ancel]> y
Using /Users/jared/.rvm/gems/ruby-1.9.3-p0 with gemset development
jareds-desktop:rvmrc jared$

Your Turn

```
in RailsFromTheGroundUp

cd rvmrc

cd 1.9.3

cd ..

cd 1.9.3_development
```

rvm create

```
rvm gemset create development
```

Back to Ruby

```
ruby <file_name>
```

```
irb
```

irb

```
“Hello World!”

2 + 2

Math.sqrt(9)
```

Everything's An Object

```
"Hello".class

3.class

(3/7).class

(3/7.0).class

3/7.class
```

Objectify

```
obj = Object.new

obj.is_a? Integer

obj.is_a? Object

obj.methods
```

Acting On Objects

object dot method

obj.methods => returns an Array

Chain methods

obj.methods.sort

Simple Math

```
4-3

3-7

2 * 9

9/2

9/2.0
```

Numbers

Integer

Fixnum & Bignum

Float

Double

Rational

Fixnum

Inherits from Integer

Native machine word (minus 1 bit)

Auto converts to Bignum

Max & Min?

Not defined

Platform specific

```
FIXNUM_MAX = (2**(0.size * 8 -2) -1)
FIXNUM_MIN = -(2**(0.size * 8 -2))
```

Your Turn

use irb

```
n = 2
```

```
r = 2**10
```

what class is r?

```
r = n**250
```

what class is r?

find out where r becomes a BigNum

Conversions

Type along...

```
i = Integer (0xA4F3A)

i = Integer “7321”

i.to_s

f = Float(“732.217”)

f.to_i
```

Strings

```
h = “Hello World!”

h.length()

h.reverse()

h.index(“d”)

h.upcase

h.downcase!
```

More Strings

```
h = "Hello" + " " + "world!"

"Hello ".concat("World")

h.sub("l", "k")

h.gsub("l", "k")

h.chomp
```

Quoted Strings What's Valid?

```
"Hello world"

'Hello world'

"Hello world'

" 'this is my' little world"

' "this is my" other little world'

" this is \" valid"
```

Embedding Variables in Strings

```
" #{ <variable> }  "

n = 17

puts "The value of n is #{n}"

puts 'The value of n is #{n}'
```

Recap

Terminal commands

gcc

Install of brew

git & SourceTree

rvm

basic ruby

Homework

http://www.ruby-lang.org/en/

http://git-scm.com/doc/ext

Recap

Terminal commands

gcc

Install of brew

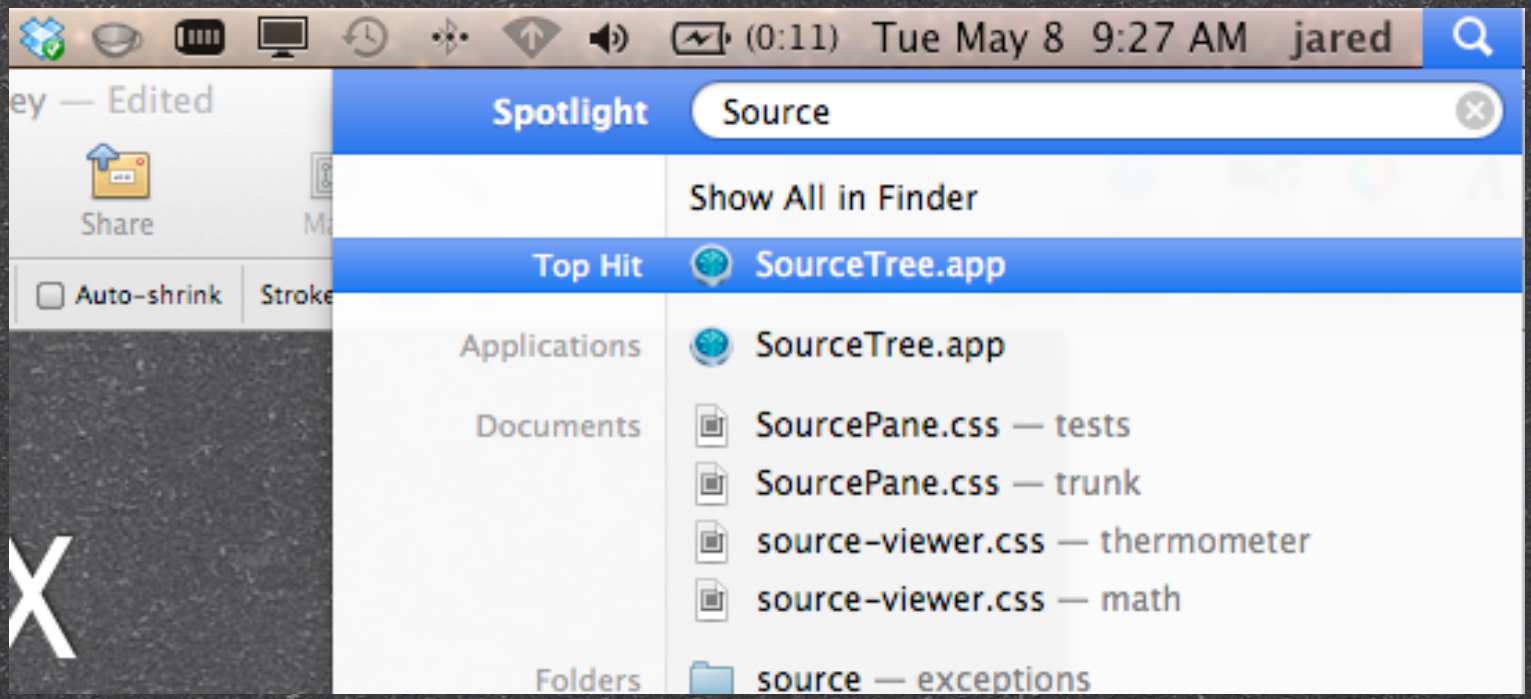
git & SourceTree

rvm

basic ruby

OS X

Use Spotlight to launch applications



OS X Terminal

```
cd

cd ..

cd ~

more <file_name>

cp <source> <dest>

mv <source> <dest>
```

OS X Terminal

Starting terminal runs ONE of .bashrc, .bash_login, .bash_profile, or .profile

Edit one. Delete the others.

OS X Terminal

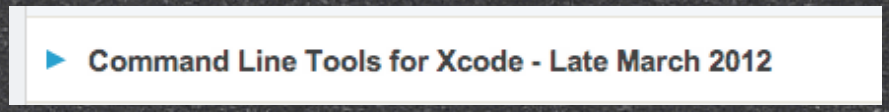
```
rm <file_name>

grep <seach_string> <files>

grep -R <search_string> <files>

open <file_name>

open <directory>

open .
```

gcc

Download from Apple Store

Command Line XCode Tools

https://developer.apple.com/downloads/index.action



brew

http://mxcl.github.com/homebrew/

A tool to compile and install tools

git

http://git-scm.com/

Distributed version control

Everyone has the entire history

Fast

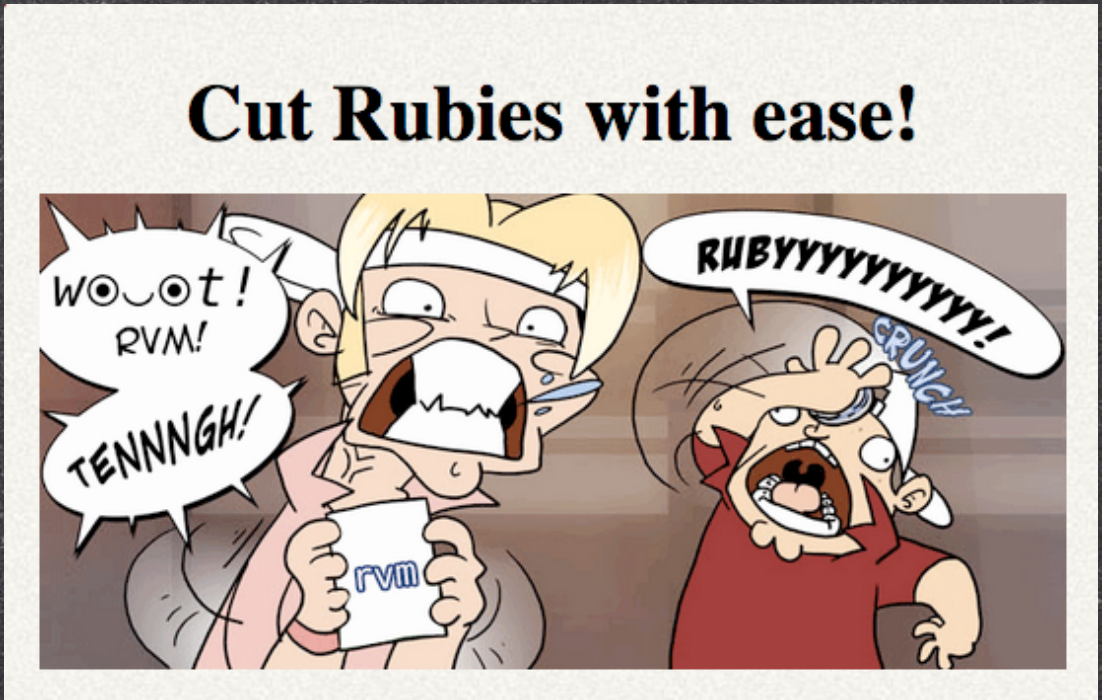
Secure

Cross platform

http://www.sourcetreeapp.com/

rvm

Ruby Version Manager



rvm

```
rvm install 1.9.3

rvm use 1.9.3

rvm list

rvm requirements 1.9.3

rvm gemset create 1.9.3@development

rvm use 1.9.3@development
```

rvm

```
rvm --default use 1.9.3
rvm use default
```

Basic Ruby

```
irb

ruby <file_name>.rb

Everything is an object

obj.class

obj.methods.sort

Auto converts data structures

"Hello " + "World"
```

Homework!

