Ruby 101

Jason Dew

Columbia Ruby Brigade

5/1/08

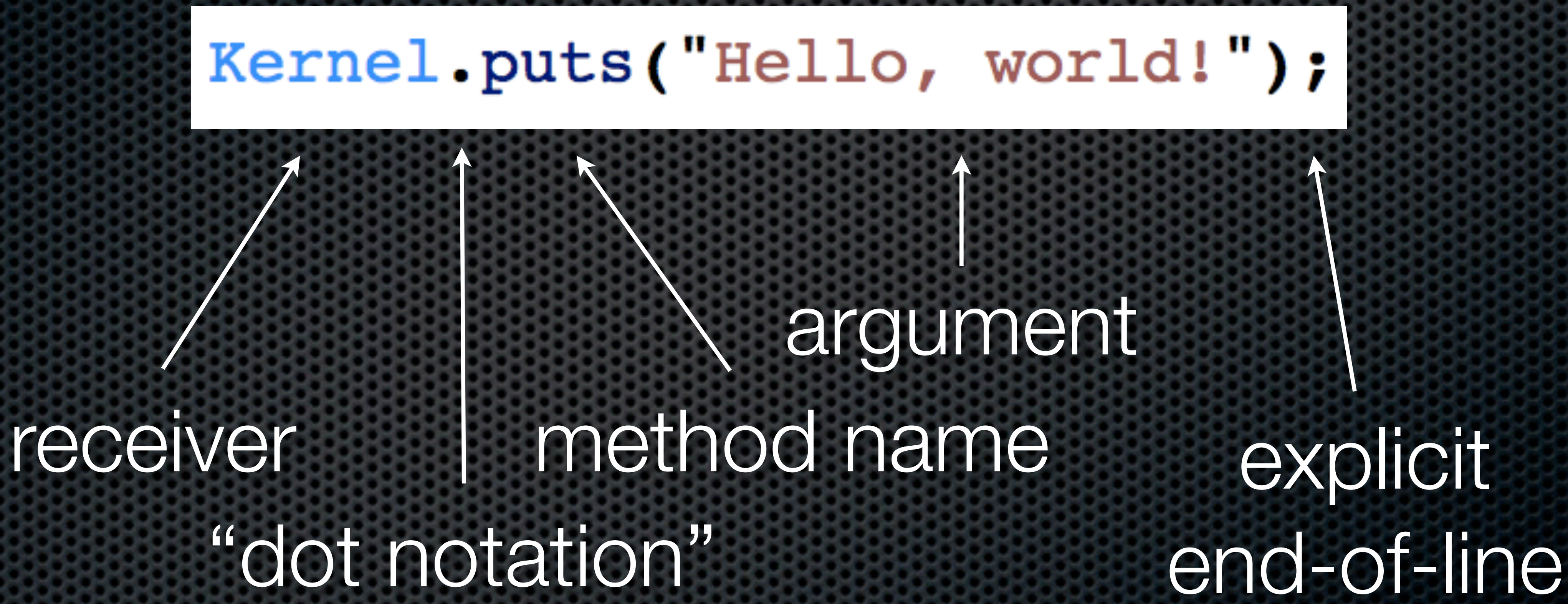
Language Overview

- interpreted
- purely object-oriented
- expression-oriented
- dynamic
- agile

OOP Primer

- **Object** - similar to a variable, any thing in Ruby
- **Class** - defines attributes and methods
- **Instance** - an object instantiated from a class
- **Method** - something an object can “do”
- **Attribute** - a characteristic of an instance

