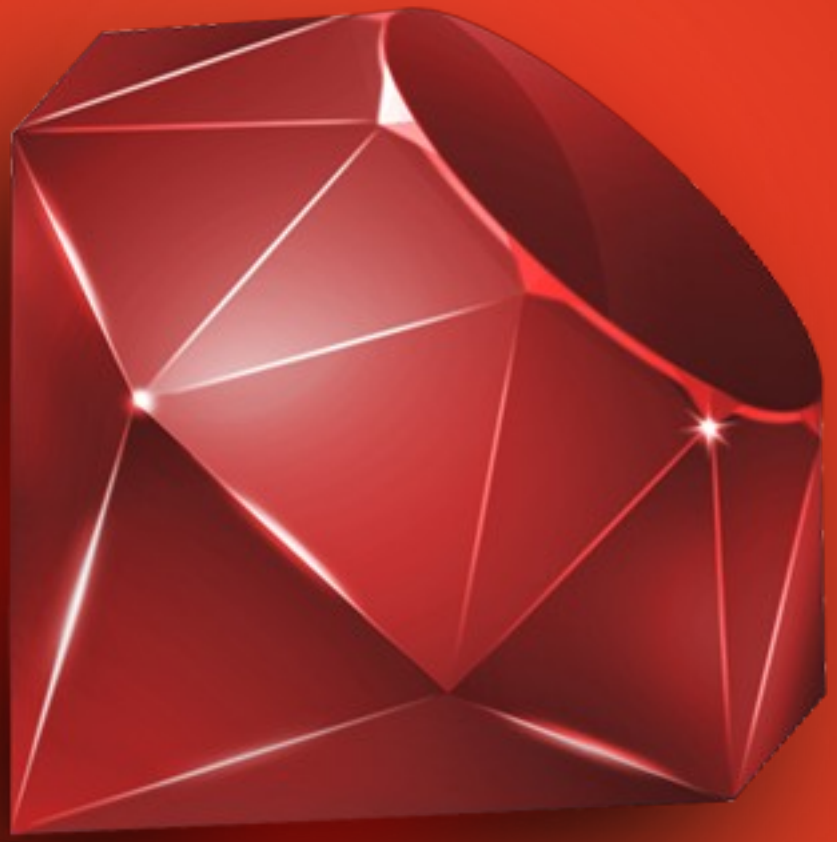
Ruby Programming



Classes

Overview

- Classes/Objects
- Access Control
- Inheritance
- Conventions
- Using other classes in other gems

Why study Classes?

- Ruby can perform in a functional manner
- Good programming practice
- Write Elegant Code

KISS

- Keep It Simple Stupid

DRY

- Don't Repeat Yourself

Class

- An object/entity representing some “noun” or thing
  - Properties/attributes
  - Methods/actions
  - Can inherit other classes
- Dog
  - Name
  - Breed
  - Color
  - Bark
  - work

Constructors

- Initializer methods that are called when creating a new instance of a class
- Instance variables – own distinct set of variables for an object instance (starts with @) and can only be used within the class

```
class Dog
	def initialize(name, breed)
		@name = name
		@breed = breed
	end
end

dog = Dog.new(“Bomber”, “Labrador”)
```

to_s

- A special method that will allow you to override the string representation of a class instance

```
class Dog
	def initialize(name, breed)
		@name = name
		@breed = breed
	end

	def to_s
		“#{@name}”
	end
end

dog = Dog.new(“Bomber”, “Labrador”)
puts dog
```

Attributes

- Attributes are properties of a class/object
- Accessed through methods

```
class Dog
	def initialize(name, breed)
		@name = name
		@breed = breed
	end

	def name
		@name
	end

	def to_s
		“#{@name}”
	end
end

dog = Dog.new(“Bomber”, “Labrador”)
puts dog.name
```

attr_reader

- Use a symbol to create a readable instance variable of the same name

```
class Dog

	attr_reader :name, :breed

	def initialize(name, breed)
		@name = name
		@breed = breed
	end

	def to_s
		“#{@name}”
	end
end

dog = Dog.new(“Bomber”, “Labrador”)
puts dog.name
```

Writable Attributes

- Methods ending with “=” and accepts arguments

```
class Dog

    attr_reader :name, :breed

    def initialize(name, breed)
        @name = name
        @breed = breed
    end

    def name=(name)
        @name = name
    end

    def to_s
        “#{@name}”
    end
end
```

attr_accessor

- Allows you to use symbols for read and write access of instance variables

```
class Dog

	attr_accessor :name, :breed

	def initialize(name, breed)
		@name = name
		@breed = breed
	end
end

dog = Dog.new(“Bomber”, “Labrador”)
puts “Dog's name is #{dog.name} of type #{dog.labrador}”
```

Virtual Attributes

- Methods that return attribute values in a different format. Nothing special.

```
class Book

	attr_accessor :title, :price

	def initialize(title, price)
		@title = title
		@price = price
	end

	def price_in_cents
		Integer(@price * 100 + 0.5)
	end
end
```

Access Control

- How much of the methods in a class are exposed to a user or another program
  - Public – can be called by anyone. No control is enforced (default)
  - Private – can be called only by the object instance itself
  - Protected – can be invoked only by objects of the defining class and its subclasses

Access Control

```
class MyClass

# default is public
	def method1
	end

private
	def method2
	end

protected
	def method3
	end

end
```

Inheritance

- A relation between two classes in a hierarchal manner (parent and child)
- Gains the power of the parent
- Can only inherit from one class

```
class Mammal
	def breathe
		puts “inhale and exhale”
	end
end

class Dog < Mammal
end

dog = Dog.new
puts dog.breathe
```

Conventions

Local Variable	Instance Variable	Class/Object	Constant
name	@name	Dog	PI
fish_and_chips	@fish_and_chips	Cat	FLEET_PER_MILE
x_axis	@x_axis	MyClass	DEBUG
_26	@_26	ActiveRecord	RATE

Gems

- Packaged Ruby classes/libraries/entire frameworks
  - Gives you more functionality in your applications
  - Extends the Ruby language
  - Makes life easier

RubyGems

- Command line tool to install gems in an external repository to be part of your “gemset”
  - gem install [gem_name]
- Download gems from multiple sources
  - http://rubygems.org
  - http://rubyforge.org
  - http://ruby-toolbox.com
- Example: Ruby on Rails (gem install rails)
- Places gems in your ruby “classpath”
  - In the case of rvm, it installs gems independently of a ruby installation

Installing Rails

- Rails can be installed as a gem with all the scripts/tools included (assuming you installed it with rvm)

```
gem install rails
```

Running Rails

- You should be able to create a new project by issuing the following command:

```
rails new [project_name]
cd [project_name]
rails server
```