One thing from a Ruby tutorials

One thing from a git tutorial

Back to Ruby...

Adding with Strings

```
"Hello " + "World"

puts "Hello " + 7

"Hello " + 7.to_s

print "Hello " * 7
```

Flow Control

```
if <condition>
  <code>
else
  <else code>
end
```

More Flow Control

```
if <condition 1>
  <code>
elsif <condition 2>
  <else code>
elsif <condition 3>
  <else code>
end
```

Puts What's False?

```
if false
 puts "Hello"
end
```

Puts What's False?

```
if 0
  puts "Hello"
end
```

Puts What's False?

```
if nil
  puts "Hello"
end
```

???

```
false.class

nil.class

nil.class.class

0.class
```

Looping

```
loop{
 # put your code here
 puts "I'm loopy"
}
```

Looping

```
i = 0
loop do
 i += 1
 puts "I'm loopy * #{i}"
 break if i==10
end
```

Looping

```
i = 0
loop do
 i += 1
 print "I'm loopy * #{i} \n"
 break if i==10
end
```

Looping

```
i = 0
loop do
 i += 1
 next if i == 3
 puts "I'm loopy * #{i}"
 break if i==10
end
```

While We're Here...

```
i = 0
while i < 10
  i += 1
  puts "Loopy-er"
end
```

Until You Learn It

```
i = 0
until i < 10
  i += 1
  puts "Loopy-er"
end
```

For Whom Does the Bell Toll?

```
for i in 1..10
  puts "Toll..."
end
```

http://www.flickr.com/photos/kenanderson/2741418056

btw

what is 1..10 ?

```
(1..10).class
```

Looping

```
5.times do
 # put your code here
 puts "I'm loopy"
end
```

Iterators

```
<collection>.each{ | <variable> |
  # do something with <variable>
}
```

Iterators

```
(1..10).each{ |i|
  puts i
}
```

Iterators

```
my_list.each{ |i|
  puts i
}
```

Your Turn

```
Fizz Buzz

From 1 to 100

print "fizz" if divisible by 3

print "buzz" if divisible by 5

print "fizzbuzz" if div by 3 and 5
```

Further FizzBuzz Reading

http://rubyquiz.com/quiz126.html

Commonly Used Classes

- Array
- File
- Hash
- Regexp
- Symbol
- Time

Which Class Am I?

```
5.class

“”.class

“Hello”.class.class

[].class

{}.class

Object.class
```

What's a Class?

Has state

Has behavior

An object!

More on this later...

Classes

```
class ClassName

end
```

Methods

```
def method_name

end
```

Methods

```
def method_name
  # do stuff
  return x
end
```

Method's Return Value

```
def method_name
  # returns the last thing in the method
  x
end
```

Method's (Multiple) Return Values

```
def method_name
  # returns the last thing in the method
  return x, y, z
end
```

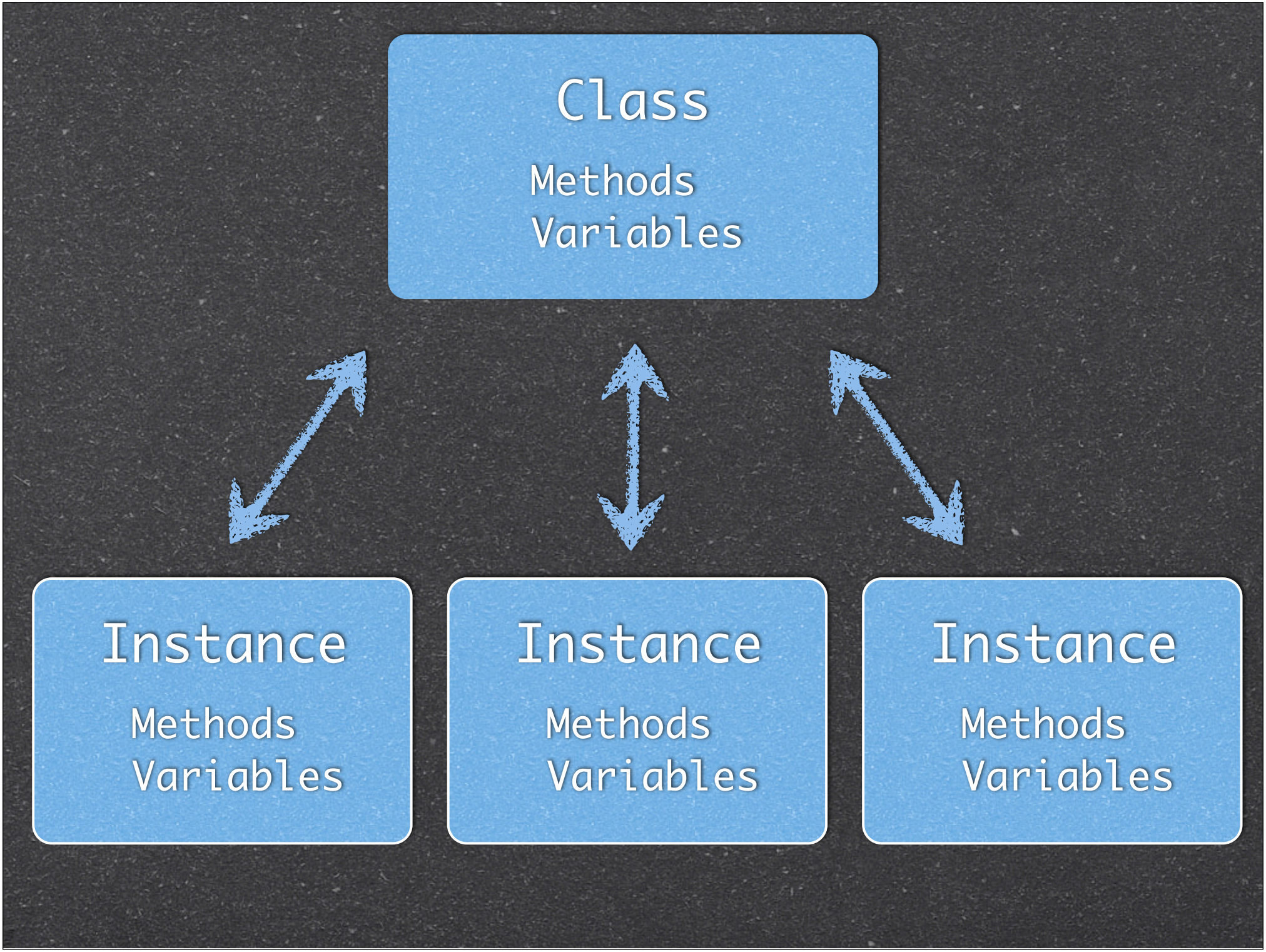
irb

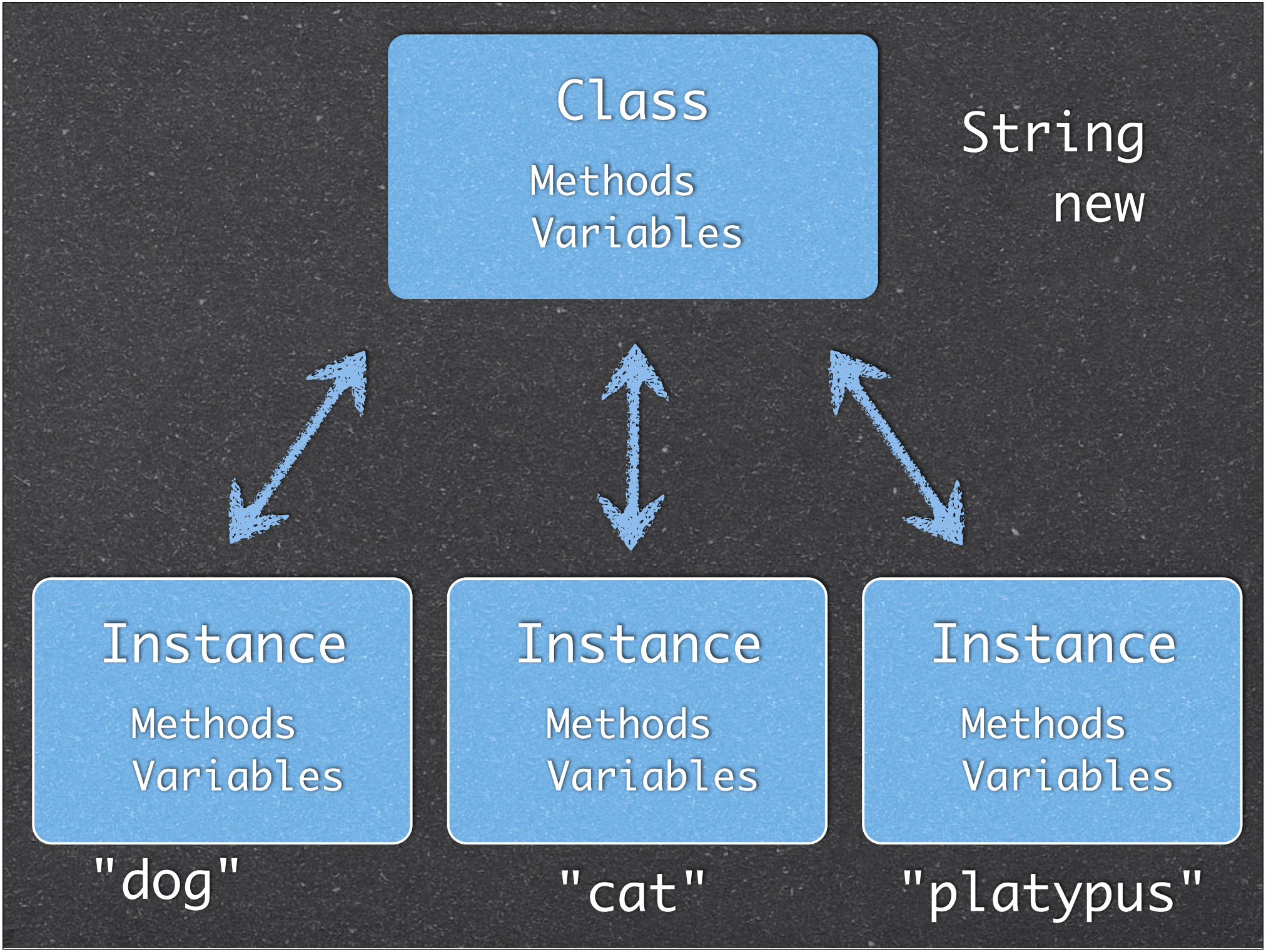
```
def m1
  a, b, c = 1, 2, 3
  return a, b, c
end
x, y, z = m1
x
y
z
```