Anatomy of a Method Call



```ruby
Kernel.puts("Hello, world!");
```

Typical Invocation

```ruby
puts "Hello, world!"
```

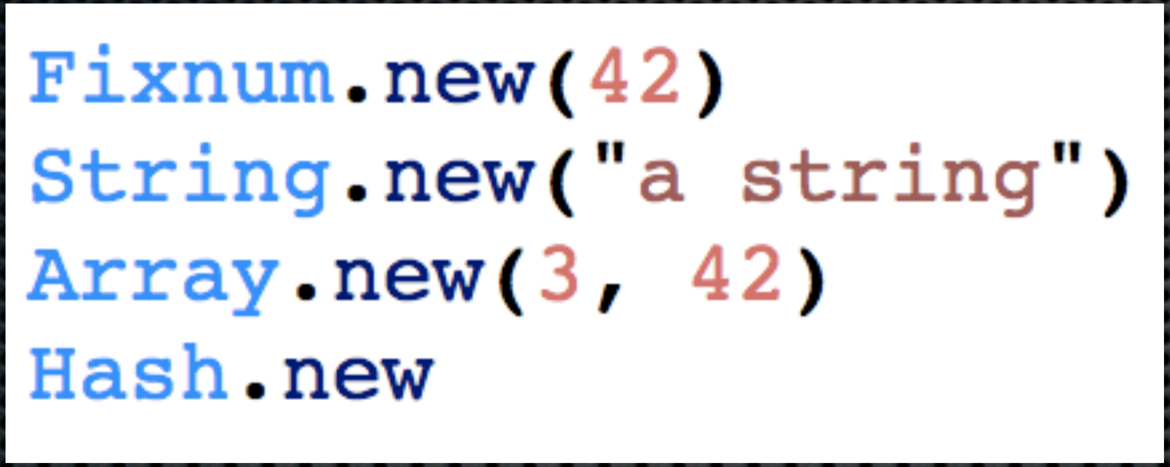
- implicit receiver
- implicit end-of-line
- no parentheses needed (in this case)
- this is more idiomatic

Commonly Used Classes

- Fixnum - the integers
- String - a series of characters
- Symbol - a lightweight string-like object, just a name
- Array - a series of objects
- Hash - a series of key/value pairs, like a dictionary

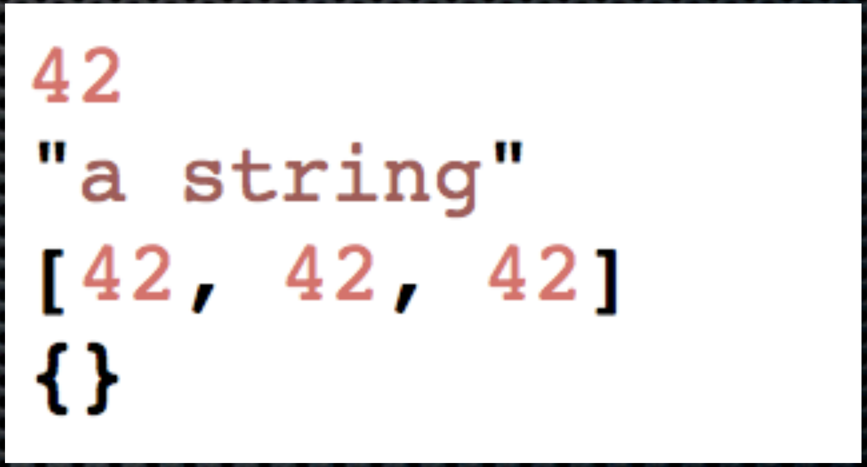
Instantiating Objects

The long way



```
Fixnum.new(42)
String.new("a string")
Array.new(3, 42)
Hash.new
```

The short way



```
42
"a string"
[42, 42, 42]
{}
```

Data Types

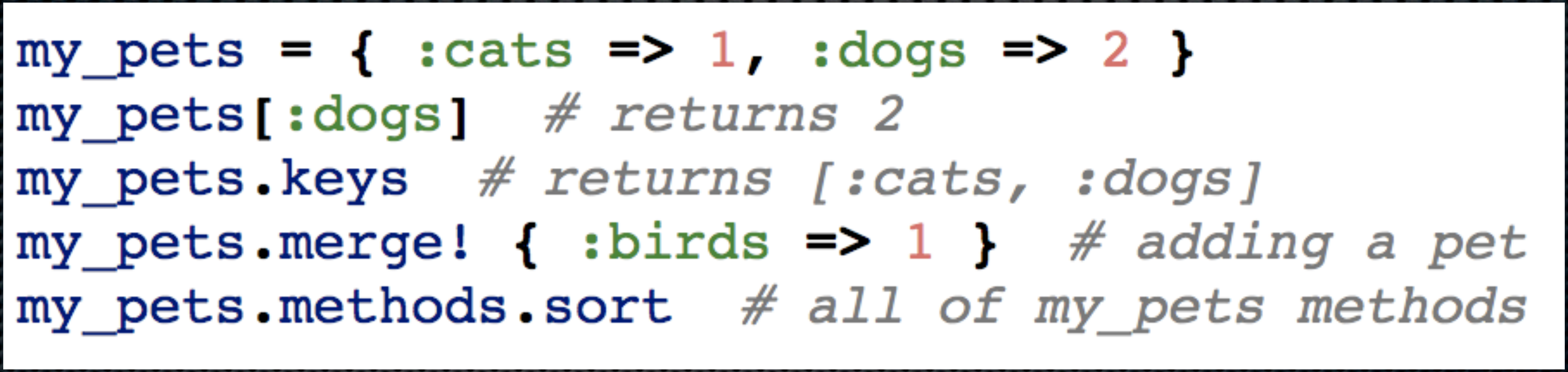
- Remember that everything is an object
- You can always ask an object for its class

`42.class` $\longrightarrow$ returns Fixnum

`"string".class` $\longrightarrow$ returns String

`:symbol.class` $\longrightarrow$ returns Symbol

Using Objects



```
my_pets = { :cats => 1, :dogs => 2 }
my_pets[:dogs]  # returns 2
my_pets.keys  # returns [:cats, :dogs]
my_pets.merge! { :birds => 1 }  # adding a pet
my_pets.methods.sort  # all of my_pets methods
```

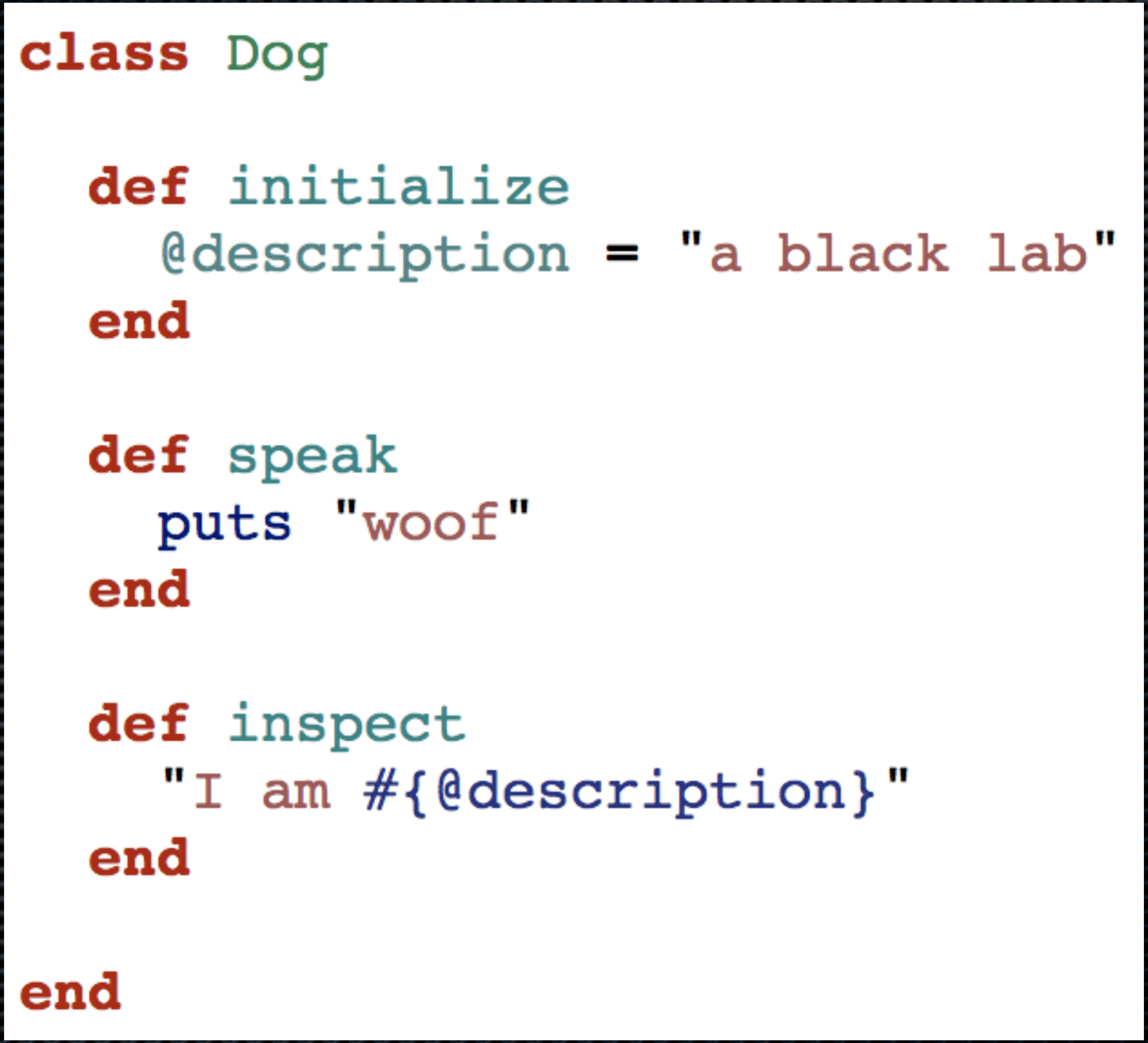
Rolling our own class

class name

class definition

```ruby
class Dog

  def initialize
    @description = "a black lab"
  end

  def speak
    puts "woof"
  end

  def inspect
    "I am #{@description}"
  end

end
```