.new

```
# when you call <class.new>
# this runs
def initialize
  # init stuff
end
```

Class vs Instance


Class
Methods
Variables
Instance
Methods
Variables
Instance
Methods
Variables
Instance
Methods
Variables


Class
Methods
Variables
String
new
Instance
Methods
Variables
Instance
Methods
Variables
Instance
Methods
Variables
"dog"
"cat"
"platypus"

Class Method or Variable

Shared among all like variables

Updates seen by all like variables

Variables indicated by @@

Methods called on ClassName

Math.exp

Instance Method or Variable

Seen by a single variable

Shared by all it's methods

Variables indicated by @

Methods called on a variable name

a_string.length

Example

```
class Test1
  def initialize
    @@cv1 = "class variable one"
    @iv1 = "instance variable one"
  end
  def print_info
    str = @@cv1 + ":"+ @iv1
    puts str
  end
end
```

Class Variable

```
class Test1
  def initialize
    @@cv1 = "class variable one"
    @iv1 = "instance variable one"
  end
  def print_info
    str = @@cv1 + ":"+ @iv1
    puts str
  end
end
```

```
class Test1
  def initialize
    @@cv1 = "class variable one"
    @iv1 = "instance variable one"
  end
  def print_info
    str = @@cv1 + ":"+ @iv1
    puts str
  end
end
```

Instance Variable

```
class Test1
  def initialize
    @@cv1 = "class variable one"
    @iv1 = "instance variable one"
  end
  def print_info
    str =          + ":"+ @iv1
    puts str
  end
end
```

Local Variable

irb

```
class Test1
  def initialize
    @@cv1 = "class variable one"
    @iv1 = "instance variable one"
  end
  def print_info
    str = @@cv1 + ":"+ @iv1
    puts str
  end
end
```

Usage

```
t = Test1.new

t.print_info
```

```ruby
class Test

 def self.hello
  puts "A class method"
 end

 def world
  puts "An instance method"
 end

end
```

```ruby
t = Test.new

# Good

Test.hello

t.world

# Bad

t.hello

Test.world

# Info

Test.methods.sort

Test.instance_methods
```

```
class Test


 def self.hello
  puts "A class method"
 end


 def hello
  puts "An instance method"
 end
end
```

```
t = Test.new


Test.hello

t.hello
```

Class or Instance Method?

```
Math.sqrt(9)

str = String.new("Hello World!")

str.reverse!
```

Classes

```
class Learning

end
```

Classes

```
class Learning
  def self.class_method
    puts "This is a class method"
  end
end
```

Classes

```
class Learning
  def self.class_method
    puts "This is a class method"
  end
  def instance_method
    puts "This is an instance method"
  end
end
```

Classes

```
class Animal

end
```

Classes

```
class Animal
  def speak
    puts “whut?”
  end
end
```

Dogs

```
class Dog < Animal
  def speak
    puts “bow wow”
  end
end
```

Use Them

```
a = Animals.new
d = Dog.new
a.speak()
d.speak()
```

What Did We Do?

Dog inherited from Animal

Dog overrode speak

Method Arguments

```
class Animal

end
```

Classes

```
class Animal
  def initialize( speech )
    @speech = speech
  end
end
```

Classes in Files

One class per file is BETTER (not required)

animal.rb

dog.rb

```
require "./animal"

require 'socket'
```

Variables

```
Matz is from Japan

good_ruby_convention

BadRubyConvention
```

Class vs Instance

in scope variable

var_name

instance variable

@var_name

class variable

@@var_name

Method Arguments

```
class Animal

end
```

Classes

```
class Animal
  def initialize( speech )
    @speech = speech
  end
end
```

Dogs

```
class Dog < Animal
  def speak
    puts @speech
  end
end
```

Use Them

```
my_dog = Dog.new("woof")

your_dog = Dog.new("bow wow")

my_dog.speak()

your_dog.speak()
```

What Did We Do?

Dog inherited it's constructor

Animal set an instance variable

Each dog printed it's own instance variable

Open Classes

Classes can be “opened” and extended

aka “Monkey Patched”

Re-open Animals

```
class Animals
  def speak
    puts “I can’t speak. I'm an animal!”
  end
end
```

Now Use It

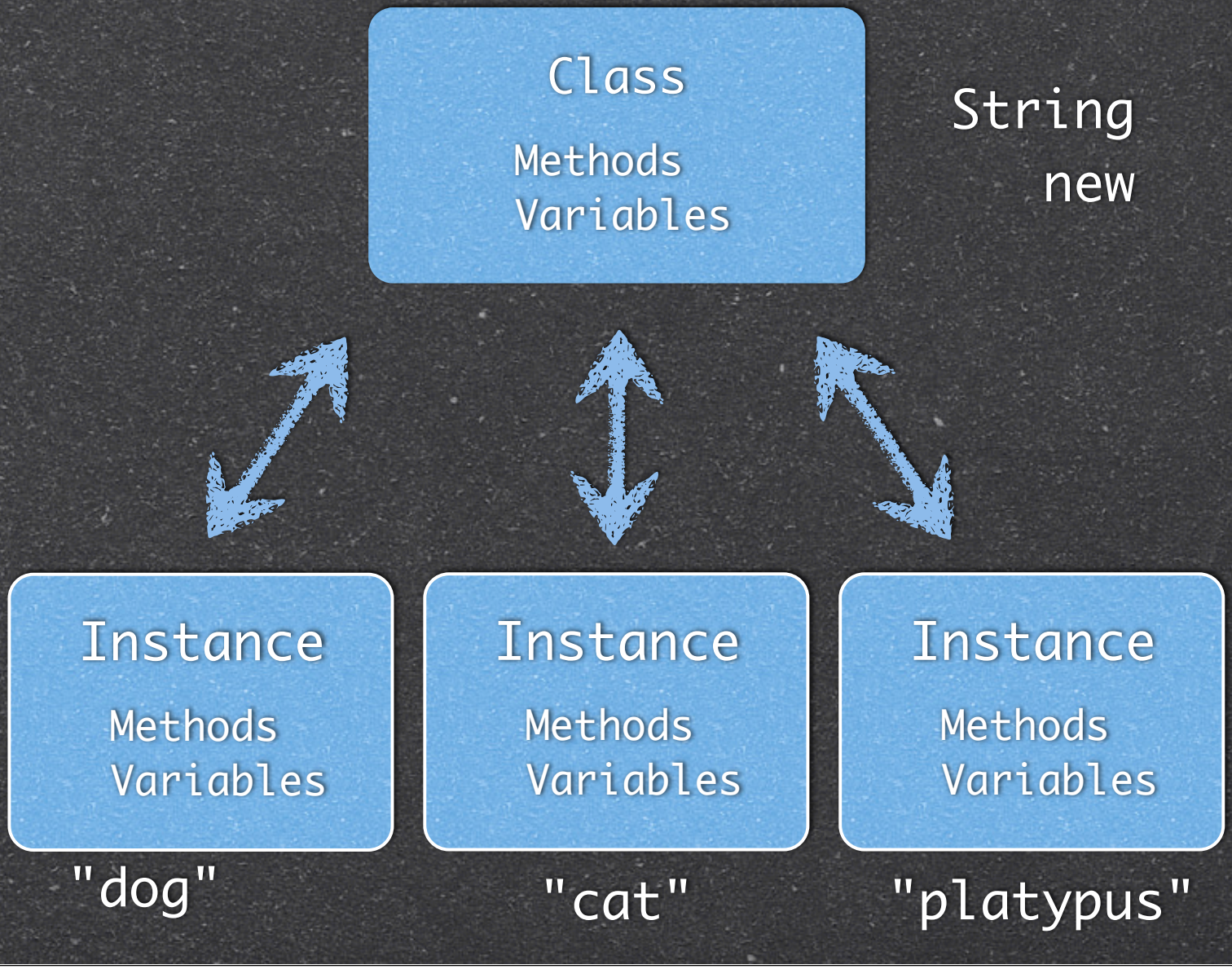
```
a.speak();

a.speak()

a.speak
```

How Did That Work?