Rolling our own class



```
class Dog

  def initialize
    @description = "a black lab"
  end

  def speak
    puts "woof"
  end

  def inspect
    "I am #{@description}"
  end

end
```

attribute

methods

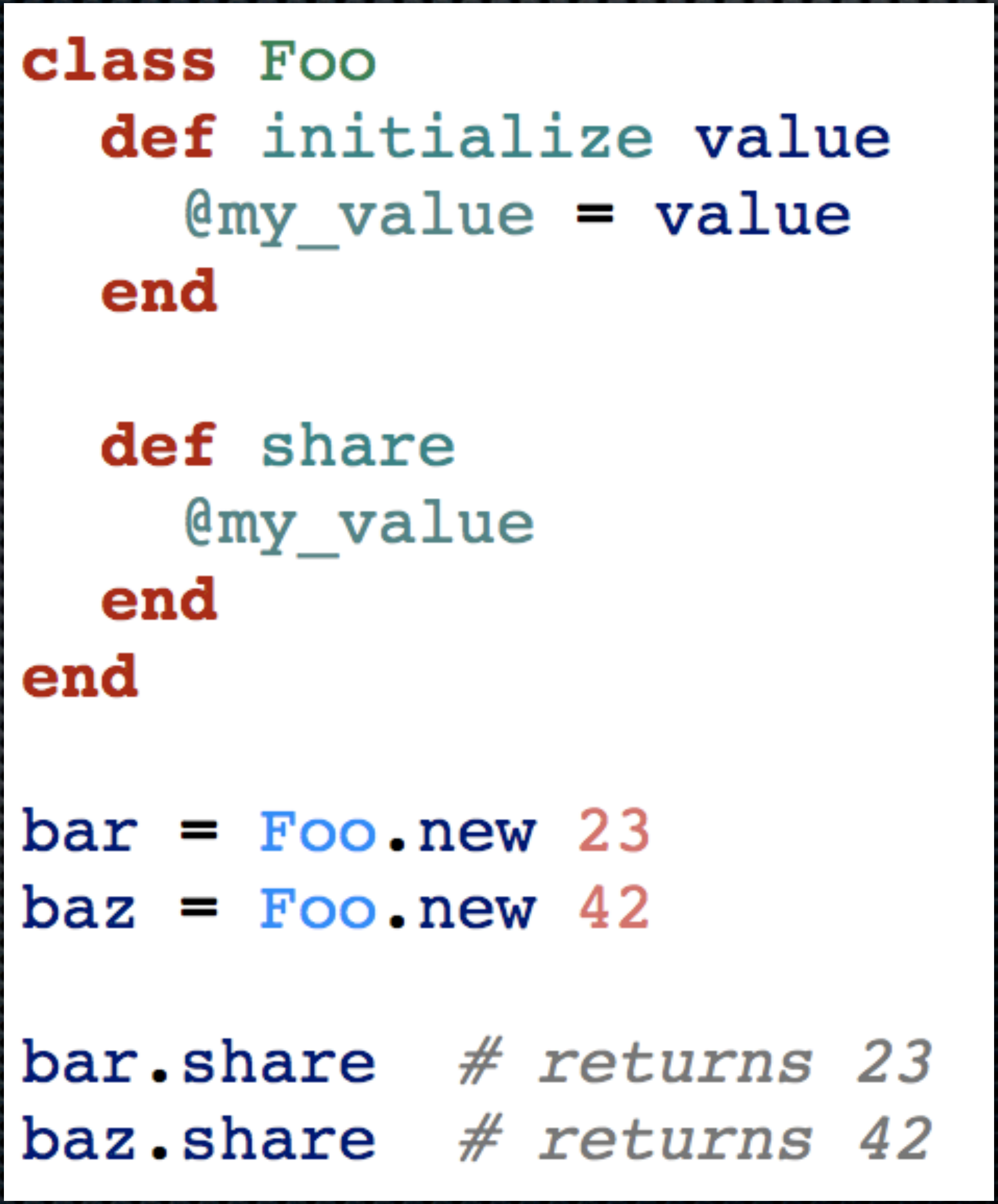
Tying it all together

```ruby
loki = Dog.new
loki.speak   # writes "woof" to the screen
loki.inspect   # returns "I am a black lab"
```