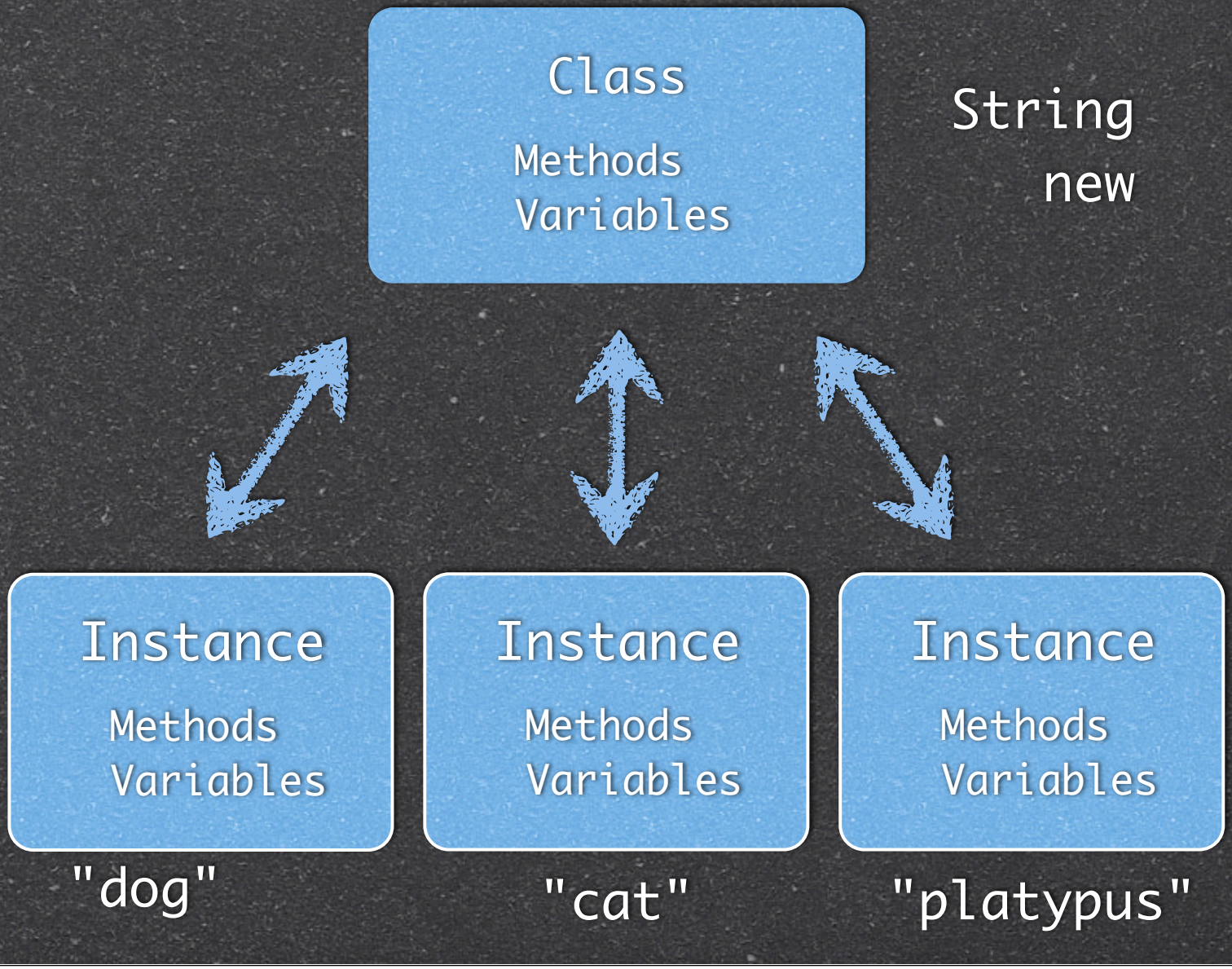
Instance methods are "owned" by the class

Pointed to by the instance

Remember this diagram?

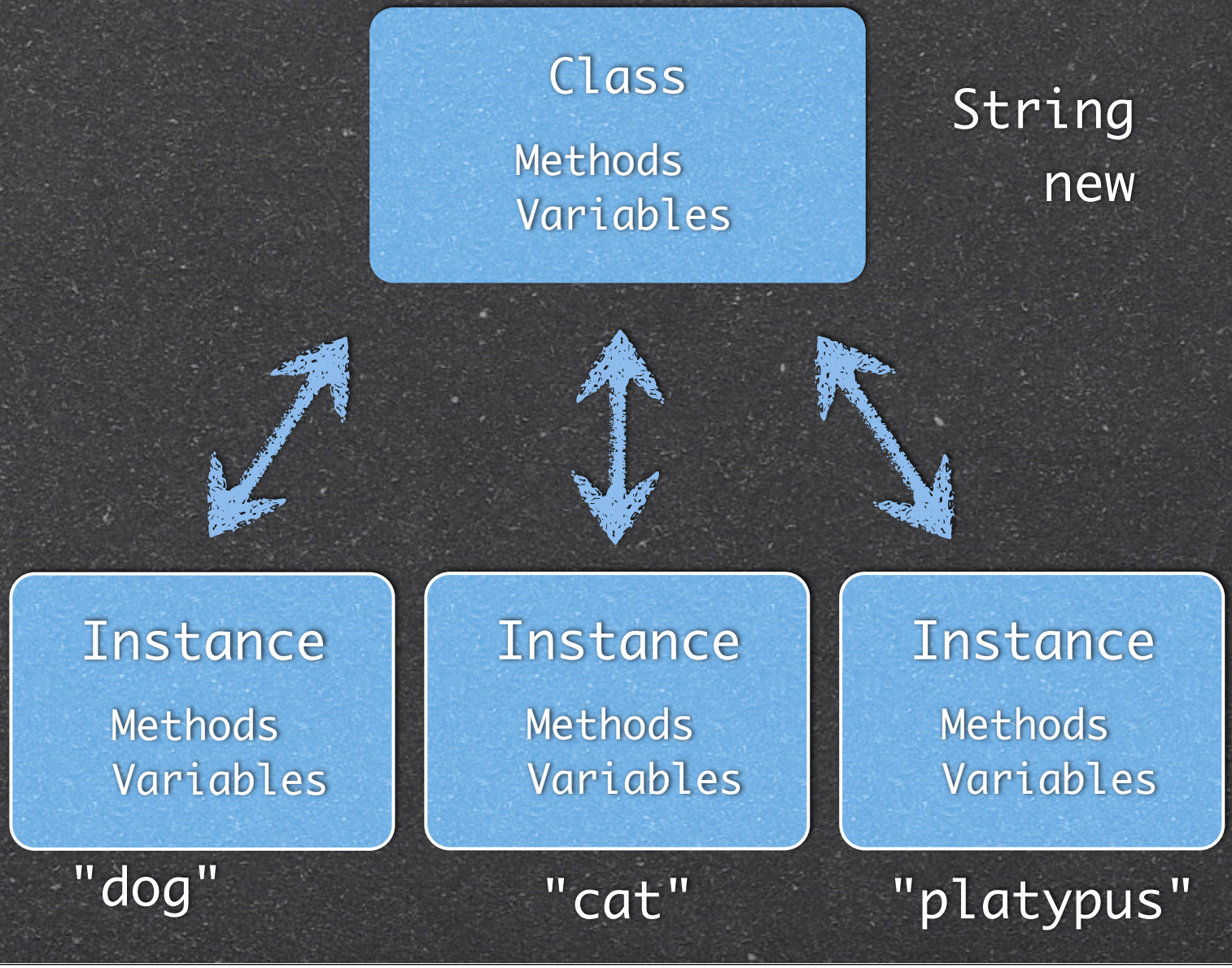


Conceptually accurate



Conceptually accurate

Actually wrong



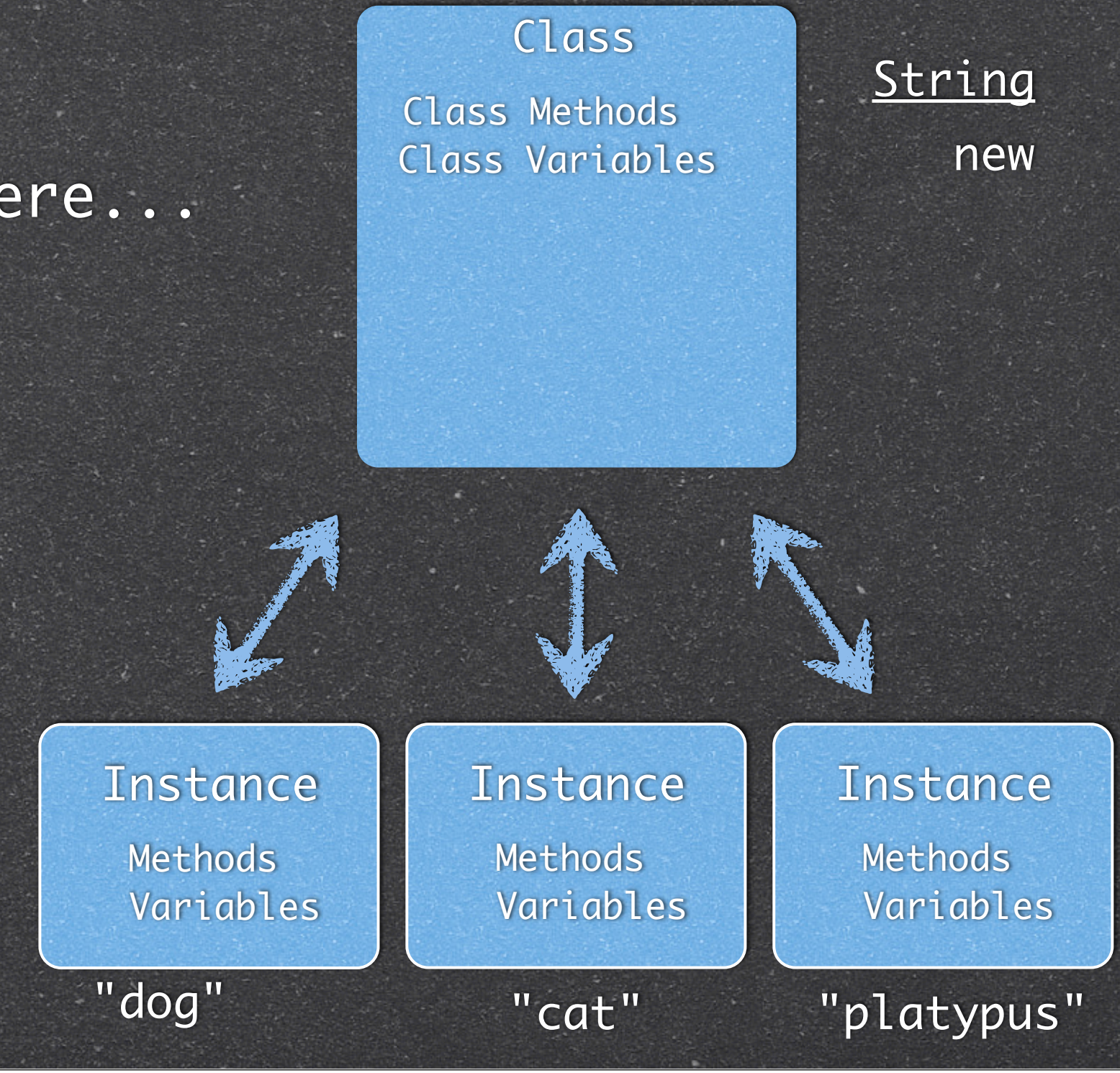
It's All About Class

Class owns instance methods

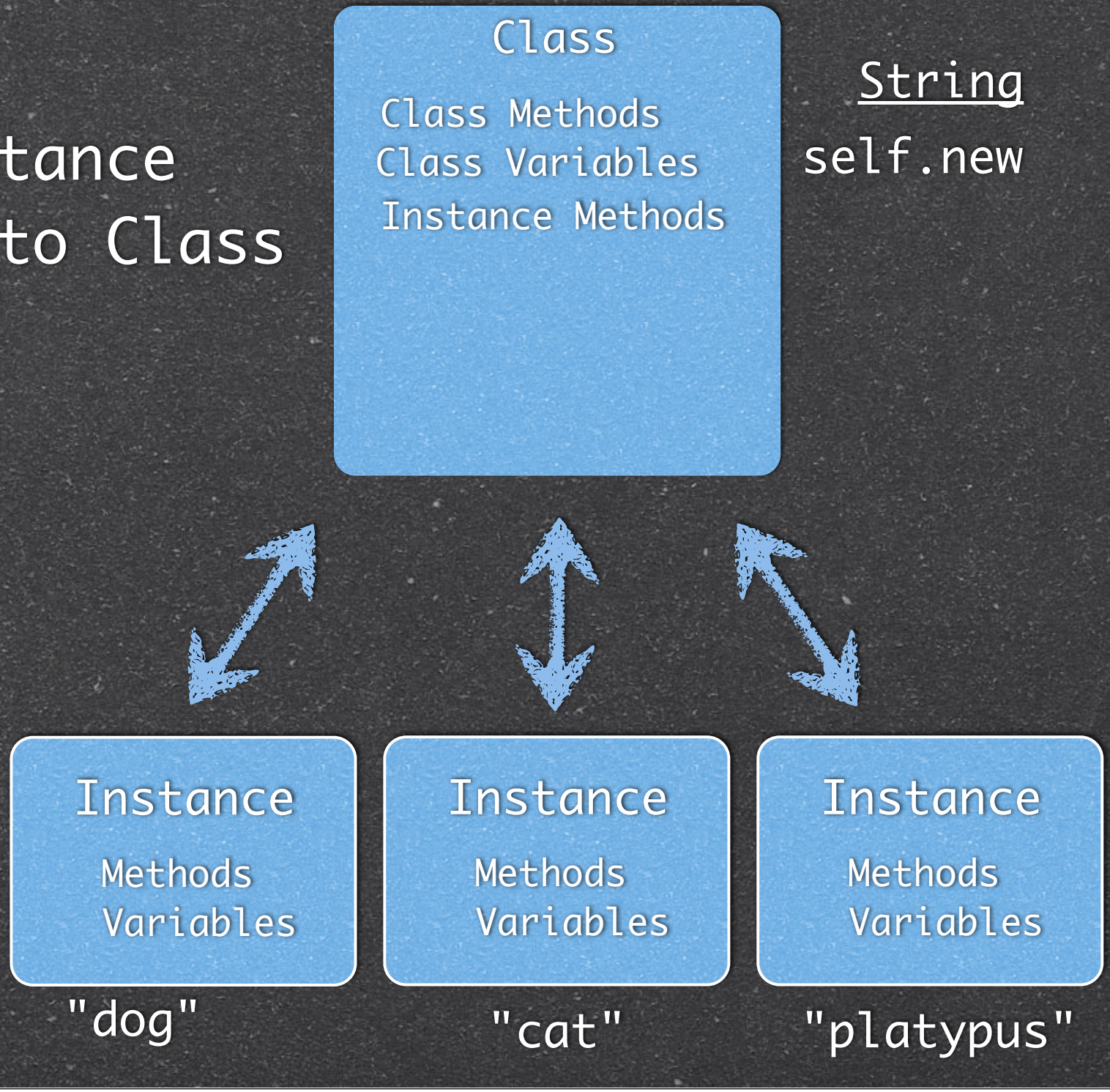
Instances point to shared methods...

Stored in the class

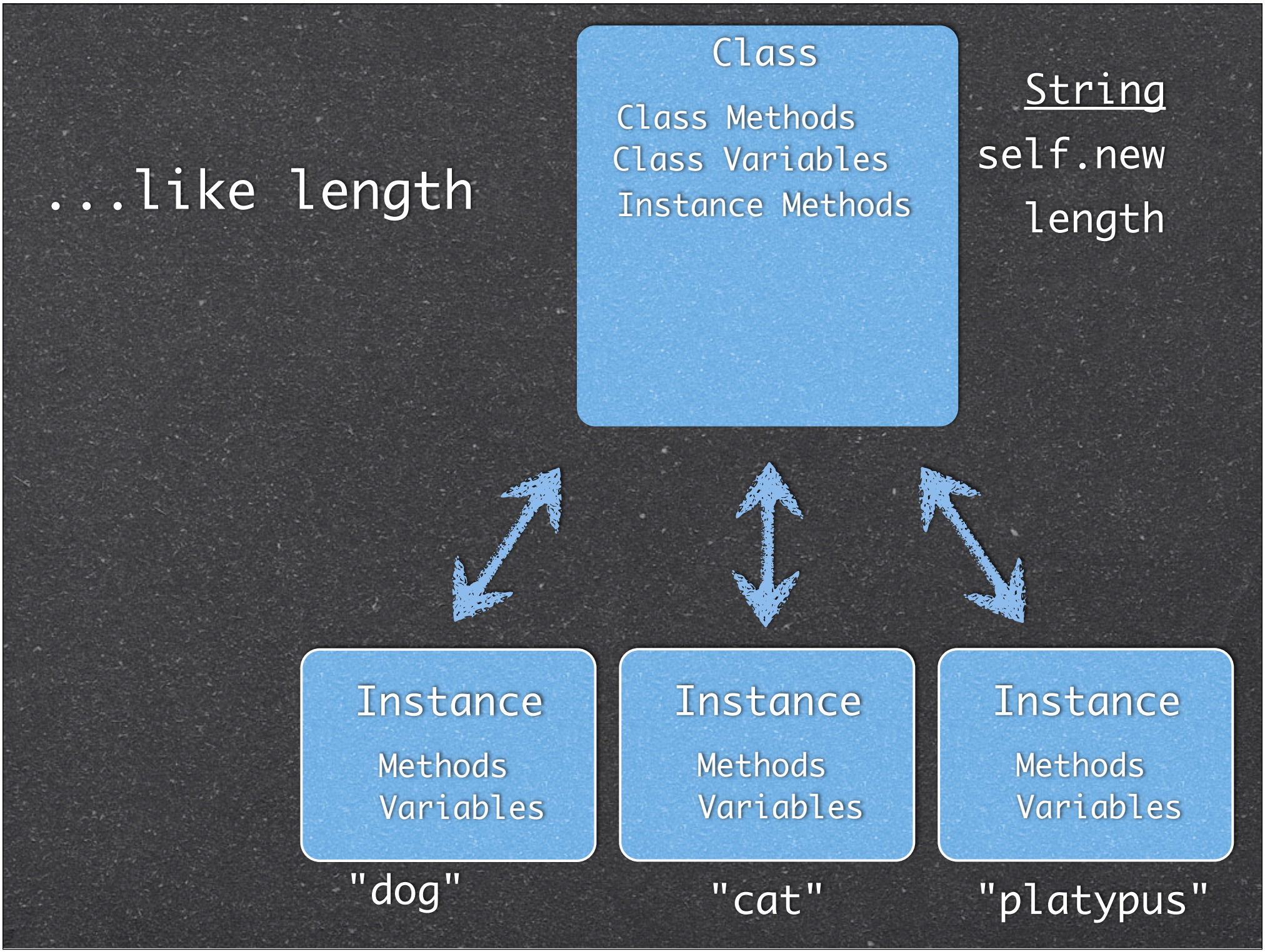
Start here...