Variable Types

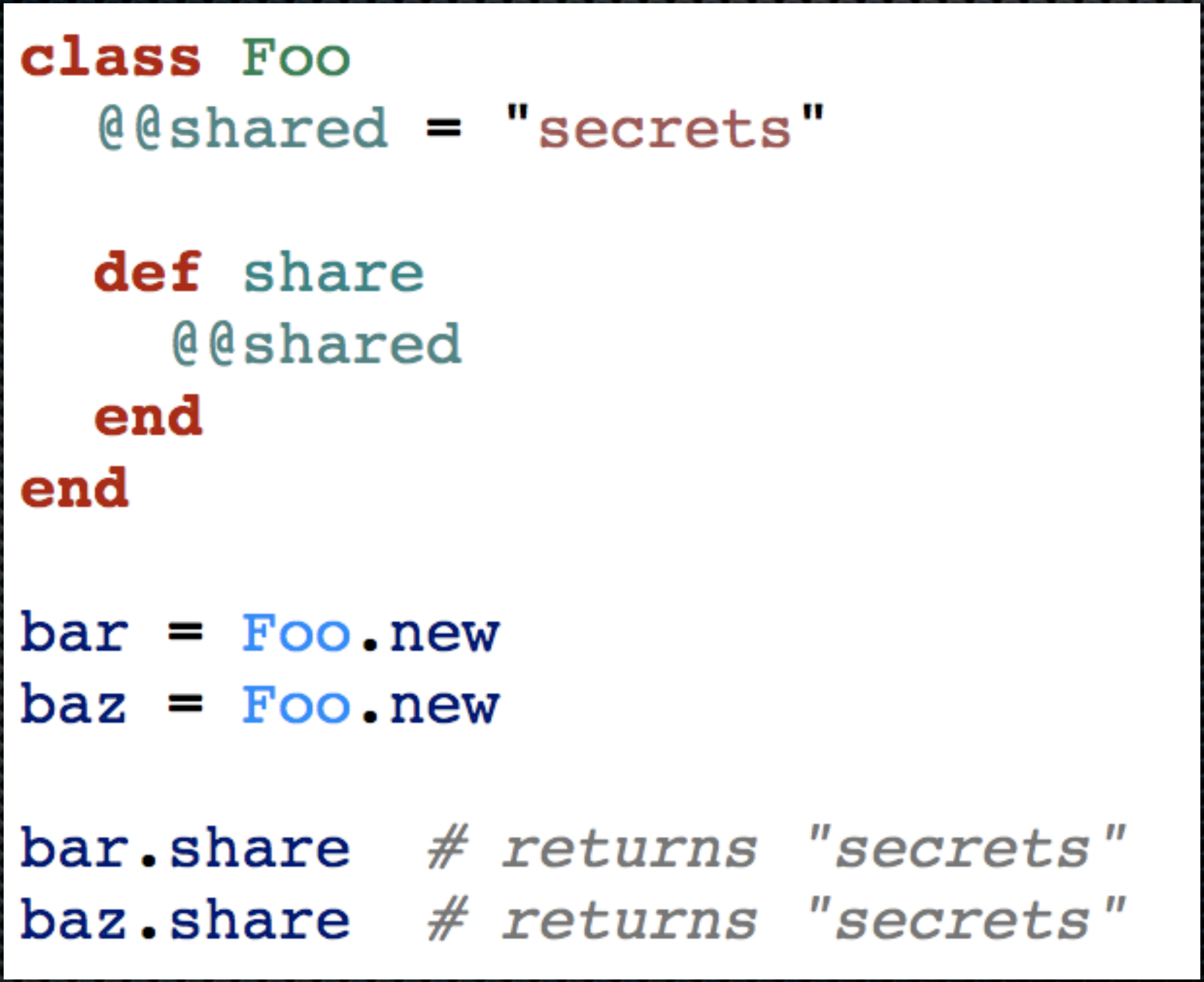
```
$global = "rarely used"

class Foo
  CONSTANT = "notice the case"
  @@class_variable = "shared by instances"

  def initialize
    @instance_variable = "not shared"
    local_variable = "only in this method"
  end
end
```