Class
Class Methods
Class Variables
String
new
Instance
Methods
Variables
Instance
Methods
Variables
Instance
Methods
Variables
"dog"
"cat"
"platypus"

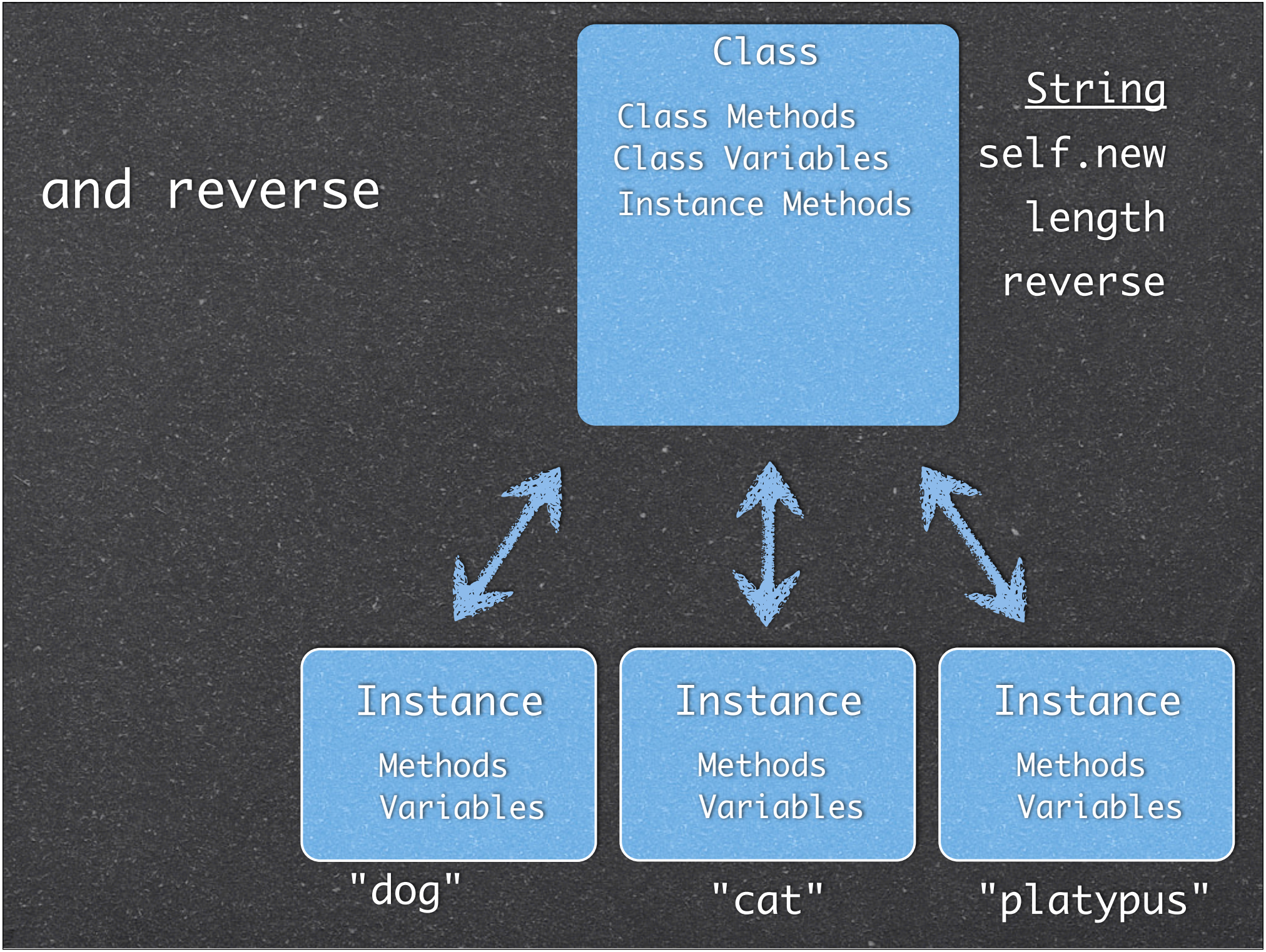
Move instance methods to Class



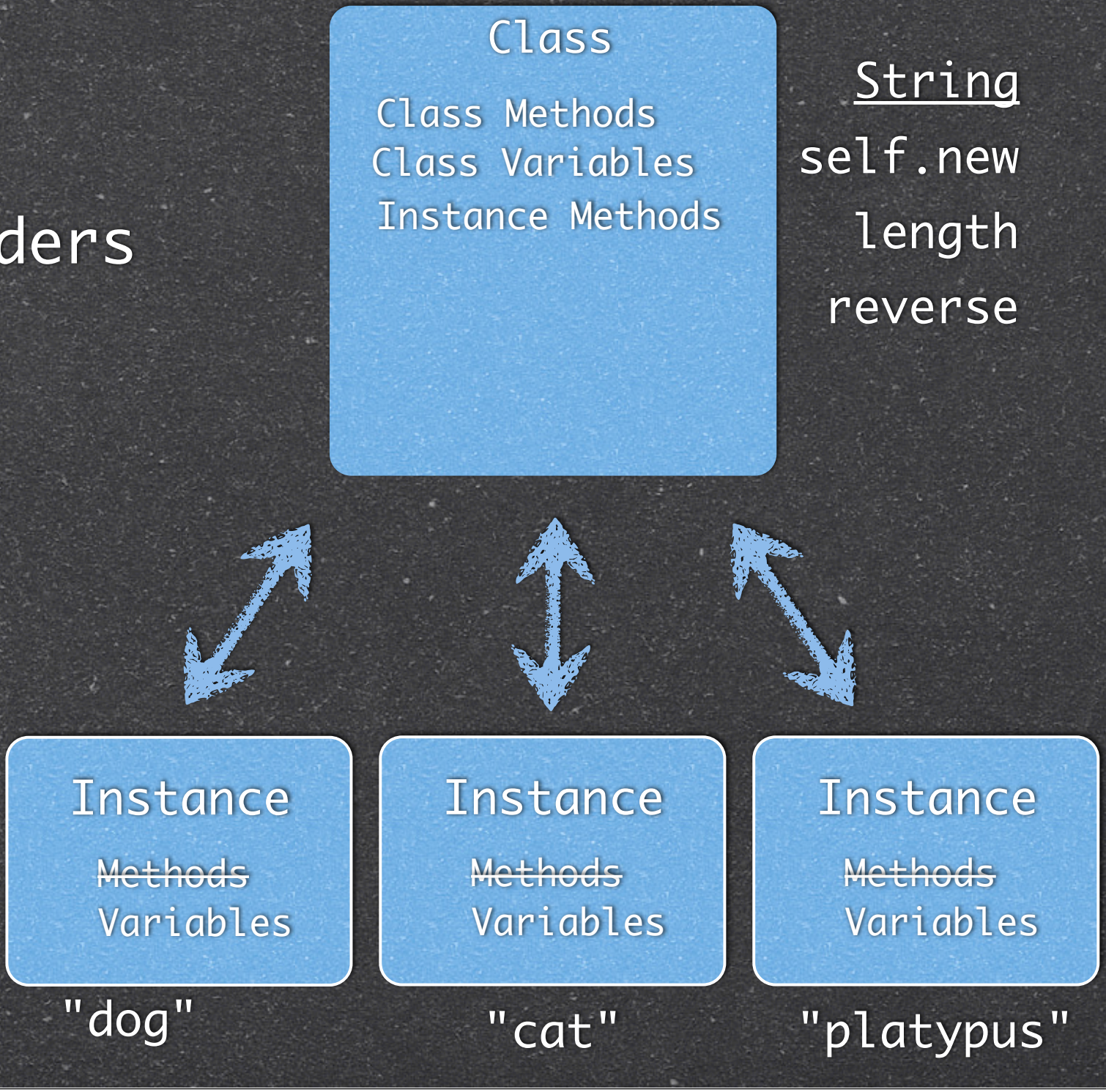
...like length

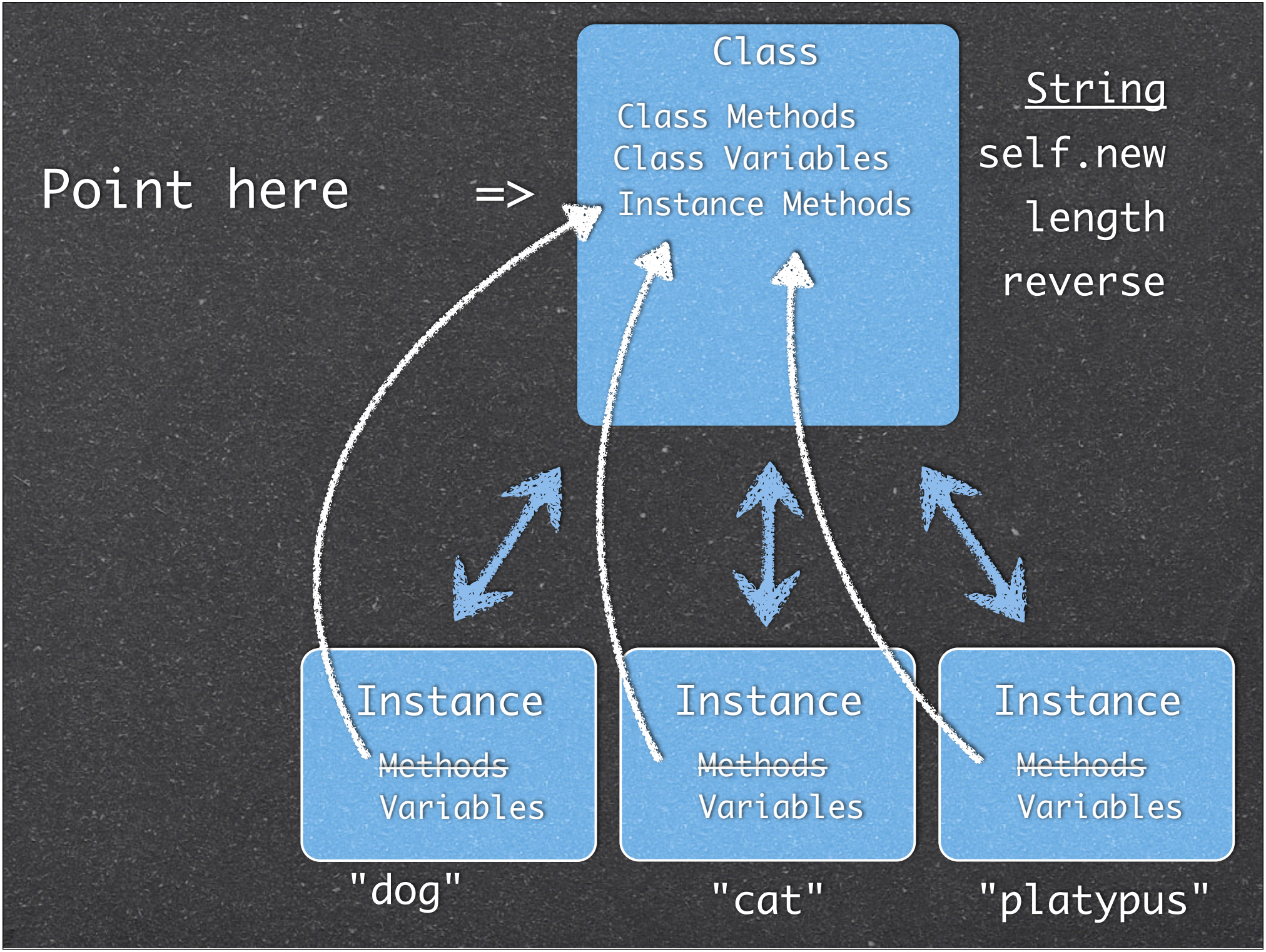

Class
Class Methods
Class Variables
Instance Methods
String
self.new
length
Instance
Methods
Variables
Instance
Methods
Variables
Instance
Methods
Variables
"dog"
"cat"
"platypus"

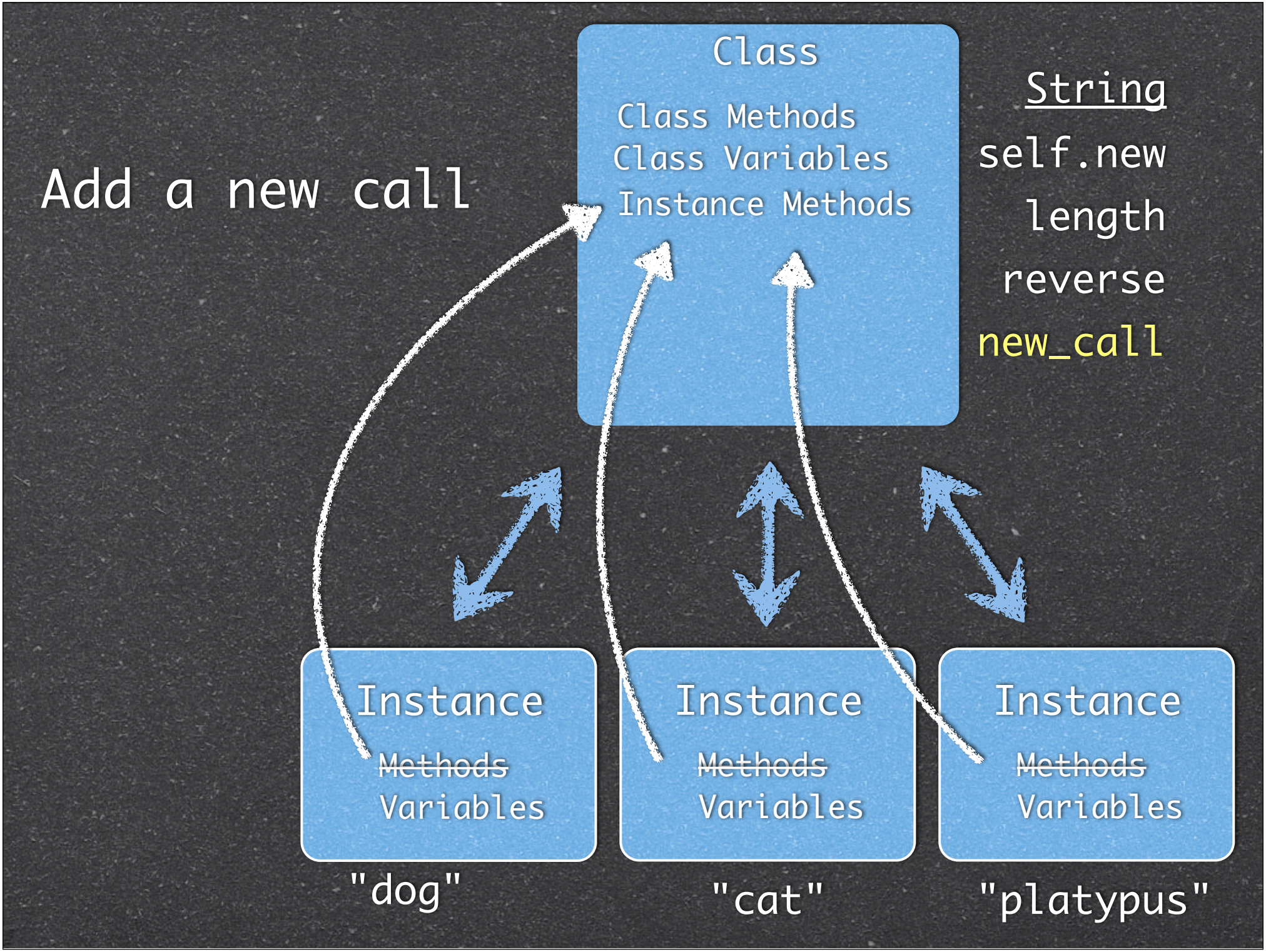
and reverse


Class
Class Methods
Class Variables
Instance Methods
String
self.new
length
reverse
Instance
Methods
Variables
Instance
Methods
Variables
Instance
Methods
Variables
"dog"
"cat"
"platypus"