Instance Variables



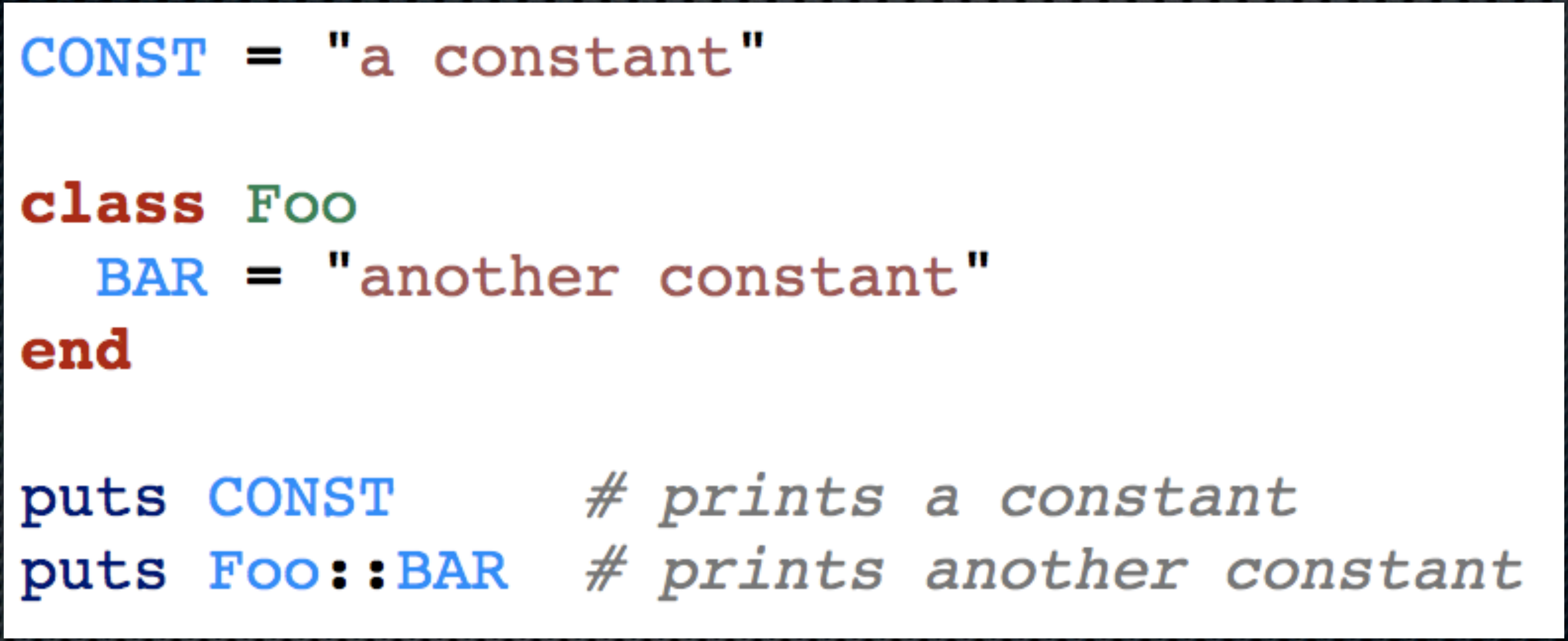
```
class Foo
  def initialize value
    @my_value = value
  end

  def share
    @my_value
  end
end

bar = Foo.new 23
baz = Foo.new 42

bar.share  # returns 23
baz.share  # returns 42
```

Class Variables



```
class Foo
  @@shared = "secrets"

  def share
    @@shared
  end
end

bar = Foo.new
baz = Foo.new

bar.share  # returns "secrets"
baz.share  # returns "secrets"
```

Constants



```ruby
CONST = "a constant"

class Foo
  BAR = "another constant"
end

puts CONST        # prints a constant
puts Foo::BAR     # prints another constant
```

Modules

- Like a class except that it only contains methods
- Cannot be instantiated
- Can be nested inside classes or other modules

Module Example

```ruby
module MyModule
  A_CONSTANT = 42

  class Dog
    ...
  end

  class Foo
    ...
  end
end

puts MyModule::A_CONSTANT # prints 42
MyModule::Dog.new  # creates an instance
```

Questions?

Questions?

- The Ruby homepage is at http://ruby-lang.org/
- Lots of information on the web, Google is your friend
- I'll continue this talk next meeting (if desired)
- Next meeting is June 5th, 11:30 am, same place