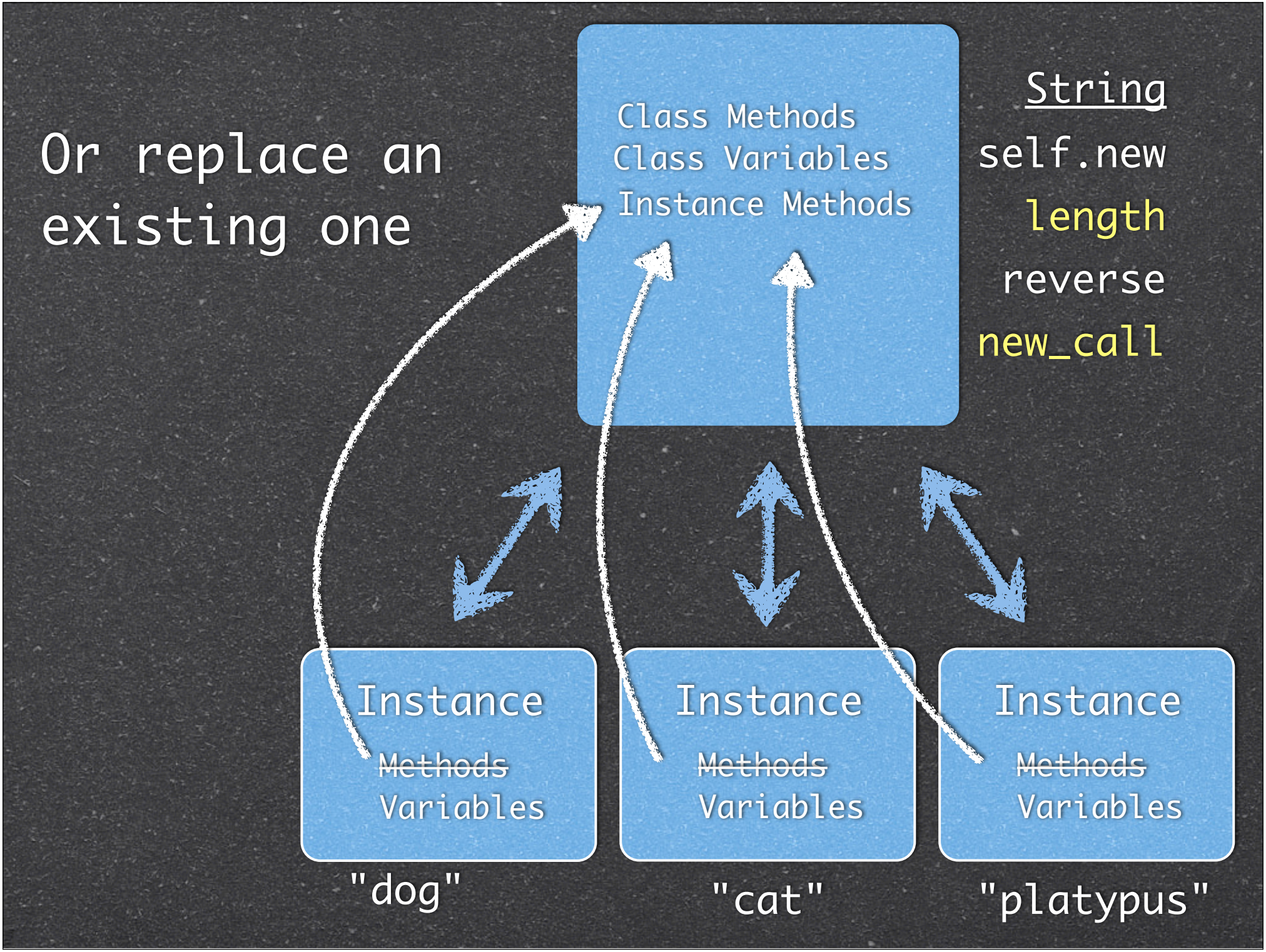
remove placeholders

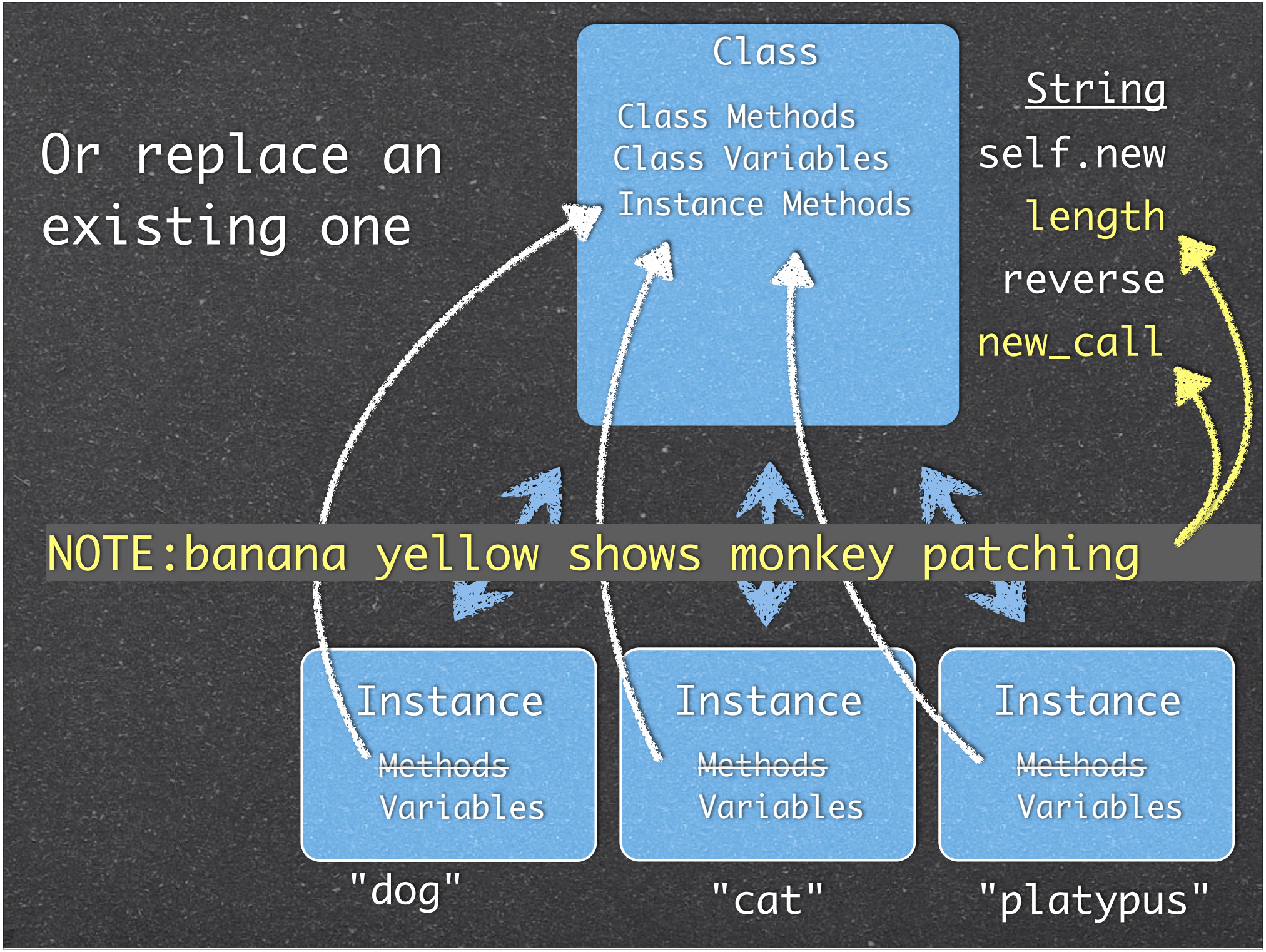



Point here
=>
Class
Class Methods
Class Variables
Instance Methods
String
self.new
length
reverse
Instance
~~Methods~~
Variables
Instance
~~Methods~~
Variables
Instance
~~Methods~~
Variables
"dog"
"cat"
"platypus"


Add a new call
Class
Class Methods
Class Variables
Instance Methods
String
self.new
length
reverse
new_call
Instance
~~Methods~~
Variables
Instance
~~Methods~~
Variables
Instance
~~Methods~~
Variables
"dog"
"cat"
"platypus"

Or replace an existing one




Or replace an existing one
Class
Class Methods
Class Variables
Instance Methods
String
self.new
length
reverse
new_call
NOTE:banana yellow shows monkey patching
Instance
Methods
Variables
Instance
Methods
Variables
Instance
Methods
Variables
"dog"
"cat"
"platypus"

And Bad Programmers

Monkey patch core classes

Like String

Or 3rd party libraries

Frequently!

Use moderation

Your Turn

Open String

add a new method to aLtErNaTe up/lower case

"string".up_down => sTrInG

Overwrite .reverse to duplicate .upcase

Setters/Getters

```
attr_accessor

attr_reader

attr_writer
```

http://www.flickr.com/photos/schauecker/5323011023/

attr_reader

```
class Soldier
 attr_accessor :name, :rank, :serial_num
end
```

Your Turn

Use the attr_* calls

attr_accessor

attr_reader

attr_writer

What calls does each generate?

http://www.flickr.com/photos/schauecker/5323011023/

Passing Arguments to Methods

```
t = Test.new

t.name= “Jared”

t.name=(“Jared”)

t.name=(“Jared”);
```

Your Turn

What’s the relationship of...

Automobile to Ford?

Common "methods"?

Your Turn (again!)

Base class is Shape

Write area & perimeter

Write rectangle & triangle

Both subclass Shape

Overwrite area & perimeter

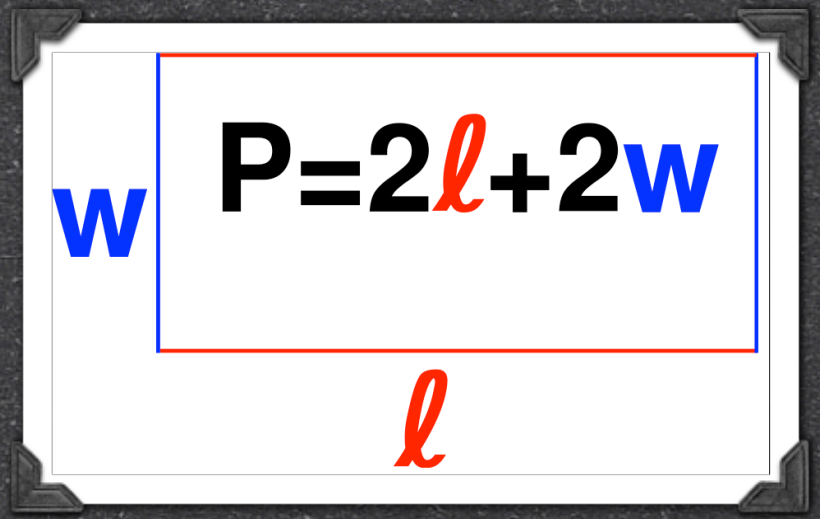
What other methods/variables are needed?

Area & Circumference

Rectangle

A = l*w

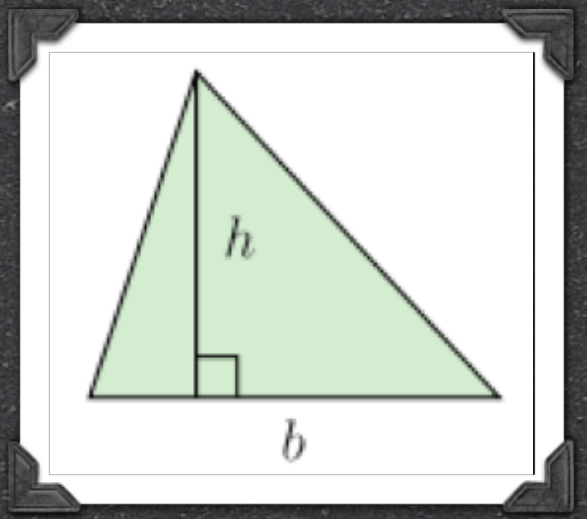
P = 2*l + 2*w



Triangle

A = 1/2 b * h

P = a + b + c



http://en.wikipedia.org/wiki/File:Triangle.GeometryArea.svg

Next...

Ruby Part 2

Common classes and usage